F:\git\java\mar3\filemonitor\target\go-ethereum\go-ethereum-1.doc

0:F:\git\coin\ethereum\go-ethereum\common\hexutil\hexutil.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

/*
Package hexutil implements hex encoding with 0x prefix.
This encoding is used by the Ethereum RPC API to transport binary data in JSON payloads.

Encoding Rules

All hex data must have prefix "0x".

For byte slices, the hex data must be of even length. An empty byte slice
encodes as "0x".

Integers are encoded using the least amount of digits (no leading zero digits). Their
encoding may be of uneven length. The number zero encodes as "0x0".
*/
package hexutil

import (
"encoding/hex"
"fmt"
"math/big"
"strconv"
)

const uintBits = 32 << (uint64(^uint(0)) >> 63)

var (
ErrEmptyString   = &decError{"empty hex string"}
ErrSyntax   = &decError{"invalid hex string"}
ErrMissingPrefix = &decError{"hex string without 0x prefix"}
ErrOddLength   = &decError{"hex string of odd length"}
ErrEmptyNumber   = &decError{"hex string \"0x\""}
ErrLeadingZero   = &decError{"hex number with leading zero digits"}
ErrUint64Range   = &decError{"hex number > 64 bits"}
ErrUintRange   = &decError{fmt.Sprintf("hex number > %d bits", uintBits)}
ErrBig256Range   = &decError{"hex number > 256 bits"}
)

```
type decError struct{ msg string }

func (err decError) Error() string {
return string(err.msg)
}

// Decode decodes a hex string with 0x prefix.
func Decode(input string) ([]byte, error) {
if len(input) == 0 {
return nil, ErrEmptyString
}
if !has0xPrefix(input) {
return nil, ErrMissingPrefix
}
b, err := hex.DecodeString(input[2:])
if err != nil {
err = mapError(err)
}
return b, err
}

// MustDecode decodes a hex string with 0x prefix. It panics for invalid input.
func MustDecode(input string) []byte {
dec, err := Decode(input)
if err != nil {
panic(err)
}
return dec
}

// Encode encodes b as a hex string with 0x prefix.
func Encode(b []byte) string {
enc := make([]byte, len(b)*2+2)
copy(enc, "0x")
hex.Encode(enc[2:], b)
return string(enc)
}

// DecodeUint64 decodes a hex string with 0x prefix as a quantity.
func DecodeUint64(input string) (uint64, error) {
raw, err := checkNumber(input)
if err != nil {
```

```
return 0, err
}
dec, err := strconv.ParseUint(raw, 16, 64)
if err != nil {
err = mapError(err)
}
return dec, err
}

// MustDecodeUint64 decodes a hex string with 0x prefix as a quantity.
// It panics for invalid input.
func MustDecodeUint64(input string) uint64 {
dec, err := DecodeUint64(input)
if err != nil {
panic(err)
}
return dec
}

// EncodeUint64 encodes i as a hex string with 0x prefix.
func EncodeUint64(i uint64) string {
enc := make([]byte, 2, 10)
copy(enc, "0x")
return string(strconv.AppendUint(enc, i, 16))
}

var bigWordNibbles int

func init() {
// This is a weird way to compute the number of nibbles required for big.Word.
// The usual way would be to use constant arithmetic but go vet can't handle that.
b, _ := new(big.Int).SetString("FFFFFFFFFF", 16)
switch len(b.Bits()) {
case 1:
bigWordNibbles = 16
case 2:
bigWordNibbles = 8
default:
panic("weird big.Word size")
}
}
```

```
// DecodeBig decodes a hex string with 0x prefix as a quantity.
// Numbers larger than 256 bits are not accepted.
func DecodeBig(input string) (*big.Int, error) {
raw, err := checkNumber(input)
if err != nil {
return nil, err
}
if len(raw) > 64 {
return nil, ErrBig256Range
}
words := make([]big.Word, len(raw)/bigWordNibbles+1)
end := len(raw)
for i := range words {
start := end - bigWordNibbles
if start < 0 {
start = 0
}
for ri := start; ri < end; ri++ {
nib := decodeNibble(raw[ri])
if nib == badNibble {
return nil, ErrSyntax
}
words[i] *= 16
words[i] += big.Word(nib)
}
end = start
}
dec := new(big.Int).SetBits(words)
return dec, nil
}

// MustDecodeBig decodes a hex string with 0x prefix as a quantity.
// It panics for invalid input.
func MustDecodeBig(input string) *big.Int {
dec, err := DecodeBig(input)
if err != nil {
panic(err)
}
return dec
}

// EncodeBig encodes bigint as a hex string with 0x prefix.
```

```
// The sign of the integer is ignored.
func EncodeBig(bigint *big.Int) string {
nbits := bigint.BitLen()
if nbits == 0 {
return "0x0"
}
return fmt.Sprintf("%#x", bigint)
}

func has0xPrefix(input string) bool {
return len(input) >= 2 && input[0] == '0' && (input[1] == 'x' || input[1] == 'X')
}

func checkNumber(input string) (raw string, err error) {
if len(input) == 0 {
return "", ErrEmptyString
}
if !has0xPrefix(input) {
return "", ErrMissingPrefix
}
input = input[2:]
if len(input) == 0 {
return "", ErrEmptyNumber
}
if len(input) > 1 && input[0] == '0' {
return "", ErrLeadingZero
}
return input, nil
}

const badNibble = ^uint64(0)

func decodeNibble(in byte) uint64 {
switch {
case in >= '0' && in <= '9':
return uint64(in - '0')
case in >= 'A' && in <= 'F':
return uint64(in - 'A' + 10)
case in >= 'a' && in <= 'f':
return uint64(in - 'a' + 10)
default:
return badNibble
```

```
}
}

func mapError(err error) error {
if err, ok := err.(*strconv.NumError); ok {
switch err.Err {
case strconv.ErrRange:
return ErrUint64Range
case strconv.ErrSyntax:
return ErrSyntax
}
}
if _, ok := err.(hex.InvalidByteError); ok {
return ErrSyntax
}
if err == hex.ErrLength {
return ErrOddLength
}
return err
}

1:F:\git\coin\ethereum\go-ethereum\common\hexutil\hexutil_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package hexutil

import (
"bytes"
"math/big"
"testing"
)

type marshalTest struct {
input interface{}
want  string
}

type unmarshalTest struct {
input        string
want         interface{}
wantErr      error // if set, decoding must fail on any platform
wantErr32bit error // if set, decoding must fail on 32bit platforms (used for Uint tests)
```

```
}

var (
encodeBytesTests = []marshalTest{
{[]byte{}, "0x"},
{[]byte{0}, "0x00"},
{[]byte{0, 0, 1, 2}, "0x00000102"},
}

encodeBigTests = []marshalTest{
{referenceBig("0"), "0x0"},
{referenceBig("1"), "0x1"},
{referenceBig("ff"), "0xff"},
{referenceBig("112233445566778899aabbccddeeff"), "0x112233445566778899aabbccddeeff"},
{referenceBig("80a7f2c1bcc396c00"), "0x80a7f2c1bcc396c00"},
{referenceBig("-80a7f2c1bcc396c00"), "-0x80a7f2c1bcc396c00"},
}

encodeUint64Tests = []marshalTest{
{uint64(0), "0x0"},
{uint64(1), "0x1"},
{uint64(0xff), "0xff"},
{uint64(0x1122334455667788), "0x1122334455667788"},
}

encodeUintTests = []marshalTest{
{uint(0), "0x0"},
{uint(1), "0x1"},
{uint(0xff), "0xff"},
{uint(0x11223344), "0x11223344"},
}

decodeBytesTests = []unmarshalTest{
// invalid
{input: ``, wantErr: ErrEmptyString},
{input: `0`, wantErr: ErrMissingPrefix},
{input: `0x0`, wantErr: ErrOddLength},
{input: `0x023`, wantErr: ErrOddLength},
{input: `0xxx`, wantErr: ErrSyntax},
{input: `0x01zz01`, wantErr: ErrSyntax},
// valid
{input: `0x`, want: []byte{}},
```

```
{input: `0X`, want: []byte{}},
{input: `0x02`, want: []byte{0x02}},
{input: `0X02`, want: []byte{0x02}},
{input: `0xffffffffff`, want: []byte{0xff, 0xff, 0xff, 0xff, 0xff}},
{
input: `0xffffffffffffffffffffffffffffffffff`,
want:  []byte{0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff},
},
}

decodeBigTests = []unmarshalTest{
// invalid
{input: `0`, wantErr: ErrMissingPrefix},
{input: `0x`, wantErr: ErrEmptyNumber},
{input: `0x01`, wantErr: ErrLeadingZero},
{input: `0xx`, wantErr: ErrSyntax},
{input: `0x1zz01`, wantErr: ErrSyntax},
{
input:   `0x10000000000000000000000000000000000000000000000000000000000000000`,
wantErr: ErrBig256Range,
},
// valid
{input: `0x0`, want: big.NewInt(0)},
{input: `0x2`, want: big.NewInt(0x2)},
{input: `0x2F2`, want: big.NewInt(0x2f2)},
{input: `0X2F2`, want: big.NewInt(0x2f2)},
{input: `0x1122aaff`, want: big.NewInt(0x1122aaff)},
{input: `0xbBb`, want: big.NewInt(0xbbb)},
{input: `0xffffffff`, want: big.NewInt(0xffffffff)},
{
input: `0x112233445566778899aabbccddeeff`,
want:  referenceBig("112233445566778899aabbccddeeff"),
},
{
input: `0xffffffffffffffffffffffffffffffffff`,
want:  referenceBig("ffffffffffffffffffffffffffffffffff"),
},
{
input: `0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff`,
want:  referenceBig("ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"),
},
```

```
}

decodeUint64Tests = []unmarshalTest{
// invalid
{input: `0`, wantErr: ErrMissingPrefix},
{input: `0x`, wantErr: ErrEmptyNumber},
{input: `0x01`, wantErr: ErrLeadingZero},
{input: `0xffffffffffffffff`, wantErr: ErrUint64Range},
{input: `0xx`, wantErr: ErrSyntax},
{input: `0x1zz01`, wantErr: ErrSyntax},
// valid
{input: `0x0`, want: uint64(0)},
{input: `0x2`, want: uint64(0x2)},
{input: `0x2F2`, want: uint64(0x2f2)},
{input: `0X2F2`, want: uint64(0x2f2)},
{input: `0x1122aaff`, want: uint64(0x1122aaff)},
{input: `0xbbb`, want: uint64(0xbbb)},
{input: `0xffffffffffffffff`, want: uint64(0xffffffffffffffff)},
}
)

func TestEncode(t *testing.T) {
for _, test := range encodeBytesTests {
enc := Encode(test.input.([]byte))
if enc != test.want {
t.Errorf("input %x: wrong encoding %s", test.input, enc)
}
}
}

func TestDecode(t *testing.T) {
for _, test := range decodeBytesTests {
dec, err := Decode(test.input)
if !checkError(t, test.input, err, test.wantErr) {
continue
}
if !bytes.Equal(test.want.([]byte), dec) {
t.Errorf("input %s: value mismatch: got %x, want %x", test.input, dec, test.want)
continue
}
}
}
```

```
func TestEncodeBig(t *testing.T) {
for _, test := range encodeBigTests {
enc := EncodeBig(test.input.(*big.Int))
if enc != test.want {
t.Errorf("input %x: wrong encoding %s", test.input, enc)
}
}
}

func TestDecodeBig(t *testing.T) {
for _, test := range decodeBigTests {
dec, err := DecodeBig(test.input)
if !checkError(t, test.input, err, test.wantErr) {
continue
}
if dec.Cmp(test.want.(*big.Int)) != 0 {
t.Errorf("input %s: value mismatch: got %x, want %x", test.input, dec, test.want)
continue
}
}
}

func TestEncodeUint64(t *testing.T) {
for _, test := range encodeUint64Tests {
enc := EncodeUint64(test.input.(uint64))
if enc != test.want {
t.Errorf("input %x: wrong encoding %s", test.input, enc)
}
}
}

func TestDecodeUint64(t *testing.T) {
for _, test := range decodeUint64Tests {
dec, err := DecodeUint64(test.input)
if !checkError(t, test.input, err, test.wantErr) {
continue
}
if dec != test.want.(uint64) {
t.Errorf("input %s: value mismatch: got %x, want %x", test.input, dec, test.want)
continue
}
```

```
}
}

2:F:\git\coin\ethereum\go-ethereum\common\hexutil\json.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package hexutil

import (
"encoding/hex"
"encoding/json"
"fmt"
"math/big"
"reflect"
"strconv"
)

var (
textZero = []byte(`0x0`)
bytesT   = reflect.TypeOf(Bytes(nil))
bigT     = reflect.TypeOf((*Big)(nil))
uintT    = reflect.TypeOf(Uint(0))
uint64T  = reflect.TypeOf(Uint64(0))
)

// Bytes marshals/unmarshals as a JSON string with 0x prefix.
// The empty slice marshals as "0x".
type Bytes []byte

// MarshalText implements encoding.TextMarshaler
func (b Bytes) MarshalText() ([]byte, error) {
result := make([]byte, len(b)*2+2)
copy(result, `0x`)
hex.Encode(result[2:], b)
return result, nil
}

// UnmarshalJSON implements json.Unmarshaler.
func (b *Bytes) UnmarshalJSON(input []byte) error {
if !isString(input) {
return errNonString(bytesT)
}
```

```
return wrapTypeError(b.UnmarshalText(input[1:len(input)-1]), bytesT)
}

// UnmarshalText implements encoding.TextUnmarshaler.
func (b *Bytes) UnmarshalText(input []byte) error {
raw, err := checkText(input, true)
if err != nil {
return err
}
dec := make([]byte, len(raw)/2)
if _, err = hex.Decode(dec, raw); err != nil {
err = mapError(err)
} else {
*b = dec
}
return err
}

// String returns the hex encoding of b.
func (b Bytes) String() string {
return Encode(b)
}

// UnmarshalFixedJSON decodes the input as a string with 0x prefix. The length of out
// determines the required input length. This function is commonly used to implement the
// UnmarshalJSON method for fixed-size types.
func UnmarshalFixedJSON(typ reflect.Type, input, out []byte) error {
if !isString(input) {
return errNonString(typ)
}
return wrapTypeError(UnmarshalFixedText(typ.String(), input[1:len(input)-1], out), typ)
}

// UnmarshalFixedText decodes the input as a string with 0x prefix. The length of out
// determines the required input length. This function is commonly used to implement the
// UnmarshalText method for fixed-size types.
func UnmarshalFixedText(typname string, input, out []byte) error {
raw, err := checkText(input, true)
if err != nil {
return err
}
if len(raw)/2 != len(out) {
```

```
return fmt.Errorf("hex string has length %d, want %d for %s", len(raw), len(out)*2, typname)
}
// Pre-verify syntax before modifying out.
for _, b := range raw {
if decodeNibble(b) == badNibble {
return ErrSyntax
}
}
hex.Decode(out, raw)
return nil
}

// UnmarshalFixedUnprefixedText decodes the input as a string with optional 0x prefix. The
// length of out determines the required input length. This function is commonly used to
// implement the UnmarshalText method for fixed-size types.
func UnmarshalFixedUnprefixedText(typname string, input, out []byte) error {
raw, err := checkText(input, false)
if err != nil {
return err
}
if len(raw)/2 != len(out) {
return fmt.Errorf("hex string has length %d, want %d for %s", len(raw), len(out)*2, typname)
}
// Pre-verify syntax before modifying out.
for _, b := range raw {
if decodeNibble(b) == badNibble {
return ErrSyntax
}
}
hex.Decode(out, raw)
return nil
}

// Big marshals/unmarshals as a JSON string with 0x prefix.
// The zero value marshals as "0x0".
//
// Negative integers are not supported at this time. Attempting to marshal them will
// return an error. Values larger than 256bits are rejected by Unmarshal but will be
// marshaled without error.
type Big big.Int

// MarshalText implements encoding.TextMarshaler
```

```
func (b Big) MarshalText() ([]byte, error) {
return []byte(EncodeBig((*big.Int)(&b))), nil
}

// UnmarshalJSON implements json.Unmarshaler.
func (b *Big) UnmarshalJSON(input []byte) error {
if !isString(input) {
return errNonString(bigT)
}
return wrapTypeError(b.UnmarshalText(input[1:len(input)-1]), bigT)
}

// UnmarshalText implements encoding.TextUnmarshaler
func (b *Big) UnmarshalText(input []byte) error {
raw, err := checkNumberText(input)
if err != nil {
return err
}
if len(raw) > 64 {
return ErrBig256Range
}
words := make([]big.Word, len(raw)/bigWordNibbles+1)
end := len(raw)
for i := range words {
start := end - bigWordNibbles
if start < 0 {
start = 0
}
for ri := start; ri < end; ri++ {
nib := decodeNibble(raw[ri])
if nib == badNibble {
return ErrSyntax
}
words[i] *= 16
words[i] += big.Word(nib)
}
end = start
}
var dec big.Int
dec.SetBits(words)
*b = (Big)(dec)
return nil
```

```
}

// ToInt converts b to a big.Int.
func (b *Big) ToInt() *big.Int {
return (*big.Int)(b)
}

// String returns the hex encoding of b.
func (b *Big) String() string {
return EncodeBig(b.ToInt())
}

// Uint64 marshals/unmarshals as a JSON string with 0x prefix.
// The zero value marshals as "0x0".
type Uint64 uint64

// MarshalText implements encoding.TextMarshaler.
func (b Uint64) MarshalText() ([]byte, error) {
buf := make([]byte, 2, 10)
copy(buf, `0x`)
buf = strconv.AppendUint(buf, uint64(b), 16)
return buf, nil
}

// UnmarshalJSON implements json.Unmarshaler.
func (b *Uint64) UnmarshalJSON(input []byte) error {
if !isString(input) {
return errNonString(uint64T)
}
return wrapTypeError(b.UnmarshalText(input[1:len(input)-1]), uint64T)
}

// UnmarshalText implements encoding.TextUnmarshaler
func (b *Uint64) UnmarshalText(input []byte) error {
raw, err := checkNumberText(input)
if err != nil {
return err
}
if len(raw) > 16 {
return ErrUint64Range
}
var dec uint64
```

```
for _, byte := range raw {
nib := decodeNibble(byte)
if nib == badNibble {
return ErrSyntax
}
dec *= 16
dec += uint64(nib)
}
*b = Uint64(dec)
return nil
}

// String returns the hex encoding of b.
func (b Uint64) String() string {
return EncodeUint64(uint64(b))
}

// Uint marshals/unmarshals as a JSON string with 0x prefix.
// The zero value marshals as "0x0".
type Uint uint

// MarshalText implements encoding.TextMarshaler.
func (b Uint) MarshalText() ([]byte, error) {
return Uint64(b).MarshalText()
}

// UnmarshalJSON implements json.Unmarshaler.
func (b *Uint) UnmarshalJSON(input []byte) error {
if !isString(input) {
return errNonString(uintT)
}
return wrapTypeError(b.UnmarshalText(input[1:len(input)-1]), uintT)
}

// UnmarshalText implements encoding.TextUnmarshaler.
func (b *Uint) UnmarshalText(input []byte) error {
var u64 Uint64
err := u64.UnmarshalText(input)
if u64 > Uint64(^uint(0)) || err == ErrUint64Range {
return ErrUintRange
} else if err != nil {
return err
```

```
}
*b = Uint(u64)
return nil
}

// String returns the hex encoding of b.
func (b Uint) String() string {
return EncodeUint64(uint64(b))
}

func isString(input []byte) bool {
return len(input) >= 2 && input[0] == '"' && input[len(input)-1] == '"'
}

func bytesHave0xPrefix(input []byte) bool {
return len(input) >= 2 && input[0] == '0' && (input[1] == 'x' || input[1] == 'X')
}

func checkText(input []byte, wantPrefix bool) ([]byte, error) {
if len(input) == 0 {
return nil, nil // empty strings are allowed
}
if bytesHave0xPrefix(input) {
input = input[2:]
} else if wantPrefix {
return nil, ErrMissingPrefix
}
if len(input)%2 != 0 {
return nil, ErrOddLength
}
return input, nil
}

func checkNumberText(input []byte) (raw []byte, err error) {
if len(input) == 0 {
return nil, nil // empty strings are allowed
}
if !bytesHave0xPrefix(input) {
return nil, ErrMissingPrefix
}
input = input[2:]
if len(input) == 0 {
```

```
return nil, ErrEmptyNumber
}
if len(input) > 1 && input[0] == '0' {
return nil, ErrLeadingZero
}
return input, nil
}

func wrapTypeError(err error, typ reflect.Type) error {
if _, ok := err.(*decError); ok {
return &json.UnmarshalTypeError{Value: err.Error(), Type: typ}
}
return err
}

func errNonString(typ reflect.Type) error {
return &json.UnmarshalTypeError{Value: "non-string", Type: typ}
}

3:F:\git\coin\ethereum\go-ethereum\common\hexutil\json_example_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package hexutil_test

import (
"encoding/json"
"fmt"

"github.com/ethereum/go-ethereum/common/hexutil"
)

type MyType [5]byte

func (v *MyType) UnmarshalText(input []byte) error {
return hexutil.UnmarshalFixedText("MyType", input, v[:])
}

func (v MyType) String() string {
return hexutil.Bytes(v[:]).String()
}

func ExampleUnmarshalFixedText() {
```

```
var v1, v2 MyType
fmt.Println("v1 error:", json.Unmarshal([]byte(`"0x01"`), &v1))
fmt.Println("v2 error:", json.Unmarshal([]byte(`"0x0101010101"`), &v2))
fmt.Println("v2:", v2)
// Output:
// v1 error: hex string has length 2, want 10 for MyType
// v2 error: <nil>
// v2: 0x0101010101
}

4:F:\git\coin\ethereum\go-ethereum\common\hexutil\json_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package hexutil

import (
"bytes"
"encoding/hex"
"encoding/json"
"errors"
"math/big"
"testing"
)

func checkError(t *testing.T, input string, got, want error) bool {
if got == nil {
if want != nil {
t.Errorf("input %s: got no error, want %q", input, want)
return false
}
return true
}
if want == nil {
t.Errorf("input %s: unexpected error %q", input, got)
} else if got.Error() != want.Error() {
t.Errorf("input %s: got error %q, want %q", input, got, want)
}
return false
}

func referenceBig(s string) *big.Int {
b, ok := new(big.Int).SetString(s, 16)
```

```
if !ok {
panic("invalid")
}
return b
}

func referenceBytes(s string) []byte {
b, err := hex.DecodeString(s)
if err != nil {
panic(err)
}
return b
}

var errJSONEOF = errors.New("unexpected end of JSON input")

var unmarshalBytesTests = []unmarshalTest{
// invalid encoding
{input: "", wantErr: errJSONEOF},
{input: "null", wantErr: errNonString(bytesT)},
{input: "10", wantErr: errNonString(bytesT)},
{input: `"0"`, wantErr: wrapTypeError(ErrMissingPrefix, bytesT)},
{input: `"0x0"`, wantErr: wrapTypeError(ErrOddLength, bytesT)},
{input: `"0xxx"`, wantErr: wrapTypeError(ErrSyntax, bytesT)},
{input: `"0x01zz01"`, wantErr: wrapTypeError(ErrSyntax, bytesT)},

// valid encoding
{input: `""`, want: referenceBytes("")},
{input: `"0x"`, want: referenceBytes("")},
{input: `"0x02"`, want: referenceBytes("02")},
{input: `"0X02"`, want: referenceBytes("02")},
{input: `"0xffffffffff"`, want: referenceBytes("ffffffffff")},
{
input: `"0xffffffffffffffffffffffffffffffffffff"`,
want:  referenceBytes("ffffffffffffffffffffffffffffffffffff"),
},
}

func TestUnmarshalBytes(t *testing.T) {
for _, test := range unmarshalBytesTests {
var v Bytes
err := json.Unmarshal([]byte(test.input), &v)
```

```
if !checkError(t, test.input, err, test.wantErr) {
continue
}
if !bytes.Equal(test.want.([]byte), []byte(v)) {
t.Errorf("input %s: value mismatch: got %x, want %x", test.input, &v, test.want)
continue
}
}
}

func BenchmarkUnmarshalBytes(b *testing.B) {
input := []byte(`"0x123456789abcdef123456789abcdef"`)
for i := 0; i < b.N; i++ {
var v Bytes
if err := v.UnmarshalJSON(input); err != nil {
b.Fatal(err)
}
}
}

func TestMarshalBytes(t *testing.T) {
for _, test := range encodeBytesTests {
in := test.input.([]byte)
out, err := json.Marshal(Bytes(in))
if err != nil {
t.Errorf("%x: %v", in, err)
continue
}
if want := `"` + test.want + `"`; string(out) != want {
t.Errorf("%x: MarshalJSON output mismatch: got %q, want %q", in, out, want)
continue
}
if out := Bytes(in).String(); out != test.want {
t.Errorf("%x: String mismatch: got %q, want %q", in, out, test.want)
continue
}
}
}

var unmarshalBigTests = []unmarshalTest{
// invalid encoding
{input: "", wantErr: errJSONEOF},
```

```
{input: "null", wantErr: errNonString(bigT)},
{input: "10", wantErr: errNonString(bigT)},
{input: `"0"`, wantErr: wrapTypeError(ErrMissingPrefix, bigT)},
{input: `"0x"`, wantErr: wrapTypeError(ErrEmptyNumber, bigT)},
{input: `"0x01"`, wantErr: wrapTypeError(ErrLeadingZero, bigT)},
{input: `"0xx"`, wantErr: wrapTypeError(ErrSyntax, bigT)},
{input: `"0x1zz01"`, wantErr: wrapTypeError(ErrSyntax, bigT)},
{
input:   `"0x10000000000000000000000000000000000000000000000000000000000000000"`,
wantErr: wrapTypeError(ErrBig256Range, bigT),
},

// valid encoding
{input: `""`, want: big.NewInt(0)},
{input: `"0x0"`, want: big.NewInt(0)},
{input: `"0x2"`, want: big.NewInt(0x2)},
{input: `"0x2F2"`, want: big.NewInt(0x2f2)},
{input: `"0X2F2"`, want: big.NewInt(0x2f2)},
{input: `"0x1122aaff"`, want: big.NewInt(0x1122aaff)},
{input: `"0xbBb"`, want: big.NewInt(0xbbb)},
{input: `"0xffffffff"`, want: big.NewInt(0xffffffff)},
{
input: `"0x112233445566778899aabbccddeeff"`,
want:  referenceBig("112233445566778899aabbccddeeff"),
},
{
input: `"0xffffffffffffffffffffffffffffffffffff"`,
want:  referenceBig("ffffffffffffffffffffffffffffffffffff"),
},
{
input: `"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"`,
want:  referenceBig("ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"),
},
}

func TestUnmarshalBig(t *testing.T) {
for _, test := range unmarshalBigTests {
var v Big
err := json.Unmarshal([]byte(test.input), &v)
if !checkError(t, test.input, err, test.wantErr) {
continue
}
```

```
if test.want != nil && test.want.(*big.Int).Cmp((*big.Int)(&v)) != 0 {
t.Errorf("input %s: value mismatch: got %x, want %x", test.input, (*big.Int)(&v), test.want)
continue
}
}
}

func BenchmarkUnmarshalBig(b *testing.B) {
input := []byte(`"0x123456789abcdef123456789abcdef"`)
for i := 0; i < b.N; i++ {
var v Big
if err := v.UnmarshalJSON(input); err != nil {
b.Fatal(err)
}
}
}

func TestMarshalBig(t *testing.T) {
for _, test := range encodeBigTests {
in := test.input.(*big.Int)
out, err := json.Marshal((*Big)(in))
if err != nil {
t.Errorf("%d: %v", in, err)
continue
}
if want := `"` + test.want + `"`; string(out) != want {
t.Errorf("%d: MarshalJSON output mismatch: got %q, want %q", in, out, want)
continue
}
if out := (*Big)(in).String(); out != test.want {
t.Errorf("%x: String mismatch: got %q, want %q", in, out, test.want)
continue
}
}
}

var unmarshalUint64Tests = []unmarshalTest{
// invalid encoding
{input: "", wantErr: errJSONEOF},
{input: "null", wantErr: errNonString(uint64T)},
{input: "10", wantErr: errNonString(uint64T)},
{input: `"0"`, wantErr: wrapTypeError(ErrMissingPrefix, uint64T)},
```

```
{input: `"0x"`, wantErr: wrapTypeError(ErrEmptyNumber, uint64T)},
{input: `"0x01"`, wantErr: wrapTypeError(ErrLeadingZero, uint64T)},
{input: `"0xffffffffffffffff"`, wantErr: wrapTypeError(ErrUint64Range, uint64T)},
{input: `"0xx"`, wantErr: wrapTypeError(ErrSyntax, uint64T)},
{input: `"0x1zz01"`, wantErr: wrapTypeError(ErrSyntax, uint64T)},

// valid encoding
{input: `""`, want: uint64(0)},
{input: `"0x0"`, want: uint64(0)},
{input: `"0x2"`, want: uint64(0x2)},
{input: `"0x2F2"`, want: uint64(0x2f2)},
{input: `"0X2F2"`, want: uint64(0x2f2)},
{input: `"0x1122aaff"`, want: uint64(0x1122aaff)},
{input: `"0xbbb"`, want: uint64(0xbbb)},
{input: `"0xffffffffffffffff"`, want: uint64(0xffffffffffffffff)},
}

func TestUnmarshalUint64(t *testing.T) {
for _, test := range unmarshalUint64Tests {
var v Uint64
err := json.Unmarshal([]byte(test.input), &v)
if !checkError(t, test.input, err, test.wantErr) {
continue
}
if uint64(v) != test.want.(uint64) {
t.Errorf("input %s: value mismatch: got %d, want %d", test.input, v, test.want)
continue
}
}
}

func BenchmarkUnmarshalUint64(b *testing.B) {
input := []byte(`"0x123456789abcdf"`)
for i := 0; i < b.N; i++ {
var v Uint64
v.UnmarshalJSON(input)
}
}

func TestMarshalUint64(t *testing.T) {
for _, test := range encodeUint64Tests {
in := test.input.(uint64)
```

```
out, err := json.Marshal(Uint64(in))
if err != nil {
t.Errorf("%d: %v", in, err)
continue
}
if want := `"` + test.want + `"`; string(out) != want {
t.Errorf("%d: MarshalJSON output mismatch: got %q, want %q", in, out, want)
continue
}
if out := (Uint64)(in).String(); out != test.want {
t.Errorf("%x: String mismatch: got %q, want %q", in, out, test.want)
continue
}
}
}

func TestMarshalUint(t *testing.T) {
for _, test := range encodeUintTests {
in := test.input.(uint)
out, err := json.Marshal(Uint(in))
if err != nil {
t.Errorf("%d: %v", in, err)
continue
}
if want := `"` + test.want + `"`; string(out) != want {
t.Errorf("%d: MarshalJSON output mismatch: got %q, want %q", in, out, want)
continue
}
if out := (Uint)(in).String(); out != test.want {
t.Errorf("%x: String mismatch: got %q, want %q", in, out, test.want)
continue
}
}
}

var (
// These are variables (not constants) to avoid constant overflow
// checks in the compiler on 32bit platforms.
maxUint33bits = uint64(^uint32(0)) + 1
maxUint64bits = ^uint64(0)
)
```

```
var unmarshalUintTests = []unmarshalTest{
// invalid encoding
{input: "", wantErr: errJSONEOF},
{input: "null", wantErr: errNonString(uintT)},
{input: "10", wantErr: errNonString(uintT)},
{input: `"0"`, wantErr: wrapTypeError(ErrMissingPrefix, uintT)},
{input: `"0x"`, wantErr: wrapTypeError(ErrEmptyNumber, uintT)},
{input: `"0x01"`, wantErr: wrapTypeError(ErrLeadingZero, uintT)},
{input: `"0x100000000"`, want: uint(maxUint33bits), wantErr32bit: wrapTypeError(ErrUintRange,
uintT)},
{input: `"0xffffffffffffffff"`, wantErr: wrapTypeError(ErrUintRange, uintT)},
{input: `"0xx"`, wantErr: wrapTypeError(ErrSyntax, uintT)},
{input: `"0x1zz01"`, wantErr: wrapTypeError(ErrSyntax, uintT)},

// valid encoding
{input: `""`, want: uint(0)},
{input: `"0x0"`, want: uint(0)},
{input: `"0x2"`, want: uint(0x2)},
{input: `"0x2F2"`, want: uint(0x2f2)},
{input: `"0X2F2"`, want: uint(0x2f2)},
{input: `"0x1122aaff"`, want: uint(0x1122aaff)},
{input: `"0xbbb"`, want: uint(0xbbb)},
{input: `"0xffffffff"`, want: uint(0xffffffff)},
{input: `"0xffffffffffffffff"`, want: uint(maxUint64bits), wantErr32bit: wrapTypeError(ErrUintRange,
uintT)},
}

func TestUnmarshalUint(t *testing.T) {
for _, test := range unmarshalUintTests {
var v Uint
err := json.Unmarshal([]byte(test.input), &v)
if uintBits == 32 && test.wantErr32bit != nil {
checkError(t, test.input, err, test.wantErr32bit)
continue
}
if !checkError(t, test.input, err, test.wantErr) {
continue
}
if uint(v) != test.want.(uint) {
t.Errorf("input %s: value mismatch: got %d, want %d", test.input, v, test.want)
continue
}
```

```
}
}

func TestUnmarshalFixedUnprefixedText(t *testing.T) {
tests := []struct {
input   string
want    []byte
wantErr error
}{
{input: "0x2", wantErr: ErrOddLength},
{input: "2", wantErr: ErrOddLength},
{input: "4444", wantErr: errors.New("hex string has length 4, want 8 for x")},
{input: "4444", wantErr: errors.New("hex string has length 4, want 8 for x")},
// check that output is not modified for partially correct input
{input: "444444gg", wantErr: ErrSyntax, want: []byte{0, 0, 0, 0}},
{input: "0x444444gg", wantErr: ErrSyntax, want: []byte{0, 0, 0, 0}},
// valid inputs
{input: "44444444", want: []byte{0x44, 0x44, 0x44, 0x44}},
{input: "0x44444444", want: []byte{0x44, 0x44, 0x44, 0x44}},
}

for _, test := range tests {
out := make([]byte, 4)
err := UnmarshalFixedUnprefixedText("x", []byte(test.input), out)
switch {
case err == nil && test.wantErr != nil:
t.Errorf("%q: got no error, expected %q", test.input, test.wantErr)
case err != nil && test.wantErr == nil:
t.Errorf("%q: unexpected error %q", test.input, err)
case err != nil && err.Error() != test.wantErr.Error():
t.Errorf("%q: error mismatch: got %q, want %q", test.input, err, test.wantErr)
}
if test.want != nil && !bytes.Equal(out, test.want) {
t.Errorf("%q: output mismatch: got %x, want %x", test.input, out, test.want)
}
}
}

5:F:\git\coin\ethereum\go-ethereum\common\main_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package common
```

```
import (
"testing"

checker "gopkg.in/check.v1"
)

func Test(t *testing.T) { checker.TestingT(t) }

6:F:\git\coin\ethereum\go-ethereum\common\math\big.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package math provides integer math utilities.
package math

import (
"fmt"
"math/big"
)

var (
tt255     = BigPow(2, 255)
tt256     = BigPow(2, 256)
tt256m1   = new(big.Int).Sub(tt256, big.NewInt(1))
MaxBig256 = new(big.Int).Set(tt256m1)
tt63      = BigPow(2, 63)
MaxBig63  = new(big.Int).Sub(tt63, big.NewInt(1))
)

const (
// number of bits in a big.Word
wordBits = 32 << (uint64(^big.Word(0)) >> 63)
// number of bytes in a big.Word
wordBytes = wordBits / 8
)

// HexOrDecimal256 marshals big.Int as hex or decimal.
type HexOrDecimal256 big.Int

// UnmarshalText implements encoding.TextUnmarshaler.
func (i *HexOrDecimal256) UnmarshalText(input []byte) error {
bigint, ok := ParseBig256(string(input))
```

```
if !ok {
return fmt.Errorf("invalid hex or decimal integer %q", input)
}
*i = HexOrDecimal256(*bigint)
return nil
}

// MarshalText implements encoding.TextMarshaler.
func (i *HexOrDecimal256) MarshalText() ([]byte, error) {
if i == nil {
return []byte("0x0"), nil
}
return []byte(fmt.Sprintf("%#x", (*big.Int)(i))), nil
}

// ParseBig256 parses s as a 256 bit integer in decimal or hexadecimal syntax.
// Leading zeros are accepted. The empty string parses as zero.
func ParseBig256(s string) (*big.Int, bool) {
if s == "" {
return new(big.Int), true
}
var bigint *big.Int
var ok bool
if len(s) >= 2 && (s[:2] == "0x" || s[:2] == "0X") {
bigint, ok = new(big.Int).SetString(s[2:], 16)
} else {
bigint, ok = new(big.Int).SetString(s, 10)
}
if ok && bigint.BitLen() > 256 {
bigint, ok = nil, false
}
return bigint, ok
}

// MustParseBig parses s as a 256 bit big integer and panics if the string is invalid.
func MustParseBig256(s string) *big.Int {
v, ok := ParseBig256(s)
if !ok {
panic("invalid 256 bit integer: " + s)
}
return v
}
```

```
// BigPow returns a ** b as a big integer.
func BigPow(a, b int64) *big.Int {
r := big.NewInt(a)
return r.Exp(r, big.NewInt(b), nil)
}

// BigMax returns the larger of x or y.
func BigMax(x, y *big.Int) *big.Int {
if x.Cmp(y) < 0 {
return y
}
return x
}

// BigMin returns the smaller of x or y.
func BigMin(x, y *big.Int) *big.Int {
if x.Cmp(y) > 0 {
return y
}
return x
}

// FirstBitSet returns the index of the first 1 bit in v, counting from LSB.
func FirstBitSet(v *big.Int) int {
for i := 0; i < v.BitLen(); i++ {
if v.Bit(i) > 0 {
return i
}
}
return v.BitLen()
}

// PaddedBigBytes encodes a big integer as a big-endian byte slice. The length
// of the slice is at least n bytes.
func PaddedBigBytes(bigint *big.Int, n int) []byte {
if bigint.BitLen()/8 >= n {
return bigint.Bytes()
}
ret := make([]byte, n)
ReadBits(bigint, ret)
return ret
```

```
}

// bigEndianByteAt returns the byte at position n,
// in Big-Endian encoding
// So n==0 returns the least significant byte
func bigEndianByteAt(bigint *big.Int, n int) byte {
words := bigint.Bits()
// Check word-bucket the byte will reside in
i := n / wordBytes
if i >= len(words) {
return byte(0)
}
word := words[i]
// Offset of the byte
shift := 8 * uint(n%wordBytes)

return byte(word >> shift)
}

// Byte returns the byte at position n,
// with the supplied padlength in Little-Endian encoding.
// n==0 returns the MSB
// Example: bigint '5', padlength 32, n=31 => 5
func Byte(bigint *big.Int, padlength, n int) byte {
if n >= padlength {
return byte(0)
}
return bigEndianByteAt(bigint, padlength-1-n)
}

// ReadBits encodes the absolute value of bigint as big-endian bytes. Callers must ensure
// that buf has enough space. If buf is too short the result will be incomplete.
func ReadBits(bigint *big.Int, buf []byte) {
i := len(buf)
for _, d := range bigint.Bits() {
for j := 0; j < wordBytes && i > 0; j++ {
i--
buf[i] = byte(d)
d >>= 8
}
}
}
```

```
// U256 encodes as a 256 bit two's complement number. This operation is destructive.
func U256(x *big.Int) *big.Int {
return x.And(x, tt256m1)
}

// S256 interprets x as a two's complement number.
// x must not exceed 256 bits (the result is undefined if it does) and is not modified.
//
//   S256(0)        = 0
//   S256(1)        = 1
//   S256(2**255)   = -2**255
//   S256(2**256-1) = -1
func S256(x *big.Int) *big.Int {
if x.Cmp(tt255) < 0 {
return x
} else {
return new(big.Int).Sub(x, tt256)
}
}

// Exp implements exponentiation by squaring.
// Exp returns a newly-allocated big integer and does not change
// base or exponent. The result is truncated to 256 bits.
//
// Courtesy @karalabe and @chfast
func Exp(base, exponent *big.Int) *big.Int {
result := big.NewInt(1)

for _, word := range exponent.Bits() {
for i := 0; i < wordBits; i++ {
if word&1 == 1 {
U256(result.Mul(result, base))
}
U256(base.Mul(base, base))
word >>= 1
}
}
return result
}
```

7:F:\git\coin\ethereum\go-ethereum\common\math\big_test.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package math

import (
"bytes"
"encoding/hex"
"math/big"
"testing"

"github.com/ethereum/go-ethereum/common"
)

func TestHexOrDecimal256(t *testing.T) {
tests := []struct {
input string
num   *big.Int
ok    bool
}{
{"", big.NewInt(0), true},
{"0", big.NewInt(0), true},
{"0x0", big.NewInt(0), true},
{"12345678", big.NewInt(12345678), true},
{"0x12345678", big.NewInt(0x12345678), true},
{"0X12345678", big.NewInt(0x12345678), true},
// Tests for leading zero behaviour:
{"0123456789", big.NewInt(123456789), true}, // note: not octal
{"00", big.NewInt(0), true},
{"0x00", big.NewInt(0), true},
{"0x012345678abc", big.NewInt(0x12345678abc), true},
// Invalid syntax:
{"abcdef", nil, false},
{"0xgg", nil, false},
// Larger than 256 bits:
{"115792089237316195423570985008687907853269984665640564039457584007913129639936", nil, false},
}
for _, test := range tests {
var num HexOrDecimal256
err := num.UnmarshalText([]byte(test.input))
if (err == nil) != test.ok {
t.Errorf("ParseBig(%q) -> (err == nil) == %t, want %t", test.input, err == nil, test.ok)
```

```
continue
}
if test.num != nil && (*big.Int)(&num).Cmp(test.num) != 0 {
t.Errorf("ParseBig(%q) -> %d, want %d", test.input, (*big.Int)(&num), test.num)
}
}
}

func TestMustParseBig256(t *testing.T) {
defer func() {
if recover() == nil {
t.Error("MustParseBig should've panicked")
}
}()
MustParseBig256("ggg")
}

func TestBigMax(t *testing.T) {
a := big.NewInt(10)
b := big.NewInt(5)

max1 := BigMax(a, b)
if max1 != a {
t.Errorf("Expected %d got %d", a, max1)
}

max2 := BigMax(b, a)
if max2 != a {
t.Errorf("Expected %d got %d", a, max2)
}
}

func TestBigMin(t *testing.T) {
a := big.NewInt(10)
b := big.NewInt(5)

min1 := BigMin(a, b)
if min1 != b {
t.Errorf("Expected %d got %d", b, min1)
}

min2 := BigMin(b, a)
```

```
if min2 != b {
t.Errorf("Expected %d got %d", b, min2)
}
}

func TestFirstBigSet(t *testing.T) {
tests := []struct {
num *big.Int
ix  int
}{
{big.NewInt(0), 0},
{big.NewInt(1), 0},
{big.NewInt(2), 1},
{big.NewInt(0x100), 8},
}
for _, test := range tests {
if ix := FirstBitSet(test.num); ix != test.ix {
t.Errorf("FirstBitSet(b%b) = %d, want %d", test.num, ix, test.ix)
}
}
}

func TestPaddedBigBytes(t *testing.T) {
tests := []struct {
num    *big.Int
n      int
result []byte
}{
{num: big.NewInt(0), n: 4, result: []byte{0, 0, 0, 0}},
{num: big.NewInt(1), n: 4, result: []byte{0, 0, 0, 1}},
{num: big.NewInt(512), n: 4, result: []byte{0, 0, 2, 0}},
{num: BigPow(2, 32), n: 4, result: []byte{1, 0, 0, 0, 0}},
}
for _, test := range tests {
if result := PaddedBigBytes(test.num, test.n); !bytes.Equal(result, test.result) {
t.Errorf("PaddedBigBytes(%d, %d) = %v, want %v", test.num, test.n, result, test.result)
}
}
}

func BenchmarkPaddedBigBytesLargePadding(b *testing.B) {
bigint := MustParseBig256("123456789123456789123456789123456789")
```

```
for i := 0; i < b.N; i++ {
PaddedBigBytes(bigint, 200)
}
}

func BenchmarkPaddedBigBytesSmallPadding(b *testing.B) {
bigint :=
MustParseBig256("0x18F8F8F1000111000110011100222004330052300000000000000000FEFC
F3CC")
for i := 0; i < b.N; i++ {
PaddedBigBytes(bigint, 5)
}
}

func BenchmarkPaddedBigBytesSmallOnePadding(b *testing.B) {
bigint :=
MustParseBig256("0x18F8F8F1000111000110011100222004330052300000000000000000FEFC
F3CC")
for i := 0; i < b.N; i++ {
PaddedBigBytes(bigint, 32)
}
}

func BenchmarkByteAtBrandNew(b *testing.B) {
bigint :=
MustParseBig256("0x18F8F8F1000111000110011100222004330052300000000000000000FEFC
F3CC")
for i := 0; i < b.N; i++ {
bigEndianByteAt(bigint, 15)
}
}

func BenchmarkByteAt(b *testing.B) {
bigint :=
MustParseBig256("0x18F8F8F1000111000110011100222004330052300000000000000000FEFC
F3CC")
for i := 0; i < b.N; i++ {
bigEndianByteAt(bigint, 15)
}
}

func BenchmarkByteAtOld(b *testing.B) {
```

```
bigint :=
MustParseBig256("0x18F8F8F1000111000110011100222004330052300000000000000000FEFC
F3CC")
for i := 0; i < b.N; i++ {
PaddedBigBytes(bigint, 32)
}
}

func TestReadBits(t *testing.T) {
check := func(input string) {
want, _ := hex.DecodeString(input)
int, _ := new(big.Int).SetString(input, 16)
buf := make([]byte, len(want))
ReadBits(int, buf)
if !bytes.Equal(buf, want) {
t.Errorf("have: %x\nwant: %x", buf, want)
}
}
check("00000000000000000000000000000000000000000000000000FEFCF3F8F0")
check("00000000001234500000000000000000000000000000000000FEFCF3F8F0")
check("18F8F8F1000111000110011100222004330052300000000000000000FEFCF3F8F0")
}

func TestU256(t *testing.T) {
tests := []struct{ x, y *big.Int }{
{x: big.NewInt(0), y: big.NewInt(0)},
{x: big.NewInt(1), y: big.NewInt(1)},
{x: BigPow(2, 255), y: BigPow(2, 255)},
{x: BigPow(2, 256), y: big.NewInt(0)},
{x: new(big.Int).Add(BigPow(2, 256), big.NewInt(1)), y: big.NewInt(1)},
// negative values
{x: big.NewInt(-1), y: new(big.Int).Sub(BigPow(2, 256), big.NewInt(1))},
{x: big.NewInt(-2), y: new(big.Int).Sub(BigPow(2, 256), big.NewInt(2))},
{x: BigPow(2, -255), y: big.NewInt(1)},
}
for _, test := range tests {
if y := U256(new(big.Int).Set(test.x)); y.Cmp(test.y) != 0 {
t.Errorf("U256(%x) = %x, want %x", test.x, y, test.y)
}
}
}
```

```
func TestBigEndianByteAt(t *testing.T) {
tests := []struct {
x   string
y   int
exp byte
}{
{"00", 0, 0x00},
{"01", 1, 0x00},
{"00", 1, 0x00},
{"01", 0, 0x01},
{"000000000000000000000000000000000000000000000000000000000000102030", 0, 0x30},
{"000000000000000000000000000000000000000000000000000000000000102030", 1, 0x20},
{"ABCDEF090807060504030201000000000000000000000000000000000000000000", 31, 0xAB},
{"ABCDEF090807060504030201000000000000000000000000000000000000000000", 32, 0x00},
{"ABCDEF090807060504030201000000000000000000000000000000000000000000", 500, 0x00},
}
for _, test := range tests {
v := new(big.Int).SetBytes(common.Hex2Bytes(test.x))
actual := bigEndianByteAt(v, test.y)
if actual != test.exp {
t.Fatalf("Expected  [%v] %v:th byte to be %v, was %v.", test.x, test.y, test.exp, actual)
}

}
}
func TestLittleEndianByteAt(t *testing.T) {
tests := []struct {
x   string
y   int
exp byte
}{
{"00", 0, 0x00},
{"01", 1, 0x00},
{"00", 1, 0x00},
{"01", 0, 0x00},
{"000000000000000000000000000000000000000000000000000000000000102030", 0, 0x00},
{"000000000000000000000000000000000000000000000000000000000000102030", 1, 0x00},
{"ABCDEF090807060504030201000000000000000000000000000000000000000000", 31, 0x00},
{"ABCDEF090807060504030201000000000000000000000000000000000000000000", 32, 0x00},
{"ABCDEF090807060504030201000000000000000000000000000000000000000000", 0, 0xAB},
{"ABCDEF090807060504030201000000000000000000000000000000000000000000", 1, 0xCD},
```

```
{"00CDEF09080706050403020 1ffffffffffffffffffffffffffffffffffffffff", 0, 0x00},
{"00CDEF09080706050403020 1ffffffffffffffffffffffffffffffffffffffff", 1, 0xCD},
{"0000000000000000000000000000000000000000000000000000000000102030", 31, 0x30},
{"0000000000000000000000000000000000000000000000000000000000102030", 30, 0x20},
{"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", 32, 0x0},
{"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", 31, 0xFF},
{"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", 0xFFFF, 0x0},
}
for _, test := range tests {
v := new(big.Int).SetBytes(common.Hex2Bytes(test.x))
actual := Byte(v, 32, test.y)
if actual != test.exp {
t.Fatalf("Expected  [%v] %v:th byte to be %v, was %v.", test.x, test.y, test.exp, actual)
}

}
}

func TestS256(t *testing.T) {
tests := []struct{ x, y *big.Int }{
{x: big.NewInt(0), y: big.NewInt(0)},
{x: big.NewInt(1), y: big.NewInt(1)},
{x: big.NewInt(2), y: big.NewInt(2)},
{
x: new(big.Int).Sub(BigPow(2, 255), big.NewInt(1)),
y: new(big.Int).Sub(BigPow(2, 255), big.NewInt(1)),
},
{
x: BigPow(2, 255),
y: new(big.Int).Neg(BigPow(2, 255)),
},
{
x: new(big.Int).Sub(BigPow(2, 256), big.NewInt(1)),
y: big.NewInt(-1),
},
{
x: new(big.Int).Sub(BigPow(2, 256), big.NewInt(2)),
y: big.NewInt(-2),
},
}
for _, test := range tests {
if y := S256(test.x); y.Cmp(test.y) != 0 {
```

```
t.Errorf("S256(%x) = %x, want %x", test.x, y, test.y)
}
}
}

func TestExp(t *testing.T) {
tests := []struct{ base, exponent, result *big.Int }{
{base: big.NewInt(0), exponent: big.NewInt(0), result: big.NewInt(1)},
{base: big.NewInt(1), exponent: big.NewInt(0), result: big.NewInt(1)},
{base: big.NewInt(1), exponent: big.NewInt(1), result: big.NewInt(1)},
{base: big.NewInt(1), exponent: big.NewInt(2), result: big.NewInt(1)},
{base: big.NewInt(3), exponent: big.NewInt(144), result:
MustParseBig256("507528786056415600719754159741696356908742250191663887263627442114881")},
{base: big.NewInt(2), exponent: big.NewInt(255), result:
MustParseBig256("57896044618658097711785492504343953926634992332820282019728792003956564819968")},
}
for _, test := range tests {
if result := Exp(test.base, test.exponent); result.Cmp(test.result) != 0 {
t.Errorf("Exp(%d, %d) = %d, want %d", test.base, test.exponent, result, test.result)
}
}
}

8:F:\git\coin\ethereum\go-ethereum\common\math\integer.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package math

import (
"fmt"
"strconv"
)

const (
// Integer limit values.
MaxInt8   = 1<<7 - 1
MinInt8   = -1 << 7
MaxInt16  = 1<<15 - 1
MinInt16  = -1 << 15
MaxInt32  = 1<<31 - 1
```

```
MinInt32  = -1 << 31
MaxInt64  = 1<<63 - 1
MinInt64  = -1 << 63
MaxUint8  = 1<<8 - 1
MaxUint16 = 1<<16 - 1
MaxUint32 = 1<<32 - 1
MaxUint64 = 1<<64 - 1
)

// HexOrDecimal64 marshals uint64 as hex or decimal.
type HexOrDecimal64 uint64

// UnmarshalText implements encoding.TextUnmarshaler.
func (i *HexOrDecimal64) UnmarshalText(input []byte) error {
int, ok := ParseUint64(string(input))
if !ok {
return fmt.Errorf("invalid hex or decimal integer %q", input)
}
*i = HexOrDecimal64(int)
return nil
}

// MarshalText implements encoding.TextMarshaler.
func (i HexOrDecimal64) MarshalText() ([]byte, error) {
return []byte(fmt.Sprintf("%#x", uint64(i))), nil
}

// ParseUint64 parses s as an integer in decimal or hexadecimal syntax.
// Leading zeros are accepted. The empty string parses as zero.
func ParseUint64(s string) (uint64, bool) {
if s == "" {
return 0, true
}
if len(s) >= 2 && (s[:2] == "0x" || s[:2] == "0X") {
v, err := strconv.ParseUint(s[2:], 16, 64)
return v, err == nil
}
v, err := strconv.ParseUint(s, 10, 64)
return v, err == nil
}

// MustParseUint64 parses s as an integer and panics if the string is invalid.
```

```
func MustParseUint64(s string) uint64 {
v, ok := ParseUint64(s)
if !ok {
panic("invalid unsigned 64 bit integer: " + s)
}
return v
}

// NOTE: The following methods need to be optimised using either bit checking or asm

// SafeSub returns subtraction result and whether overflow occurred.
func SafeSub(x, y uint64) (uint64, bool) {
return x - y, x < y
}

// SafeAdd returns the result and whether overflow occurred.
func SafeAdd(x, y uint64) (uint64, bool) {
return x + y, y > MaxUint64-x
}

// SafeMul returns multiplication result and whether overflow occurred.
func SafeMul(x, y uint64) (uint64, bool) {
if x == 0 || y == 0 {
return 0, false
}
return x * y, y > MaxUint64/x
}

9:F:\git\coin\ethereum\go-ethereum\common\math\integer_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package math

import (
"testing"
)

type operation byte

const (
sub operation = iota
add
```

```
mul
)

func TestOverflow(t *testing.T) {
for i, test := range []struct {
x        uint64
y        uint64
overflow bool
op       operation
}{
// add operations
{MaxUint64, 1, true, add},
{MaxUint64 - 1, 1, false, add},

// sub operations
{0, 1, true, sub},
{0, 0, false, sub},

// mul operations
{0, 0, false, mul},
{10, 10, false, mul},
{MaxUint64, 2, true, mul},
{MaxUint64, 1, false, mul},
} {
var overflows bool
switch test.op {
case sub:
_, overflows = SafeSub(test.x, test.y)
case add:
_, overflows = SafeAdd(test.x, test.y)
case mul:
_, overflows = SafeMul(test.x, test.y)
}

if test.overflow != overflows {
t.Errorf("%d failed. Expected test to be %v, got %v", i, test.overflow, overflows)
}
}
}

func TestHexOrDecimal64(t *testing.T) {
tests := []struct {
```

```
input string
num   uint64
ok    bool
}{
{"", 0, true},
{"0", 0, true},
{"0x0", 0, true},
{"12345678", 12345678, true},
{"0x12345678", 0x12345678, true},
{"0X12345678", 0x12345678, true},
// Tests for leading zero behaviour:
{"0123456789", 123456789, true}, // note: not octal
{"0x00", 0, true},
{"0x012345678abc", 0x12345678abc, true},
// Invalid syntax:
{"abcdef", 0, false},
{"0xgg", 0, false},
// Doesn't fit into 64 bits:
{"18446744073709551617", 0, false},
}
for _, test := range tests {
var num HexOrDecimal64
err := num.UnmarshalText([]byte(test.input))
if (err == nil) != test.ok {
t.Errorf("ParseUint64(%q) -> (err == nil) = %t, want %t", test.input, err == nil, test.ok)
continue
}
if err == nil && uint64(num) != test.num {
t.Errorf("ParseUint64(%q) -> %d, want %d", test.input, num, test.num)
}
}
}

func TestMustParseUint64(t *testing.T) {
if v := MustParseUint64("12345"); v != 12345 {
t.Errorf(`MustParseUint64("12345") = %d, want 12345`, v)
}
}

func TestMustParseUint64Panic(t *testing.T) {
defer func() {
if recover() == nil {
```

```
t.Error("MustParseBig should've panicked")
}
}()
MustParseUint64("ggg")
}

10:F:\git\coin\ethereum\go-ethereum\common\mclock\mclock.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// package mclock is a wrapper for a monotonic clock source
package mclock

import (
"time"

"github.com/aristanetworks/goarista/monotime"
)

type AbsTime time.Duration // absolute monotonic time

func Now() AbsTime {
return AbsTime(monotime.Now())
}

11:F:\git\coin\ethereum\go-ethereum\common\number\int.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package number

import (
"math/big"

"github.com/ethereum/go-ethereum/common"
)

var tt256 = new(big.Int).Lsh(big.NewInt(1), 256)
var tt256m1 = new(big.Int).Sub(new(big.Int).Lsh(big.NewInt(1), 256), big.NewInt(1))
var tt255 = new(big.Int).Lsh(big.NewInt(1), 255)

func limitUnsigned256(x *Number) *Number {
x.num.And(x.num, tt256m1)
return x
```

```
}

func limitSigned256(x *Number) *Number {
if x.num.Cmp(tt255) < 0 {
return x
} else {
x.num.Sub(x.num, tt256)
return x
}
}

// Number function
type Initialiser func(n int64) *Number

// A Number represents a generic integer with a bounding function limiter. Limit is called after each operations
// to give "fake" bounded integers. New types of Number can be created through NewInitialiser returning a lambda
// with the new Initialiser.
type Number struct {
num   *big.Int
limit func(n *Number) *Number
}

// Returns a new initialiser for a new *Number without having to expose certain fields
func NewInitialiser(limiter func(*Number) *Number) Initialiser {
return func(n int64) *Number {
return &Number{big.NewInt(n), limiter}
}
}

// Return a Number with a UNSIGNED limiter up to 256 bits
func Uint256(n int64) *Number {
return &Number{big.NewInt(n), limitUnsigned256}
}

// Return a Number with a SIGNED limiter up to 256 bits
func Int256(n int64) *Number {
return &Number{big.NewInt(n), limitSigned256}
}

// Returns a Number with a SIGNED unlimited size
```

```
func Big(n int64) *Number {
return &Number{big.NewInt(n), func(x *Number) *Number { return x }}
}

// Sets i to sum of x+y
func (i *Number) Add(x, y *Number) *Number {
i.num.Add(x.num, y.num)
return i.limit(i)
}

// Sets i to difference of x-y
func (i *Number) Sub(x, y *Number) *Number {
i.num.Sub(x.num, y.num)
return i.limit(i)
}

// Sets i to product of x*y
func (i *Number) Mul(x, y *Number) *Number {
i.num.Mul(x.num, y.num)
return i.limit(i)
}

// Sets i to the quotient prodject of x/y
func (i *Number) Div(x, y *Number) *Number {
i.num.Div(x.num, y.num)
return i.limit(i)
}

// Sets i to x % y
func (i *Number) Mod(x, y *Number) *Number {
i.num.Mod(x.num, y.num)
return i.limit(i)
}

// Sets i to x << s
func (i *Number) Lsh(x *Number, s uint) *Number {
i.num.Lsh(x.num, s)
return i.limit(i)
}

// Sets i to x^y
func (i *Number) Pow(x, y *Number) *Number {
```

```
i.num.Exp(x.num, y.num, big.NewInt(0))
return i.limit(i)
}

// Setters

// Set x to i
func (i *Number) Set(x *Number) *Number {
i.num.Set(x.num)
return i.limit(i)
}

// Set x bytes to i
func (i *Number) SetBytes(x []byte) *Number {
i.num.SetBytes(x)
return i.limit(i)
}

// Cmp compares x and y and returns:
//
//     -1 if x <  y
//     0 if x == y
//     +1 if x >  y
func (i *Number) Cmp(x *Number) int {
return i.num.Cmp(x.num)
}

// Getters

// Returns the string representation of i
func (i *Number) String() string {
return i.num.String()
}

// Returns the byte representation of i
func (i *Number) Bytes() []byte {
return i.num.Bytes()
}

// Uint64 returns the Uint64 representation of x. If x cannot be represented in an int64, the result is undefined.
func (i *Number) Uint64() uint64 {
```

```
return i.num.Uint64()
}

// Int64 returns the int64 representation of x. If x cannot be represented in an int64, the result is undefined.
func (i *Number) Int64() int64 {
return i.num.Int64()
}

// Returns the signed version of i
func (i *Number) Int256() *Number {
return Int(0).Set(i)
}

// Returns the unsigned version of i
func (i *Number) Uint256() *Number {
return Uint(0).Set(i)
}

// Returns the index of the first bit that's set to 1
func (i *Number) FirstBitSet() int {
for j := 0; j < i.num.BitLen(); j++ {
if i.num.Bit(j) > 0 {
return j
}
}

return i.num.BitLen()
}

// Variables

var (
Zero       = Uint(0)
One        = Uint(1)
Two        = Uint(2)
MaxUint256 = Uint(0).SetBytes(common.Hex2Bytes("ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"))

MinOne = Int(-1)

// "typedefs"
Uint = Uint256
```

```
Int  = Int256
)

12:F:\git\coin\ethereum\go-ethereum\common\number\uint_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package number

import (
"math/big"
"testing"

"github.com/ethereum/go-ethereum/common"
)

func TestSet(t *testing.T) {
a := Uint(0)
b := Uint(10)
a.Set(b)
if a.num.Cmp(b.num) != 0 {
t.Error("didn't compare", a, b)
}

c := Uint(0).SetBytes(common.Hex2Bytes("0a"))
if c.num.Cmp(big.NewInt(10)) != 0 {
t.Error("c set bytes failed.")
}
}

func TestInitialiser(t *testing.T) {
check := false
init := NewInitialiser(func(x *Number) *Number {
check = true
return x
})
a := init(0).Add(init(1), init(2))
if a.Cmp(init(3)) != 0 {
t.Error("expected 3. got", a)
}
if !check {
t.Error("expected limiter to be called")
}
```

```
}

func TestGet(t *testing.T) {
a := Uint(10)
if a.Uint64() != 10 {
t.Error("expected to get 10. got", a.Uint64())
}

a = Uint(10)
if a.Int64() != 10 {
t.Error("expected to get 10. got", a.Int64())
}
}

func TestCmp(t *testing.T) {
a := Uint(10)
b := Uint(10)
c := Uint(11)

if a.Cmp(b) != 0 {
t.Error("a b == 0 failed", a, b)
}

if a.Cmp(c) >= 0 {
t.Error("a c < 0 failed", a, c)
}

if c.Cmp(b) <= 0 {
t.Error("c b > 0 failed", c, b)
}
}

func TestMaxArith(t *testing.T) {
a := Uint(0).Add(MaxUint256, One)
if a.Cmp(Zero) != 0 {
t.Error("expected max256 + 1 = 0 got", a)
}

a = Uint(0).Sub(Uint(0), One)
if a.Cmp(MaxUint256) != 0 {
t.Error("expected 0 - 1 = max256 got", a)
}
```

```
a = Int(0).Sub(Int(0), One)
if a.Cmp(MinOne) != 0 {
t.Error("expected 0 - 1 = -1 got", a)
}
}

func TestConversion(t *testing.T) {
a := Int(-1)
b := a.Uint256()
if b.Cmp(MaxUint256) != 0 {
t.Error("expected -1 => unsigned to return max. got", b)
}
}

13:F:\git\coin\ethereum\go-ethereum\common\path.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package common

import (
"fmt"
"os"
"path/filepath"
"runtime"
)

// MakeName creates a node name that follows the ethereum convention
// for such names. It adds the operation system name and Go runtime version
// the name.
func MakeName(name, version string) string {
return fmt.Sprintf("%s/v%s/%s/%s", name, version, runtime.GOOS, runtime.Version())
}

func FileExist(filePath string) bool {
_, err := os.Stat(filePath)
if err != nil && os.IsNotExist(err) {
return false
}

return true
}
```

```
func AbsolutePath(Datadir string, filename string) string {
if filepath.IsAbs(filename) {
return filename
}
return filepath.Join(Datadir, filename)
}

14:F:\git\coin\ethereum\go-ethereum\common\size.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package common

import (
"fmt"
)

type StorageSize float64

func (self StorageSize) String() string {
if self > 1000000 {
return fmt.Sprintf("%.2f mB", self/1000000)
} else if self > 1000 {
return fmt.Sprintf("%.2f kB", self/1000)
} else {
return fmt.Sprintf("%.2f B", self)
}
}

func (self StorageSize) Int64() int64 {
return int64(self)
}

15:F:\git\coin\ethereum\go-ethereum\common\size_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package common

import (
"testing"
)
```

```
func TestStorageSizeString(t *testing.T) {
tests := []struct {
size StorageSize
str  string
}{
{2381273, "2.38 mB"},
{2192, "2.19 kB"},
{12, "12.00 B"},
}

for _, test := range tests {
if test.size.String() != test.str {
t.Errorf("%f: got %q, want %q", float64(test.size), test.size.String(), test.str)
}
}
}

16:F:\git\coin\ethereum\go-ethereum\common\test_utils.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package common

import (
"encoding/json"
"fmt"
"io/ioutil"
)

// LoadJSON reads the given file and unmarshals its content.
func LoadJSON(file string, val interface{}) error {
content, err := ioutil.ReadFile(file)
if err != nil {
return err
}
if err := json.Unmarshal(content, val); err != nil {
if syntaxerr, ok := err.(*json.SyntaxError); ok {
line := findLine(content, syntaxerr.Offset)
return fmt.Errorf("JSON syntax error at %v:%v: %v", file, line, err)
}
return fmt.Errorf("JSON unmarshal error in %v: %v", file, err)
}
return nil
```

```
}

// findLine returns the line number for the given offset into data.
func findLine(data []byte, offset int64) (line int) {
line = 1
for i, r := range string(data) {
if int64(i) >= offset {
return
}
if r == '\n' {
line++
}
}
return
}

17:F:\git\coin\ethereum\go-ethereum\common\types.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package common

import (
"encoding/hex"
"fmt"
"math/big"
"math/rand"
"reflect"

"github.com/ethereum/go-ethereum/common/hexutil"
)

const (
HashLength    = 32
AddressLength = 20
)

var (
hashT    = reflect.TypeOf(Hash{})
addressT = reflect.TypeOf(Address{})
)

// Hash represents the 32 byte Keccak256 hash of arbitrary data.
```

```
type Hash [HashLength]byte

func BytesToHash(b []byte) Hash {
var h Hash
h.SetBytes(b)
return h
}
func StringToHash(s string) Hash { return BytesToHash([]byte(s)) }
func BigToHash(b *big.Int) Hash  { return BytesToHash(b.Bytes()) }
func HexToHash(s string) Hash    { return BytesToHash(FromHex(s)) }

// Get the string representation of the underlying hash
func (h Hash) Str() string   { return string(h[:]) }
func (h Hash) Bytes() []byte { return h[:] }
func (h Hash) Big() *big.Int { return new(big.Int).SetBytes(h[:]) }
func (h Hash) Hex() string   { return hexutil.Encode(h[:]) }

// TerminalString implements log.TerminalStringer, formatting a string for console
// output during logging.
func (h Hash) TerminalString() string {
return fmt.Sprintf("%x…%x", h[:3], h[29:])
}

// String implements the stringer interface and is used also by the logger when
// doing full logging into a file.
func (h Hash) String() string {
return h.Hex()
}

// Format implements fmt.Formatter, forcing the byte slice to be formatted as is,
// without going through the stringer interface used for logging.
func (h Hash) Format(s fmt.State, c rune) {
fmt.Fprintf(s, "%"+string(c), h[:])
}

// UnmarshalText parses a hash in hex syntax.
func (h *Hash) UnmarshalText(input []byte) error {
return hexutil.UnmarshalFixedText("Hash", input, h[:])
}

// UnmarshalJSON parses a hash in hex syntax.
func (h *Hash) UnmarshalJSON(input []byte) error {
```

```
return hexutil.UnmarshalFixedJSON(hashT, input, h[:])
}

// MarshalText returns the hex representation of h.
func (h Hash) MarshalText() ([]byte, error) {
return hexutil.Bytes(h[:]).MarshalText()
}

// Sets the hash to the value of b. If b is larger than len(h) it will panic
func (h *Hash) SetBytes(b []byte) {
if len(b) > len(h) {
b = b[len(b)-HashLength:]
}

copy(h[HashLength-len(b):], b)
}

// Set string `s` to h. If s is larger than len(h) it will panic
func (h *Hash) SetString(s string) { h.SetBytes([]byte(s)) }

// Sets h to other
func (h *Hash) Set(other Hash) {
for i, v := range other {
h[i] = v
}
}

// Generate implements testing/quick.Generator.
func (h Hash) Generate(rand *rand.Rand, size int) reflect.Value {
m := rand.Intn(len(h))
for i := len(h) - 1; i > m; i-- {
h[i] = byte(rand.Uint32())
}
return reflect.ValueOf(h)
}

func EmptyHash(h Hash) bool {
return h == Hash{}
}

// UnprefixedHash allows marshaling a Hash without 0x prefix.
type UnprefixedHash Hash
```

```
// UnmarshalText decodes the hash from hex. The 0x prefix is optional.
func (h *UnprefixedHash) UnmarshalText(input []byte) error {
return hexutil.UnmarshalFixedUnprefixedText("UnprefixedHash", input, h[:])
}

// MarshalText encodes the hash as hex.
func (h UnprefixedHash) MarshalText() ([]byte, error) {
return []byte(hex.EncodeToString(h[:])), nil
}

////////// Address

// Address represents the 20 byte address of an Ethereum account.
type Address [AddressLength]byte

func BytesToAddress(b []byte) Address {
var a Address
a.SetBytes(b)
return a
}
func StringToAddress(s string) Address { return BytesToAddress([]byte(s)) }
func BigToAddress(b *big.Int) Address  { return BytesToAddress(b.Bytes()) }
func HexToAddress(s string) Address    { return BytesToAddress(FromHex(s)) }

// IsHexAddress verifies whether a string can represent a valid hex-encoded
// Ethereum address or not.
func IsHexAddress(s string) bool {
if len(s) == 2+2*AddressLength && IsHex(s) {
return true
}
if len(s) == 2*AddressLength && IsHex("0x"+s) {
return true
}
return false
}

// Get the string representation of the underlying address
func (a Address) Str() string   { return string(a[:]) }
func (a Address) Bytes() []byte { return a[:] }
func (a Address) Big() *big.Int { return new(big.Int).SetBytes(a[:]) }
func (a Address) Hash() Hash    { return BytesToHash(a[:]) }
```

```
func (a Address) Hex() string   { return hexutil.Encode(a[:]) }

// String implements the stringer interface and is used also by the logger.
func (a Address) String() string {
return a.Hex()
}

// Format implements fmt.Formatter, forcing the byte slice to be formatted as is,
// without going through the stringer interface used for logging.
func (a Address) Format(s fmt.State, c rune) {
fmt.Fprintf(s, "%"+string(c), a[:])
}

// Sets the address to the value of b. If b is larger than len(a) it will panic
func (a *Address) SetBytes(b []byte) {
if len(b) > len(a) {
b = b[len(b)-AddressLength:]
}
copy(a[AddressLength-len(b):], b)
}

// Set string `s` to a. If s is larger than len(a) it will panic
func (a *Address) SetString(s string) { a.SetBytes([]byte(s)) }

// Sets a to other
func (a *Address) Set(other Address) {
for i, v := range other {
a[i] = v
}
}

// MarshalText returns the hex representation of a.
func (a Address) MarshalText() ([]byte, error) {
return hexutil.Bytes(a[:]).MarshalText()
}

// UnmarshalText parses a hash in hex syntax.
func (a *Address) UnmarshalText(input []byte) error {
return hexutil.UnmarshalFixedText("Address", input, a[:])
}

// UnmarshalJSON parses a hash in hex syntax.
```

```
func (a *Address) UnmarshalJSON(input []byte) error {
return hexutil.UnmarshalFixedJSON(addressT, input, a[:])
}

// UnprefixedHash allows marshaling an Address without 0x prefix.
type UnprefixedAddress Address

// UnmarshalText decodes the address from hex. The 0x prefix is optional.
func (a *UnprefixedAddress) UnmarshalText(input []byte) error {
return hexutil.UnmarshalFixedUnprefixedText("UnprefixedAddress", input, a[:])
}

// MarshalText encodes the address as hex.
func (a UnprefixedAddress) MarshalText() ([]byte, error) {
return []byte(hex.EncodeToString(a[:])), nil
}

18:F:\git\coin\ethereum\go-ethereum\common\types_template.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// +build none
//sed -e 's/_N_/Hash/g' -e 's/_S_/32/g' -e '1d' types_template.go | gofmt -w hash.go

package common

import "math/big"

type _N_ [_S_]byte

func BytesTo_N_(b []byte) _N_ {
var h _N_
h.SetBytes(b)
return h
}
func StringTo_N_(s string) _N_ { return BytesTo_N_([]byte(s)) }
func BigTo_N_(b *big.Int) _N_  { return BytesTo_N_(b.Bytes()) }
func HexTo_N_(s string) _N_    { return BytesTo_N_(FromHex(s)) }

// Don't use the default 'String' method in case we want to overwrite

// Get the string representation of the underlying hash
func (h _N_) Str() string   { return string(h[:]) }
```

```
func (h _N_) Bytes() []byte { return h[:] }
func (h _N_) Big() *big.Int { return new(big.Int).SetBytes(h[:]) }
func (h _N_) Hex() string   { return "0x" + Bytes2Hex(h[:]) }

// Sets the hash to the value of b. If b is larger than len(h) it will panic
func (h *_N_) SetBytes(b []byte) {
// Use the right most bytes
if len(b) > len(h) {
b = b[len(b)-_S_:]
}

// Reverse the loop
for i := len(b) - 1; i >= 0; i-- {
h[_S_-len(b)+i] = b[i]
}
}

// Set string `s` to h. If s is larger than len(h) it will panic
func (h *_N_) SetString(s string) { h.SetBytes([]byte(s)) }

// Sets h to other
func (h *_N_) Set(other _N_) {
for i, v := range other {
h[i] = v
}
}

19:F:\git\coin\ethereum\go-ethereum\common\types_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package common

import (
"encoding/json"
"math/big"
"strings"
"testing"
)

func TestBytesConversion(t *testing.T) {
bytes := []byte{5}
hash := BytesToHash(bytes)
```

```
var exp Hash
exp[31] = 5

if hash != exp {
t.Errorf("expected %x got %x", exp, hash)
}
}

func TestHashJsonValidation(t *testing.T) {
var tests = []struct {
Prefix string
Size   int
Error  string
}{
{"", 62, "json: cannot unmarshal hex string without 0x prefix into Go value of type common.Hash"},
{"0x", 66, "hex string has length 66, want 64 for common.Hash"},
{"0x", 63, "json: cannot unmarshal hex string of odd length into Go value of type common.Hash"},
{"0x", 0, "hex string has length 0, want 64 for common.Hash"},
{"0x", 64, ""},
{"0X", 64, ""},
}
for _, test := range tests {
input := `"` + test.Prefix + strings.Repeat("0", test.Size) + `"`
var v Hash
err := json.Unmarshal([]byte(input), &v)
if err == nil {
if test.Error != "" {
t.Errorf("%s: error mismatch: have nil, want %q", input, test.Error)
}
} else {
if err.Error() != test.Error {
t.Errorf("%s: error mismatch: have %q, want %q", input, err, test.Error)
}
}
}
}

func TestAddressUnmarshalJSON(t *testing.T) {
var tests = []struct {
Input     string
ShouldErr bool
```

```
Output    *big.Int
}{
{"", true, nil},
{`""`, true, nil},
{`"0x"`, true, nil},
{`"0x00"`, true, nil},
{`"0xG000000000000000000000000000000000000000"`, true, nil},
{`"0x0000000000000000000000000000000000000000"`, false, big.NewInt(0)},
{`"0x0000000000000000000000000000000000000010"`, false, big.NewInt(16)},
}
for i, test := range tests {
var v Address
err := json.Unmarshal([]byte(test.Input), &v)
if err != nil && !test.ShouldErr {
t.Errorf("test #%d: unexpected error: %v", i, err)
}
if err == nil {
if test.ShouldErr {
t.Errorf("test #%d: expected error, got none", i)
}
if v.Big().Cmp(test.Output) != 0 {
t.Errorf("test #%d: address mismatch: have %v, want %v", i, v.Big(), test.Output)
}
}
}
}
```

20:F:\git\coin\ethereum\go-ethereum\compression\rle\read_write.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package rle implements the run-length encoding used for Ethereum data.
package rle

import (
"bytes"
"errors"

"github.com/ethereum/go-ethereum/crypto"
)

const (
token           byte = 0xfe
```

```
emptyShaToken          = 0xfd
emptyListShaToken      = 0xfe
tokenToken             = 0xff
)

var empty = crypto.Keccak256([]byte(""))
var emptyList = crypto.Keccak256([]byte{0x80})

func Decompress(dat []byte) ([]byte, error) {
buf := new(bytes.Buffer)

for i := 0; i < len(dat); i++ {
if dat[i] == token {
if i+1 < len(dat) {
switch dat[i+1] {
case emptyShaToken:
buf.Write(empty)
case emptyListShaToken:
buf.Write(emptyList)
case tokenToken:
buf.WriteByte(token)
default:
buf.Write(make([]byte, int(dat[i+1]-2)))
}
i++
} else {
return nil, errors.New("error reading bytes. token encountered without proceeding bytes")
}
} else {
buf.WriteByte(dat[i])
}
}

return buf.Bytes(), nil
}

func compressChunk(dat []byte) (ret []byte, n int) {
switch {
case dat[0] == token:
return []byte{token, tokenToken}, 1
case len(dat) > 1 && dat[0] == 0x0 && dat[1] == 0x0:
j := 0
```

```
for j <= 254 && j < len(dat) {
if dat[j] != 0 {
break
}
j++
}
return []byte{token, byte(j + 2)}, j
case len(dat) >= 32:
if dat[0] == empty[0] && bytes.Equal(dat[:32], empty) {
return []byte{token, emptyShaToken}, 32
} else if dat[0] == emptyList[0] && bytes.Equal(dat[:32], emptyList) {
return []byte{token, emptyListShaToken}, 32
}
fallthrough
default:
return dat[:1], 1
}
}

func Compress(dat []byte) []byte {
buf := new(bytes.Buffer)

i := 0
for i < len(dat) {
b, n := compressChunk(dat[i:])
buf.Write(b)
i += n
}

return buf.Bytes()
}

21:F:\git\coin\ethereum\go-ethereum\compression\rle\read_write_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package rle

import (
"testing"

checker "gopkg.in/check.v1"
)
```

```
func Test(t *testing.T) { checker.TestingT(t) }

type CompressionRleSuite struct{}

var _ = checker.Suite(&CompressionRleSuite{})

func (s *CompressionRleSuite) TestDecompressSimple(c *checker.C) {
exp := []byte{0xc5, 0xd2, 0x46, 0x1, 0x86, 0xf7, 0x23, 0x3c, 0x92, 0x7e, 0x7d, 0xb2, 0xdc, 0xc7,
0x3, 0xc0, 0xe5, 0x0, 0xb6, 0x53, 0xca, 0x82, 0x27, 0x3b, 0x7b, 0xfa, 0xd8, 0x4, 0x5d, 0x85,
0xa4, 0x70}
res, err := Decompress([]byte{token, 0xfd})
c.Assert(err, checker.IsNil)
c.Assert(res, checker.DeepEquals, exp)

exp = []byte{0x56, 0xe8, 0x1f, 0x17, 0x1b, 0xcc, 0x55, 0xa6, 0xff, 0x83, 0x45, 0xe6, 0x92, 0xc0,
0xf8, 0x6e, 0x5b, 0x48, 0xe0, 0x1b, 0x99, 0x6c, 0xad, 0xc0, 0x1, 0x62, 0x2f, 0xb5, 0xe3, 0x63,
0xb4, 0x21}
res, err = Decompress([]byte{token, 0xfe})
c.Assert(err, checker.IsNil)
c.Assert(res, checker.DeepEquals, exp)

res, err = Decompress([]byte{token, 0xff})
c.Assert(err, checker.IsNil)
c.Assert(res, checker.DeepEquals, []byte{token})

res, err = Decompress([]byte{token, 12})
c.Assert(err, checker.IsNil)
c.Assert(res, checker.DeepEquals, make([]byte, 10))

}

22:F:\git\coin\ethereum\go-ethereum\consensus\clique\api.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package clique

import (
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/rpc"
```

```
)

// API is a user facing RPC API to allow controlling the signer and voting
// mechanisms of the proof-of-authority scheme.
type API struct {
chain  consensus.ChainReader
clique *Clique
}

// GetSnapshot retrieves the state snapshot at a given block.
func (api *API) GetSnapshot(number *rpc.BlockNumber) (*Snapshot, error) {
// Retrieve the requested block number (or current if none requested)
var header *types.Header
if number == nil || *number == rpc.LatestBlockNumber {
header = api.chain.CurrentHeader()
} else {
header = api.chain.GetHeaderByNumber(uint64(number.Int64()))
}
// Ensure we have an actually valid block and return its snapshot
if header == nil {
return nil, errUnknownBlock
}
return api.clique.snapshot(api.chain, header.Number.Uint64(), header.Hash(), nil)
}

// GetSnapshotAtHash retrieves the state snapshot at a given block.
func (api *API) GetSnapshotAtHash(hash common.Hash) (*Snapshot, error) {
header := api.chain.GetHeaderByHash(hash)
if header == nil {
return nil, errUnknownBlock
}
return api.clique.snapshot(api.chain, header.Number.Uint64(), header.Hash(), nil)
}

// GetSigners retrieves the list of authorized signers at the specified block.
func (api *API) GetSigners(number *rpc.BlockNumber) ([]common.Address, error) {
// Retrieve the requested block number (or current if none requested)
var header *types.Header
if number == nil || *number == rpc.LatestBlockNumber {
header = api.chain.CurrentHeader()
} else {
header = api.chain.GetHeaderByNumber(uint64(number.Int64()))
```

```
}
// Ensure we have an actually valid block and return the signers from its snapshot
if header == nil {
return nil, errUnknownBlock
}
snap, err := api.clique.snapshot(api.chain, header.Number.Uint64(), header.Hash(), nil)
if err != nil {
return nil, err
}
return snap.signers(), nil
}

// GetSignersAtHash retrieves the state snapshot at a given block.
func (api *API) GetSignersAtHash(hash common.Hash) ([]common.Address, error) {
header := api.chain.GetHeaderByHash(hash)
if header == nil {
return nil, errUnknownBlock
}
snap, err := api.clique.snapshot(api.chain, header.Number.Uint64(), header.Hash(), nil)
if err != nil {
return nil, err
}
return snap.signers(), nil
}

// Proposals returns the current proposals the node tries to uphold and vote on.
func (api *API) Proposals() map[common.Address]bool {
api.clique.lock.RLock()
defer api.clique.lock.RUnlock()

proposals := make(map[common.Address]bool)
for address, auth := range api.clique.proposals {
proposals[address] = auth
}
return proposals
}

// Propose injects a new authorization proposal that the signer will attempt to
// push through.
func (api *API) Propose(address common.Address, auth bool) {
api.clique.lock.Lock()
defer api.clique.lock.Unlock()
```

```
api.clique.proposals[address] = auth
}

// Discard drops a currently running proposal, stopping the signer from casting
// further votes (either for or against).
func (api *API) Discard(address common.Address) {
api.clique.lock.Lock()
defer api.clique.lock.Unlock()

delete(api.clique.proposals, address)
}

23:F:\git\coin\ethereum\go-ethereum\consensus\clique\clique.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package clique implements the proof-of-authority consensus engine.
package clique

import (
"bytes"
"errors"
"math/big"
"math/rand"
"sync"
"time"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/hexutil"
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/crypto/sha3"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/params"
"github.com/ethereum/go-ethereum/rlp"
"github.com/ethereum/go-ethereum/rpc"
lru "github.com/hashicorp/golang-lru"
)
```

```
const (
checkpointInterval = 1024 // Number of blocks after which to save the vote snapshot to the database
inmemorySnapshots  = 128  // Number of recent vote snapshots to keep in memory
inmemorySignatures = 4096 // Number of recent block signatures to keep in memory

wiggleTime = 500 * time.Millisecond // Random delay (per signer) to allow concurrent signers
)

// Clique proof-of-authority protocol constants.
var (
epochLength = uint64(30000) // Default number of blocks after which to checkpoint and reset the pending votes
blockPeriod = uint64(15)    // Default minimum difference between two consecutive block's timestamps

extraVanity = 32 // Fixed number of extra-data prefix bytes reserved for signer vanity
extraSeal   = 65 // Fixed number of extra-data suffix bytes reserved for signer seal

nonceAuthVote = hexutil.MustDecode("0xffffffffffffffff") // Magic nonce number to vote on adding a new signer
nonceDropVote = hexutil.MustDecode("0x0000000000000000") // Magic nonce number to vote on removing a signer.

uncleHash = types.CalcUncleHash(nil) // Always Keccak256(RLP([])) as uncles are meaningless outside of PoW.

diffInTurn = big.NewInt(2) // Block difficulty for in-turn signatures
diffNoTurn = big.NewInt(1) // Block difficulty for out-of-turn signatures
)

// Various error messages to mark blocks invalid. These should be private to
// prevent engine specific errors from being referenced in the remainder of the
// codebase, inherently breaking if the engine is swapped out. Please put common
// error types into the consensus package.
var (
// errUnknownBlock is returned when the list of signers is requested for a block
// that is not part of the local blockchain.
errUnknownBlock = errors.New("unknown block")

// errInvalidCheckpointBeneficiary is returned if a checkpoint/epoch transition
```

```
// block has a beneficiary set to non-zeroes.
errInvalidCheckpointBeneficiary = errors.New("beneficiary in checkpoint block non-zero")

// errInvalidVote is returned if a nonce value is something else that the two
// allowed constants of 0x00..0 or 0xff..f.
errInvalidVote = errors.New("vote nonce not 0x00..0 or 0xff..f")

// errInvalidCheckpointVote is returned if a checkpoint/epoch transition block
// has a vote nonce set to non-zeroes.
errInvalidCheckpointVote = errors.New("vote nonce in checkpoint block non-zero")

// errMissingVanity is returned if a block's extra-data section is shorter than
// 32 bytes, which is required to store the signer vanity.
errMissingVanity = errors.New("extra-data 32 byte vanity prefix missing")

// errMissingSignature is returned if a block's extra-data section doesn't seem
// to contain a 65 byte secp256k1 signature.
errMissingSignature = errors.New("extra-data 65 byte suffix signature missing")

// errExtraSigners is returned if non-checkpoint block contain signer data in
// their extra-data fields.
errExtraSigners = errors.New("non-checkpoint block contains extra signer list")

// errInvalidCheckpointSigners is returned if a checkpoint block contains an
// invalid list of signers (i.e. non divisible by 20 bytes, or not the correct
// ones).
errInvalidCheckpointSigners = errors.New("invalid signer list on checkpoint block")

// errInvalidMixDigest is returned if a block's mix digest is non-zero.
errInvalidMixDigest = errors.New("non-zero mix digest")

// errInvalidUncleHash is returned if a block contains an non-empty uncle list.
errInvalidUncleHash = errors.New("non empty uncle hash")

// errInvalidDifficulty is returned if the difficulty of a block is not either
// of 1 or 2, or if the value does not match the turn of the signer.
errInvalidDifficulty = errors.New("invalid difficulty")

// ErrInvalidTimestamp is returned if the timestamp of a block is lower than
// the previous block's timestamp + the minimum block period.
ErrInvalidTimestamp = errors.New("invalid timestamp")
```

```
// errInvalidVotingChain is returned if an authorization list is attempted to
// be modified via out-of-range or non-contiguous headers.
errInvalidVotingChain = errors.New("invalid voting chain")

// errUnauthorized is returned if a header is signed by a non-authorized entity.
errUnauthorized = errors.New("unauthorized")
)

// SignerFn is a signer callback function to request a hash to be signed by a
// backing account.
type SignerFn func(accounts.Account, []byte) ([]byte, error)

// sigHash returns the hash which is used as input for the proof-of-authority
// signing. It is the hash of the entire header apart from the 65 byte signature
// contained at the end of the extra data.
//
// Note, the method requires the extra data to be at least 65 bytes, otherwise it
// panics. This is done to avoid accidentally using both forms (signature present
// or not), which could be abused to produce different hashes for the same header.
func sigHash(header *types.Header) (hash common.Hash) {
hasher := sha3.NewKeccak256()

rlp.Encode(hasher, []interface{}{
header.ParentHash,
header.UncleHash,
header.Coinbase,
header.Root,
header.TxHash,
header.ReceiptHash,
header.Bloom,
header.Difficulty,
header.Number,
header.GasLimit,
header.GasUsed,
header.Time,
header.Extra[:len(header.Extra)-65], // Yes, this will panic if extra is too short
header.MixDigest,
header.Nonce,
})
hasher.Sum(hash[:0])
return hash
}
```

```
// ecrecover extracts the Ethereum account address from a signed header.
func ecrecover(header *types.Header, sigcache *lru.ARCCache) (common.Address, error) {
// If the signature's already cached, return that
hash := header.Hash()
if address, known := sigcache.Get(hash); known {
return address.(common.Address), nil
}
// Retrieve the signature from the header extra-data
if len(header.Extra) < extraSeal {
return common.Address{}, errMissingSignature
}
signature := header.Extra[len(header.Extra)-extraSeal:]

// Recover the public key and the Ethereum address
pubkey, err := crypto.Ecrecover(sigHash(header).Bytes(), signature)
if err != nil {
return common.Address{}, err
}
var signer common.Address
copy(signer[:], crypto.Keccak256(pubkey[1:])[12:])

sigcache.Add(hash, signer)
return signer, nil
}

// Clique is the proof-of-authority consensus engine proposed to support the
// Ethereum testnet following the Ropsten attacks.
type Clique struct {
config *params.CliqueConfig // Consensus engine configuration parameters
db     ethdb.Database       // Database to store and retrieve snapshot checkpoints

recents    *lru.ARCCache // Snapshots for recent block to speed up reorgs
signatures *lru.ARCCache // Signatures of recent blocks to speed up mining

proposals map[common.Address]bool // Current list of proposals we are pushing

signer common.Address // Ethereum address of the signing key
signFn SignerFn       // Signer function to authorize hashes with
lock   sync.RWMutex   // Protects the signer fields
}
```

```
// New creates a Clique proof-of-authority consensus engine with the initial
// signers set to the ones provided by the user.
func New(config *params.CliqueConfig, db ethdb.Database) *Clique {
// Set any missing consensus parameters to their defaults
conf := *config
if conf.Epoch == 0 {
conf.Epoch = epochLength
}
if conf.Period == 0 {
conf.Period = blockPeriod
}
// Allocate the snapshot caches and create the engine
recents, _ := lru.NewARC(inmemorySnapshots)
signatures, _ := lru.NewARC(inmemorySignatures)

return &Clique{
config:     &conf,
db:         db,
recents:    recents,
signatures: signatures,
proposals:  make(map[common.Address]bool),
}
}

// Author implements consensus.Engine, returning the Ethereum address recovered
// from the signature in the header's extra-data section.
func (c *Clique) Author(header *types.Header) (common.Address, error) {
return ecrecover(header, c.signatures)
}

// VerifyHeader checks whether a header conforms to the consensus rules.
func (c *Clique) VerifyHeader(chain consensus.ChainReader, header *types.Header, seal bool)
error {
return c.verifyHeader(chain, header, nil)
}

// VerifyHeaders is similar to VerifyHeader, but verifies a batch of headers. The
// method returns a quit channel to abort the operations and a results channel to
// retrieve the async verifications (the order is that of the input slice).
func (c *Clique) VerifyHeaders(chain consensus.ChainReader, headers []*types.Header, seals
[]bool) (chan<- struct{}, <-chan error) {
abort := make(chan struct{})
```

```
results := make(chan error, len(headers))

go func() {
for i, header := range headers {
err := c.verifyHeader(chain, header, headers[:i])

select {
case <-abort:
return
case results <- err:
}
}
}()
return abort, results
}

// verifyHeader checks whether a header conforms to the consensus rules.The
// caller may optionally pass in a batch of parents (ascending order) to avoid
// looking those up from the database. This is useful for concurrently verifying
// a batch of new headers.
func (c *Clique) verifyHeader(chain consensus.ChainReader, header *types.Header, parents
[]*types.Header) error {
if header.Number == nil {
return errUnknownBlock
}
number := header.Number.Uint64()

// Don't waste time checking blocks from the future
if header.Time.Cmp(big.NewInt(time.Now().Unix())) > 0 {
return consensus.ErrFutureBlock
}
// Checkpoint blocks need to enforce zero beneficiary
checkpoint := (number % c.config.Epoch) == 0
if checkpoint && header.Coinbase != (common.Address{}) {
return errInvalidCheckpointBeneficiary
}
// Nonces must be 0x00..0 or 0xff..f, zeroes enforced on checkpoints
if !bytes.Equal(header.Nonce[:], nonceAuthVote) && !bytes.Equal(header.Nonce[:],
nonceDropVote) {
return errInvalidVote
}
if checkpoint && !bytes.Equal(header.Nonce[:], nonceDropVote) {
```

```
return errInvalidCheckpointVote
}
// Check that the extra-data contains both the vanity and signature
if len(header.Extra) < extraVanity {
return errMissingVanity
}
if len(header.Extra) < extraVanity+extraSeal {
return errMissingSignature
}
// Ensure that the extra-data contains a signer list on checkpoint, but none otherwise
signersBytes := len(header.Extra) - extraVanity - extraSeal
if !checkpoint && signersBytes != 0 {
return errExtraSigners
}
if checkpoint && signersBytes%common.AddressLength != 0 {
return errInvalidCheckpointSigners
}
// Ensure that the mix digest is zero as we don't have fork protection currently
if header.MixDigest != (common.Hash{}) {
return errInvalidMixDigest
}
// Ensure that the block doesn't contain any uncles which are meaningless in PoA
if header.UncleHash != uncleHash {
return errInvalidUncleHash
}
// Ensure that the block's difficulty is meaningful (may not be correct at this point)
if number > 0 {
if header.Difficulty == nil || (header.Difficulty.Cmp(diffInTurn) != 0 &&
header.Difficulty.Cmp(diffNoTurn) != 0) {
return errInvalidDifficulty
}
}
// All basic checks passed, verify cascading fields
return c.verifyCascadingFields(chain, header, parents)
}

// verifyCascadingFields verifies all the header fields that are not standalone,
// rather depend on a batch of previous headers. The caller may optionally pass
// in a batch of parents (ascending order) to avoid looking those up from the
// database. This is useful for concurrently verifying a batch of new headers.
func (c *Clique) verifyCascadingFields(chain consensus.ChainReader, header *types.Header,
parents []*types.Header) error {
```

```
// The genesis block is the always valid dead-end
number := header.Number.Uint64()
if number == 0 {
return nil
}
// Ensure that the block's timestamp isn't too close to it's parent
var parent *types.Header
if len(parents) > 0 {
parent = parents[len(parents)-1]
} else {
parent = chain.GetHeader(header.ParentHash, number-1)
}
if parent == nil || parent.Number.Uint64() != number-1 || parent.Hash() != header.ParentHash {
return consensus.ErrUnknownAncestor
}
if parent.Time.Uint64()+c.config.Period > header.Time.Uint64() {
return ErrInvalidTimestamp
}
// Retrieve the snapshot needed to verify this header and cache it
snap, err := c.snapshot(chain, number-1, header.ParentHash, parents)
if err != nil {
return err
}
// If the block is a checkpoint block, verify the signer list
if number%c.config.Epoch == 0 {
signers := make([]byte, len(snap.Signers)*common.AddressLength)
for i, signer := range snap.signers() {
copy(signers[i*common.AddressLength:], signer[:])
}
extraSuffix := len(header.Extra) - extraSeal
if !bytes.Equal(header.Extra[extraVanity:extraSuffix], signers) {
return errInvalidCheckpointSigners
}
}
// All basic checks passed, verify the seal and return
return c.verifySeal(chain, header, parents)
}

// snapshot retrieves the authorization snapshot at a given point in time.
func (c *Clique) snapshot(chain consensus.ChainReader, number uint64, hash common.Hash,
parents []*types.Header) (*Snapshot, error) {
// Search for a snapshot in memory or on disk for checkpoints
```

```
var (
headers []*types.Header
snap    *Snapshot
)
for snap == nil {
// If an in-memory snapshot was found, use that
if s, ok := c.recents.Get(hash); ok {
snap = s.(*Snapshot)
break
}
// If an on-disk checkpoint snapshot can be found, use that
if number%checkpointInterval == 0 {
if s, err := loadSnapshot(c.config, c.signatures, c.db, hash); err == nil {
log.Trace("Loaded voting snapshot form disk", "number", number, "hash", hash)
snap = s
break
}
}
// If we're at block zero, make a snapshot
if number == 0 {
genesis := chain.GetHeaderByNumber(0)
if err := c.VerifyHeader(chain, genesis, false); err != nil {
return nil, err
}
signers := make([]common.Address, (len(genesis.Extra)-extraVanity-
extraSeal)/common.AddressLength)
for i := 0; i < len(signers); i++ {
copy(signers[i][:], genesis.Extra[extraVanity+i*common.AddressLength:])
}
snap = newSnapshot(c.config, c.signatures, 0, genesis.Hash(), signers)
if err := snap.store(c.db); err != nil {
return nil, err
}
log.Trace("Stored genesis voting snapshot to disk")
break
}
// No snapshot for this header, gather the header and move backward
var header *types.Header
if len(parents) > 0 {
// If we have explicit parents, pick from there (enforced)
header = parents[len(parents)-1]
if header.Hash() != hash || header.Number.Uint64() != number {
```

```
return nil, consensus.ErrUnknownAncestor
}
parents = parents[:len(parents)-1]
} else {
// No explicit parents (or no more left), reach out to the database
header = chain.GetHeader(hash, number)
if header == nil {
return nil, consensus.ErrUnknownAncestor
}
}
headers = append(headers, header)
number, hash = number-1, header.ParentHash
}
// Previous snapshot found, apply any pending headers on top of it
for i := 0; i < len(headers)/2; i++ {
headers[i], headers[len(headers)-1-i] = headers[len(headers)-1-i], headers[i]
}
snap, err := snap.apply(headers)
if err != nil {
return nil, err
}
c.recents.Add(snap.Hash, snap)

// If we've generated a new checkpoint snapshot, save to disk
if snap.Number%checkpointInterval == 0 && len(headers) > 0 {
if err = snap.store(c.db); err != nil {
return nil, err
}
log.Trace("Stored voting snapshot to disk", "number", snap.Number, "hash", snap.Hash)
}
return snap, err
}

// VerifyUncles implements consensus.Engine, always returning an error for any
// uncles as this consensus mechanism doesn't permit uncles.
func (c *Clique) VerifyUncles(chain consensus.ChainReader, block *types.Block) error {
if len(block.Uncles()) > 0 {
return errors.New("uncles not allowed")
}
return nil
}
```

```
// VerifySeal implements consensus.Engine, checking whether the signature contained
// in the header satisfies the consensus protocol requirements.
func (c *Clique) VerifySeal(chain consensus.ChainReader, header *types.Header) error {
return c.verifySeal(chain, header, nil)
}

// verifySeal checks whether the signature contained in the header satisfies the
// consensus protocol requirements. The method accepts an optional list of parent
// headers that aren't yet part of the local blockchain to generate the snapshots
// from.
func (c *Clique) verifySeal(chain consensus.ChainReader, header *types.Header, parents
[]*types.Header) error {
// Verifying the genesis block is not supported
number := header.Number.Uint64()
if number == 0 {
return errUnknownBlock
}
// Retrieve the snapshot needed to verify this header and cache it
snap, err := c.snapshot(chain, number-1, header.ParentHash, parents)
if err != nil {
return err
}

// Resolve the authorization key and check against signers
signer, err := ecrecover(header, c.signatures)
if err != nil {
return err
}
if _, ok := snap.Signers[signer]; !ok {
return errUnauthorized
}
for seen, recent := range snap.Recents {
if recent == signer {
// Signer is among recents, only fail if the current block doesn't shift it out
if limit := uint64(len(snap.Signers)/2 + 1); seen > number-limit {
return errUnauthorized
}
}
}
// Ensure that the difficulty corresponds to the turn-ness of the signer
inturn := snap.inturn(header.Number.Uint64(), signer)
if inturn && header.Difficulty.Cmp(diffInTurn) != 0 {
```

```
return errInvalidDifficulty
}
if !inturn && header.Difficulty.Cmp(diffNoTurn) != 0 {
return errInvalidDifficulty
}
return nil
}

// Prepare implements consensus.Engine, preparing all the consensus fields of the
// header for running the transactions on top.
func (c *Clique) Prepare(chain consensus.ChainReader, header *types.Header) error {
// If the block isn't a checkpoint, cast a random vote (good enough for now)
header.Coinbase = common.Address{}
header.Nonce = types.BlockNonce{}

number := header.Number.Uint64()

// Assemble the voting snapshot to check which votes make sense
snap, err := c.snapshot(chain, number-1, header.ParentHash, nil)
if err != nil {
return err
}
if number%c.config.Epoch != 0 {
c.lock.RLock()

// Gather all the proposals that make sense voting on
addresses := make([]common.Address, 0, len(c.proposals))
for address, authorize := range c.proposals {
if snap.validVote(address, authorize) {
addresses = append(addresses, address)
}
}
// If there's pending proposals, cast a vote on them
if len(addresses) > 0 {
header.Coinbase = addresses[rand.Intn(len(addresses))]
if c.proposals[header.Coinbase] {
copy(header.Nonce[:], nonceAuthVote)
} else {
copy(header.Nonce[:], nonceDropVote)
}
}
c.lock.RUnlock()
```

```
}
// Set the correct difficulty
header.Difficulty = diffNoTurn
if snap.inturn(header.Number.Uint64(), c.signer) {
header.Difficulty = diffInTurn
}
// Ensure the extra data has all it's components
if len(header.Extra) < extraVanity {
header.Extra = append(header.Extra, bytes.Repeat([]byte{0x00}, extraVanity-len(header.Extra))...)
}
header.Extra = header.Extra[:extraVanity]

if number%c.config.Epoch == 0 {
for _, signer := range snap.signers() {
header.Extra = append(header.Extra, signer[:]...)
}
}
header.Extra = append(header.Extra, make([]byte, extraSeal)...)

// Mix digest is reserved for now, set to empty
header.MixDigest = common.Hash{}

// Ensure the timestamp has the correct delay
parent := chain.GetHeader(header.ParentHash, number-1)
if parent == nil {
return consensus.ErrUnknownAncestor
}
header.Time = new(big.Int).Add(parent.Time, new(big.Int).SetUint64(c.config.Period))
if header.Time.Int64() < time.Now().Unix() {
header.Time = big.NewInt(time.Now().Unix())
}
return nil
}

// Finalize implements consensus.Engine, ensuring no uncles are set, nor block
// rewards given, and returns the final block.
func (c *Clique) Finalize(chain consensus.ChainReader, header *types.Header, state
*state.StateDB, txs []*types.Transaction, uncles []*types.Header, receipts []*types.Receipt)
(*types.Block, error) {
// No block rewards in PoA, so the state remains as is and uncles are dropped
header.Root = state.IntermediateRoot(chain.Config().IsEIP158(header.Number))
header.UncleHash = types.CalcUncleHash(nil)
```

```
// Assemble and return the final block for sealing
return types.NewBlock(header, txs, nil, receipts), nil
}

// Authorize injects a private key into the consensus engine to mint new blocks
// with.
func (c *Clique) Authorize(signer common.Address, signFn SignerFn) {
c.lock.Lock()
defer c.lock.Unlock()

c.signer = signer
c.signFn = signFn
}

// Seal implements consensus.Engine, attempting to create a sealed block using
// the local signing credentials.
func (c *Clique) Seal(chain consensus.ChainReader, block *types.Block, stop <-chan struct{})
(*types.Block, error) {
header := block.Header()

// Sealing the genesis block is not supported
number := header.Number.Uint64()
if number == 0 {
return nil, errUnknownBlock
}
// Don't hold the signer fields for the entire sealing procedure
c.lock.RLock()
signer, signFn := c.signer, c.signFn
c.lock.RUnlock()

// Bail out if we're unauthorized to sign a block
snap, err := c.snapshot(chain, number-1, header.ParentHash, nil)
if err != nil {
return nil, err
}
if _, authorized := snap.Signers[signer]; !authorized {
return nil, errUnauthorized
}
// If we're amongst the recent signers, wait for the next block
for seen, recent := range snap.Recents {
if recent == signer {
```

```
// Signer is among recents, only wait if the current block doesn't shift it out
if limit := uint64(len(snap.Signers)/2 + 1); number < limit || seen > number-limit {
log.Info("Signed recently, must wait for others")
<-stop
return nil, nil
}
}
}
// Sweet, the protocol permits us to sign the block, wait for our time
delay := time.Unix(header.Time.Int64(), 0).Sub(time.Now())
if header.Difficulty.Cmp(diffNoTurn) == 0 {
// It's not our turn explicitly to sign, delay it a bit
wiggle := time.Duration(len(snap.Signers)/2+1) * wiggleTime
delay += time.Duration(rand.Int63n(int64(wiggle)))

log.Trace("Out-of-turn signing requested", "wiggle", common.PrettyDuration(wiggle))
}
log.Trace("Waiting for slot to sign and propagate", "delay", common.PrettyDuration(delay))

select {
case <-stop:
return nil, nil
case <-time.After(delay):
}
// Sign all the things!
sighash, err := signFn(accounts.Account{Address: signer}, sigHash(header).Bytes())
if err != nil {
return nil, err
}
copy(header.Extra[len(header.Extra)-extraSeal:], sighash)

return block.WithSeal(header), nil
}

// APIs implements consensus.Engine, returning the user facing RPC API to allow
// controlling the signer voting.
func (c *Clique) APIs(chain consensus.ChainReader) []rpc.API {
return []rpc.API{{
Namespace: "clique",
Version:   "1.0",
Service:   &API{chain: chain, clique: c},
Public:    false,
```

}}
}

24:F:\git\coin\ethereum\go-ethereum\consensus\clique\snapshot.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package clique

import (
"bytes"
"encoding/json"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/params"
lru "github.com/hashicorp/golang-lru"
)

// Vote represents a single vote that an authorized signer made to modify the
// list of authorizations.
type Vote struct {
Signer   common.Address `json:"signer"`   // Authorized signer that cast this vote
Block   uint64   `json:"block"`   // Block number the vote was cast in (expire old votes)
Address   common.Address `json:"address"`   // Account being voted on to change its authorization
Authorize bool   `json:"authorize"` // Whether to authorize or deauthorize the voted account
}

// Tally is a simple vote tally to keep the current score of votes. Votes that
// go against the proposal aren't counted since it's equivalent to not voting.
type Tally struct {
Authorize bool `json:"authorize"` // Whether the vote is about authorizing or kicking someone
Votes   int `json:"votes"`   // Number of votes until now wanting to pass the proposal
}

// Snapshot is the state of the authorization voting at a given point in time.
type Snapshot struct {
config   *params.CliqueConfig // Consensus engine parameters to fine tune behavior
sigcache *lru.ARCCache   // Cache of recent block signatures to speed up ecrecover

Number uint64   `json:"number"` // Block number where the snapshot was created

```
Hash    common.Hash                `json:"hash"`    // Block hash where the snapshot was created
Signers map[common.Address]struct{} `json:"signers"` // Set of authorized signers at this moment
Recents map[uint64]common.Address   `json:"recents"` // Set of recent signers for spam protections
Votes   []*Vote                     `json:"votes"`   // List of votes cast in chronological order
Tally   map[common.Address]Tally    `json:"tally"`   // Current vote tally to avoid recalculating
}

// newSnapshot creates a new snapshot with the specified startup parameters. This
// method does not initialize the set of recent signers, so only ever use if for
// the genesis block.
func newSnapshot(config *params.CliqueConfig, sigcache *lru.ARCCache, number uint64, hash common.Hash, signers []common.Address) *Snapshot {
snap := &Snapshot{
config:   config,
sigcache: sigcache,
Number:   number,
Hash:     hash,
Signers:  make(map[common.Address]struct{}),
Recents:  make(map[uint64]common.Address),
Tally:    make(map[common.Address]Tally),
}
for _, signer := range signers {
snap.Signers[signer] = struct{}{}
}
return snap
}

// loadSnapshot loads an existing snapshot from the database.
func loadSnapshot(config *params.CliqueConfig, sigcache *lru.ARCCache, db ethdb.Database, hash common.Hash) (*Snapshot, error) {
blob, err := db.Get(append([]byte("clique-"), hash[:]...))
if err != nil {
return nil, err
}
snap := new(Snapshot)
if err := json.Unmarshal(blob, snap); err != nil {
return nil, err
}
snap.config = config
snap.sigcache = sigcache
```

```
return snap, nil
}

// store inserts the snapshot into the database.
func (s *Snapshot) store(db ethdb.Database) error {
blob, err := json.Marshal(s)
if err != nil {
return err
}
return db.Put(append([]byte("clique-"), s.Hash[:]...), blob)
}

// copy creates a deep copy of the snapshot, though not the individual votes.
func (s *Snapshot) copy() *Snapshot {
cpy := &Snapshot{
config:   s.config,
sigcache: s.sigcache,
Number:   s.Number,
Hash:     s.Hash,
Signers:  make(map[common.Address]struct{}),
Recents:  make(map[uint64]common.Address),
Votes:    make([]*Vote, len(s.Votes)),
Tally:    make(map[common.Address]Tally),
}
for signer := range s.Signers {
cpy.Signers[signer] = struct{}{}
}
for block, signer := range s.Recents {
cpy.Recents[block] = signer
}
for address, tally := range s.Tally {
cpy.Tally[address] = tally
}
copy(cpy.Votes, s.Votes)

return cpy
}

// validVote returns whether it makes sense to cast the specified vote in the
// given snapshot context (e.g. don't try to add an already authorized signer).
func (s *Snapshot) validVote(address common.Address, authorize bool) bool {
_, signer := s.Signers[address]
```

```
return (signer && !authorize) || (!signer && authorize)
}

// cast adds a new vote into the tally.
func (s *Snapshot) cast(address common.Address, authorize bool) bool {
// Ensure the vote is meaningful
if !s.validVote(address, authorize) {
return false
}
// Cast the vote into an existing or new tally
if old, ok := s.Tally[address]; ok {
old.Votes++
s.Tally[address] = old
} else {
s.Tally[address] = Tally{Authorize: authorize, Votes: 1}
}
return true
}

// uncast removes a previously cast vote from the tally.
func (s *Snapshot) uncast(address common.Address, authorize bool) bool {
// If there's no tally, it's a dangling vote, just drop
tally, ok := s.Tally[address]
if !ok {
return false
}
// Ensure we only revert counted votes
if tally.Authorize != authorize {
return false
}
// Otherwise revert the vote
if tally.Votes > 1 {
tally.Votes--
s.Tally[address] = tally
} else {
delete(s.Tally, address)
}
return true
}

// apply creates a new authorization snapshot by applying the given headers to
// the original one.
```

```
func (s *Snapshot) apply(headers []*types.Header) (*Snapshot, error) {
// Allow passing in no headers for cleaner code
if len(headers) == 0 {
return s, nil
}
// Sanity check that the headers can be applied
for i := 0; i < len(headers)-1; i++ {
if headers[i+1].Number.Uint64() != headers[i].Number.Uint64()+1 {
return nil, errInvalidVotingChain
}
}
if headers[0].Number.Uint64() != s.Number+1 {
return nil, errInvalidVotingChain
}
// Iterate through the headers and create a new snapshot
snap := s.copy()

for _, header := range headers {
// Remove any votes on checkpoint blocks
number := header.Number.Uint64()
if number%s.config.Epoch == 0 {
snap.Votes = nil
snap.Tally = make(map[common.Address]Tally)
}
// Delete the oldest signer from the recent list to allow it signing again
if limit := uint64(len(snap.Signers)/2 + 1); number >= limit {
delete(snap.Recents, number-limit)
}
// Resolve the authorization key and check against signers
signer, err := ecrecover(header, s.sigcache)
if err != nil {
return nil, err
}
if _, ok := snap.Signers[signer]; !ok {
return nil, errUnauthorized
}
for _, recent := range snap.Recents {
if recent == signer {
return nil, errUnauthorized
}
}
snap.Recents[number] = signer
```

```
// Header authorized, discard any previous votes from the signer
for i, vote := range snap.Votes {
if vote.Signer == signer && vote.Address == header.Coinbase {
// Uncast the vote from the cached tally
snap.uncast(vote.Address, vote.Authorize)

// Uncast the vote from the chronological list
snap.Votes = append(snap.Votes[:i], snap.Votes[i+1:]...)
break // only one vote allowed
}
}
// Tally up the new vote from the signer
var authorize bool
switch {
case bytes.Compare(header.Nonce[:], nonceAuthVote) == 0:
authorize = true
case bytes.Compare(header.Nonce[:], nonceDropVote) == 0:
authorize = false
default:
return nil, errInvalidVote
}
if snap.cast(header.Coinbase, authorize) {
snap.Votes = append(snap.Votes, &Vote{
Signer:    signer,
Block:     number,
Address:   header.Coinbase,
Authorize: authorize,
})
}
// If the vote passed, update the list of signers
if tally := snap.Tally[header.Coinbase]; tally.Votes > len(snap.Signers)/2 {
if tally.Authorize {
snap.Signers[header.Coinbase] = struct{}{}
} else {
delete(snap.Signers, header.Coinbase)

// Signer list shrunk, delete any leftover recent caches
if limit := uint64(len(snap.Signers)/2 + 1); number >= limit {
delete(snap.Recents, number-limit)
}
// Discard any previous votes the deauthorized signer cast
```

```
for i := 0; i < len(snap.Votes); i++ {
if snap.Votes[i].Signer == header.Coinbase {
// Uncast the vote from the cached tally
snap.uncast(snap.Votes[i].Address, snap.Votes[i].Authorize)

// Uncast the vote from the chronological list
snap.Votes = append(snap.Votes[:i], snap.Votes[i+1:]...)

i--
}
}
}
// Discard any previous votes around the just changed account
for i := 0; i < len(snap.Votes); i++ {
if snap.Votes[i].Address == header.Coinbase {
snap.Votes = append(snap.Votes[:i], snap.Votes[i+1:]...)
i--
}
}
delete(snap.Tally, header.Coinbase)
}
}
snap.Number += uint64(len(headers))
snap.Hash = headers[len(headers)-1].Hash()

return snap, nil
}

// signers retrieves the list of authorized signers in ascending order.
func (s *Snapshot) signers() []common.Address {
signers := make([]common.Address, 0, len(s.Signers))
for signer := range s.Signers {
signers = append(signers, signer)
}
for i := 0; i < len(signers); i++ {
for j := i + 1; j < len(signers); j++ {
if bytes.Compare(signers[i][:], signers[j][:]) > 0 {
signers[i], signers[j] = signers[j], signers[i]
}
}
}
return signers
```

```
}

// inturn returns if a signer at a given block height is in-turn or not.
func (s *Snapshot) inturn(number uint64, signer common.Address) bool {
signers, offset := s.signers(), 0
for offset < len(signers) && signers[offset] != signer {
offset++
}
return (number % uint64(len(signers))) == uint64(offset)
}

25:F:\git\coin\ethereum\go-ethereum\consensus\clique\snapshot_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package clique

import (
"bytes"
"crypto/ecdsa"
"math/big"
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/params"
)

type testerVote struct {
signer string
voted  string
auth   bool
}

// testerAccountPool is a pool to maintain currently active tester accounts,
// mapped from textual names used in the tests below to actual Ethereum private
// keys capable of signing transactions.
type testerAccountPool struct {
accounts map[string]*ecdsa.PrivateKey
}
```

```
func newTesterAccountPool() *testerAccountPool {
return &testerAccountPool{
accounts: make(map[string]*ecdsa.PrivateKey),
}
}

func (ap *testerAccountPool) sign(header *types.Header, signer string) {
// Ensure we have a persistent key for the signer
if ap.accounts[signer] == nil {
ap.accounts[signer], _ = crypto.GenerateKey()
}
// Sign the header and embed the signature in extra data
sig, _ := crypto.Sign(sigHash(header).Bytes(), ap.accounts[signer])
copy(header.Extra[len(header.Extra)-65:], sig)
}

func (ap *testerAccountPool) address(account string) common.Address {
// Ensure we have a persistent key for the account
if ap.accounts[account] == nil {
ap.accounts[account], _ = crypto.GenerateKey()
}
// Resolve and return the Ethereum address
return crypto.PubkeyToAddress(ap.accounts[account].PublicKey)
}

// testerChainReader implements consensus.ChainReader to access the genesis
// block. All other methods and requests will panic.
type testerChainReader struct {
db ethdb.Database
}

func (r *testerChainReader) Config() *params.ChainConfig                 { panic("not supported") }
func (r *testerChainReader) CurrentHeader() *types.Header                { panic("not supported") }
func (r *testerChainReader) GetHeader(common.Hash, uint64) *types.Header { panic("not
supported") }
func (r *testerChainReader) GetBlock(common.Hash, uint64) *types.Block   { panic("not
supported") }
func (r *testerChainReader) GetHeaderByHash(common.Hash) *types.Header   { panic("not
supported") }
func (r *testerChainReader) GetHeaderByNumber(number uint64) *types.Header {
if number == 0 {
```

```
return core.GetHeader(r.db, core.GetCanonicalHash(r.db, 0), 0)
}
panic("not supported")
}

// Tests that voting is evaluated correctly for various simple and complex scenarios.
func TestVoting(t *testing.T) {
// Define the various voting scenarios to test
tests := []struct {
epoch   uint64
signers []string
votes   []testerVote
results []string
}{
{
// Single signer, no votes cast
signers: []string{"A"},
votes:   []testerVote{{signer: "A"}},
results: []string{"A"},
}, {
// Single signer, voting to add two others (only accept first, second needs 2 votes)
signers: []string{"A"},
votes: []testerVote{
{signer: "A", voted: "B", auth: true},
{signer: "B"},
{signer: "A", voted: "C", auth: true},
},
results: []string{"A", "B"},
}, {
// Two signers, voting to add three others (only accept first two, third needs 3 votes already)
signers: []string{"A", "B"},
votes: []testerVote{
{signer: "A", voted: "C", auth: true},
{signer: "B", voted: "C", auth: true},
{signer: "A", voted: "D", auth: true},
{signer: "B", voted: "D", auth: true},
{signer: "C"},
{signer: "A", voted: "E", auth: true},
{signer: "B", voted: "E", auth: true},
},
results: []string{"A", "B", "C", "D"},
}, {
```

```
// Single signer, dropping itself (weird, but one less cornercase by explicitly allowing this)
signers: []string{"A"},
votes: []testerVote{
{signer: "A", voted: "A", auth: false},
},
results: []string{},
}, {
// Two signers, actually needing mutual consent to drop either of them (not fulfilled)
signers: []string{"A", "B"},
votes: []testerVote{
{signer: "A", voted: "B", auth: false},
},
results: []string{"A", "B"},
}, {
// Two signers, actually needing mutual consent to drop either of them (fulfilled)
signers: []string{"A", "B"},
votes: []testerVote{
{signer: "A", voted: "B", auth: false},
{signer: "B", voted: "B", auth: false},
},
results: []string{"A"},
}, {
// Three signers, two of them deciding to drop the third
signers: []string{"A", "B", "C"},
votes: []testerVote{
{signer: "A", voted: "C", auth: false},
{signer: "B", voted: "C", auth: false},
},
results: []string{"A", "B"},
}, {
// Four signers, consensus of two not being enough to drop anyone
signers: []string{"A", "B", "C", "D"},
votes: []testerVote{
{signer: "A", voted: "C", auth: false},
{signer: "B", voted: "C", auth: false},
},
results: []string{"A", "B", "C", "D"},
}, {
// Four signers, consensus of three already being enough to drop someone
signers: []string{"A", "B", "C", "D"},
votes: []testerVote{
{signer: "A", voted: "D", auth: false},
```

```
{signer: "B", voted: "D", auth: false},
{signer: "C", voted: "D", auth: false},
},
results: []string{"A", "B", "C"},
}, {
// Authorizations are counted once per signer per target
signers: []string{"A", "B"},
votes: []testerVote{
{signer: "A", voted: "C", auth: true},
{signer: "B"},
{signer: "A", voted: "C", auth: true},
{signer: "B"},
{signer: "A", voted: "C", auth: true},
},
results: []string{"A", "B"},
}, {
// Authorizing multiple accounts concurrently is permitted
signers: []string{"A", "B"},
votes: []testerVote{
{signer: "A", voted: "C", auth: true},
{signer: "B"},
{signer: "A", voted: "D", auth: true},
{signer: "B"},
{signer: "A"},
{signer: "B", voted: "D", auth: true},
{signer: "A"},
{signer: "B", voted: "C", auth: true},
},
results: []string{"A", "B", "C", "D"},
}, {
// Deauthorizations are counted once per signer per target
signers: []string{"A", "B"},
votes: []testerVote{
{signer: "A", voted: "B", auth: false},
{signer: "B"},
{signer: "A", voted: "B", auth: false},
{signer: "B"},
{signer: "A", voted: "B", auth: false},
},
results: []string{"A", "B"},
}, {
// Deauthorizing multiple accounts concurrently is permitted
```

```
signers: []string{"A", "B", "C", "D"},
votes: []testerVote{
{signer: "A", voted: "C", auth: false},
{signer: "B"},
{signer: "C"},
{signer: "A", voted: "D", auth: false},
{signer: "B"},
{signer: "C"},
{signer: "A"},
{signer: "B", voted: "D", auth: false},
{signer: "C", voted: "D", auth: false},
{signer: "A"},
{signer: "B", voted: "C", auth: false},
},
results: []string{"A", "B"},
}, {
// Votes from deauthorized signers are discarded immediately (deauth votes)
signers: []string{"A", "B", "C"},
votes: []testerVote{
{signer: "C", voted: "B", auth: false},
{signer: "A", voted: "C", auth: false},
{signer: "B", voted: "C", auth: false},
{signer: "A", voted: "B", auth: false},
},
results: []string{"A", "B"},
}, {
// Votes from deauthorized signers are discarded immediately (auth votes)
signers: []string{"A", "B", "C"},
votes: []testerVote{
{signer: "C", voted: "B", auth: false},
{signer: "A", voted: "C", auth: false},
{signer: "B", voted: "C", auth: false},
{signer: "A", voted: "B", auth: false},
},
results: []string{"A", "B"},
}, {
// Cascading changes are not allowed, only the account being voted on may change
signers: []string{"A", "B", "C", "D"},
votes: []testerVote{
{signer: "A", voted: "C", auth: false},
{signer: "B"},
{signer: "C"},
```

```
{signer: "A", voted: "D", auth: false},
{signer: "B", voted: "C", auth: false},
{signer: "C"},
{signer: "A"},
{signer: "B", voted: "D", auth: false},
{signer: "C", voted: "D", auth: false},
},
results: []string{"A", "B", "C"},
}, {
// Changes reaching consensus out of bounds (via a deauth) execute on touch
signers: []string{"A", "B", "C", "D"},
votes: []testerVote{
{signer: "A", voted: "C", auth: false},
{signer: "B"},
{signer: "C"},
{signer: "A", voted: "D", auth: false},
{signer: "B", voted: "C", auth: false},
{signer: "C"},
{signer: "A"},
{signer: "B", voted: "D", auth: false},
{signer: "C", voted: "D", auth: false},
{signer: "A"},
{signer: "C", voted: "C", auth: true},
},
results: []string{"A", "B"},
}, {
// Changes reaching consensus out of bounds (via a deauth) may go out of consensus on first
touch
signers: []string{"A", "B", "C", "D"},
votes: []testerVote{
{signer: "A", voted: "C", auth: false},
{signer: "B"},
{signer: "C"},
{signer: "A", voted: "D", auth: false},
{signer: "B", voted: "C", auth: false},
{signer: "C"},
{signer: "A"},
{signer: "B", voted: "D", auth: false},
{signer: "C", voted: "D", auth: false},
{signer: "A"},
{signer: "B", voted: "C", auth: true},
},
```

```
results: []string{"A", "B", "C"},
}, {
// Ensure that pending votes don't survive authorization status changes. This
// corner case can only appear if a signer is quickly added, removed and then
// readded (or the inverse), while one of the original voters dropped. If a
// past vote is left cached in the system somewhere, this will interfere with
// the final signer outcome.
signers: []string{"A", "B", "C", "D", "E"},
votes: []testerVote{
{signer: "A", voted: "F", auth: true}, // Authorize F, 3 votes needed
{signer: "B", voted: "F", auth: true},
{signer: "C", voted: "F", auth: true},
{signer: "D", voted: "F", auth: false}, // Deauthorize F, 4 votes needed (leave A's previous vote "unchanged")
{signer: "E", voted: "F", auth: false},
{signer: "B", voted: "F", auth: false},
{signer: "C", voted: "F", auth: false},
{signer: "D", voted: "F", auth: true}, // Almost authorize F, 2/3 votes needed
{signer: "E", voted: "F", auth: true},
{signer: "B", voted: "A", auth: false}, // Deauthorize A, 3 votes needed
{signer: "C", voted: "A", auth: false},
{signer: "D", voted: "A", auth: false},
{signer: "B", voted: "F", auth: true}, // Finish authorizing F, 3/3 votes needed
},
results: []string{"B", "C", "D", "E", "F"},
}, {
// Epoch transitions reset all votes to allow chain checkpointing
epoch:   3,
signers: []string{"A", "B"},
votes: []testerVote{
{signer: "A", voted: "C", auth: true},
{signer: "B"},
{signer: "A"}, // Checkpoint block, (don't vote here, it's validated outside of snapshots)
{signer: "B", voted: "C", auth: true},
},
results: []string{"A", "B"},
},
}
// Run through the scenarios and test them
for i, tt := range tests {
// Create the account pool and generate the initial set of signers
accounts := newTesterAccountPool()
```

```
signers := make([]common.Address, len(tt.signers))
for j, signer := range tt.signers {
signers[j] = accounts.address(signer)
}
for j := 0; j < len(signers); j++ {
for k := j + 1; k < len(signers); k++ {
if bytes.Compare(signers[j][:], signers[k][:]) > 0 {
signers[j], signers[k] = signers[k], signers[j]
}
}
}
// Create the genesis block with the initial set of signers
genesis := &core.Genesis{
ExtraData: make([]byte, extraVanity+common.AddressLength*len(signers)+extraSeal),
}
for j, signer := range signers {
copy(genesis.ExtraData[extraVanity+j*common.AddressLength:], signer[:])
}
// Create a pristine blockchain with the genesis injected
db, _ := ethdb.NewMemDatabase()
genesis.Commit(db)

// Assemble a chain of headers from the cast votes
headers := make([]*types.Header, len(tt.votes))
for j, vote := range tt.votes {
headers[j] = &types.Header{
Number:   big.NewInt(int64(j) + 1),
Time:     big.NewInt(int64(j) * int64(blockPeriod)),
Coinbase: accounts.address(vote.voted),
Extra:    make([]byte, extraVanity+extraSeal),
}
if j > 0 {
headers[j].ParentHash = headers[j-1].Hash()
}
if vote.auth {
copy(headers[j].Nonce[:], nonceAuthVote)
}
accounts.sign(headers[j], vote.signer)
}
// Pass all the headers through clique and ensure tallying succeeds
head := headers[len(headers)-1]
```

```
snap, err := New(&params.CliqueConfig{Epoch: tt.epoch}, db).snapshot(&testerChainReader{db:
db}, head.Number.Uint64(), head.Hash(), headers)
if err != nil {
t.Errorf("test %d: failed to create voting snapshot: %v", i, err)
continue
}
// Verify the final list of signers against the expected ones
signers = make([]common.Address, len(tt.results))
for j, signer := range tt.results {
signers[j] = accounts.address(signer)
}
for j := 0; j < len(signers); j++ {
for k := j + 1; k < len(signers); k++ {
if bytes.Compare(signers[j][:], signers[k][:]) > 0 {
signers[j], signers[k] = signers[k], signers[j]
}
}
}
result := snap.signers()
if len(result) != len(signers) {
t.Errorf("test %d: signers mismatch: have %x, want %x", i, result, signers)
continue
}
for j := 0; j < len(result); j++ {
if !bytes.Equal(result[j][:], signers[j][:]) {
t.Errorf("test %d, signer %d: signer mismatch: have %x, want %x", i, j, result[j], signers[j])
}
}
}
}
```

26:F:\git\coin\ethereum\go-ethereum\consensus\consensus.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package consensus implements different Ethereum consensus engines.
package consensus

import (
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
```

```
"github.com/ethereum/go-ethereum/params"
"github.com/ethereum/go-ethereum/rpc"
)

// ChainReader defines a small collection of methods needed to access the local
// blockchain during header and/or uncle verification.
type ChainReader interface {
// Config retrieves the blockchain's chain configuration.
Config() *params.ChainConfig

// CurrentHeader retrieves the current header from the local chain.
CurrentHeader() *types.Header

// GetHeader retrieves a block header from the database by hash and number.
GetHeader(hash common.Hash, number uint64) *types.Header

// GetHeaderByNumber retrieves a block header from the database by number.
GetHeaderByNumber(number uint64) *types.Header

// GetHeaderByHash retrieves a block header from the database by its hash.
GetHeaderByHash(hash common.Hash) *types.Header

// GetBlock retrieves a block from the database by hash and number.
GetBlock(hash common.Hash, number uint64) *types.Block
}

// Engine is an algorithm agnostic consensus engine.
type Engine interface {
// Author retrieves the Ethereum address of the account that minted the given
// block, which may be different from the header's coinbase if a consensus
// engine is based on signatures.
Author(header *types.Header) (common.Address, error)

// VerifyHeader checks whether a header conforms to the consensus rules of a
// given engine. Verifying the seal may be done optionally here, or explicitly
// via the VerifySeal method.
VerifyHeader(chain ChainReader, header *types.Header, seal bool) error

// VerifyHeaders is similar to VerifyHeader, but verifies a batch of headers
// concurrently. The method returns a quit channel to abort the operations and
// a results channel to retrieve the async verifications (the order is that of
// the input slice).
```

```
VerifyHeaders(chain ChainReader, headers []*types.Header, seals []bool) (chan<- struct{}, <-chan error)

// VerifyUncles verifies that the given block's uncles conform to the consensus
// rules of a given engine.
VerifyUncles(chain ChainReader, block *types.Block) error

// VerifySeal checks whether the crypto seal on a header is valid according to
// the consensus rules of the given engine.
VerifySeal(chain ChainReader, header *types.Header) error

// Prepare initializes the consensus fields of a block header according to the
// rules of a particular engine. The changes are executed inline.
Prepare(chain ChainReader, header *types.Header) error

// Finalize runs any post-transaction state modifications (e.g. block rewards)
// and assembles the final block.
// Note: The block header and state database might be updated to reflect any
// consensus rules that happen at finalization (e.g. block rewards).
Finalize(chain ChainReader, header *types.Header, state *state.StateDB, txs []*types.Transaction, uncles []*types.Header, receipts []*types.Receipt) (*types.Block, error)

// Seal generates a new block for the given input block with the local miner's
// seal place on top.
Seal(chain ChainReader, block *types.Block, stop <-chan struct{}) (*types.Block, error)

// APIs returns the RPC APIs this consensus engine provides.
APIs(chain ChainReader) []rpc.API
}

// PoW is a consensus engine based on proof-of-work.
type PoW interface {
Engine

// Hashrate returns the current mining hashrate of a PoW consensus engine.
Hashrate() float64
}

27:F:\git\coin\ethereum\go-ethereum\consensus\errors.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package consensus
```

```
import "errors"

var (
// ErrUnknownAncestor is returned when validating a block requires an ancestor
// that is unknown.
ErrUnknownAncestor = errors.New("unknown ancestor")

// ErrFutureBlock is returned when a block's timestamp is in the future according
// to the current node.
ErrFutureBlock = errors.New("block in the future")

// ErrInvalidNumber is returned if a block's number doesn't equal it's parent's
// plus one.
ErrInvalidNumber = errors.New("invalid block number")
)

28:F:\git\coin\ethereum\go-ethereum\consensus\ethash\algorithm.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ethash

import (
"encoding/binary"
"hash"
"reflect"
"runtime"
"sync"
"sync/atomic"
"time"
"unsafe"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/bitutil"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/crypto/sha3"
"github.com/ethereum/go-ethereum/log"
)

const (
datasetInitBytes   = 1 << 30 // Bytes in dataset at genesis
datasetGrowthBytes = 1 << 23 // Dataset growth per epoch
```

```
cacheInitBytes     = 1 << 24 // Bytes in cache at genesis
cacheGrowthBytes   = 1 << 17 // Cache growth per epoch
epochLength        = 30000   // Blocks per epoch
mixBytes           = 128     // Width of mix
hashBytes          = 64      // Hash length in bytes
hashWords          = 16      // Number of 32 bit ints in a hash
datasetParents     = 256     // Number of parents of each dataset element
cacheRounds        = 3       // Number of rounds in cache production
loopAccesses       = 64      // Number of accesses in hashimoto loop
)

// hasher is a repetitive hasher allowing the same hash data structures to be
// reused between hash runs instead of requiring new ones to be created.
type hasher func(dest []byte, data []byte)

// makeHasher creates a repetitive hasher, allowing the same hash data structures
// to be reused between hash runs instead of requiring new ones to be created.
// The returned function is not thread safe!
func makeHasher(h hash.Hash) hasher {
return func(dest []byte, data []byte) {
h.Write(data)
h.Sum(dest[:0])
h.Reset()
}
}

// seedHash is the seed to use for generating a verification cache and the mining
// dataset.
func seedHash(block uint64) []byte {
seed := make([]byte, 32)
if block < epochLength {
return seed
}
keccak256 := makeHasher(sha3.NewKeccak256())
for i := 0; i < int(block/epochLength); i++ {
keccak256(seed, seed)
}
return seed
}

// generateCache creates a verification cache of a given size for an input seed.
// The cache production process involves first sequentially filling up 32 MB of
```

```
// memory, then performing two passes of Sergio Demian Lerner's RandMemoHash
// algorithm from Strict Memory Hard Hashing Functions (2014). The output is a
// set of 524288 64-byte values.
// This method places the result into dest in machine byte order.
func generateCache(dest []uint32, epoch uint64, seed []byte) {
// Print some debug logs to allow analysis on low end devices
logger := log.New("epoch", epoch)

start := time.Now()
defer func() {
elapsed := time.Since(start)

logFn := logger.Debug
if elapsed > 3*time.Second {
logFn = logger.Info
}
logFn("Generated ethash verification cache", "elapsed", common.PrettyDuration(elapsed))
}()
// Convert our destination slice to a byte buffer
header := *(*reflect.SliceHeader)(unsafe.Pointer(&dest))
header.Len *= 4
header.Cap *= 4
cache := *(*[]byte)(unsafe.Pointer(&header))

// Calculate the number of thoretical rows (we'll store in one buffer nonetheless)
size := uint64(len(cache))
rows := int(size) / hashBytes

// Start a monitoring goroutine to report progress on low end devices
var progress uint32

done := make(chan struct{})
defer close(done)

go func() {
for {
select {
case <-done:
return
case <-time.After(3 * time.Second):
logger.Info("Generating ethash verification cache", "percentage",
atomic.LoadUint32(&progress)*100/uint32(rows)/4, "elapsed",
```

```
common.PrettyDuration(time.Since(start)))
}
}
}()
// Create a hasher to reuse between invocations
keccak512 := makeHasher(sha3.NewKeccak512())

// Sequentially produce the initial dataset
keccak512(cache, seed)
for offset := uint64(hashBytes); offset < size; offset += hashBytes {
keccak512(cache[offset:], cache[offset-hashBytes:offset])
atomic.AddUint32(&progress, 1)
}
// Use a low-round version of randmemohash
temp := make([]byte, hashBytes)

for i := 0; i < cacheRounds; i++ {
for j := 0; j < rows; j++ {
var (
srcOff = ((j - 1 + rows) % rows) * hashBytes
dstOff = j * hashBytes
xorOff = (binary.LittleEndian.Uint32(cache[dstOff:]) % uint32(rows)) * hashBytes
)
bitutil.XORBytes(temp, cache[srcOff:srcOff+hashBytes], cache[xorOff:xorOff+hashBytes])
keccak512(cache[dstOff:], temp)

atomic.AddUint32(&progress, 1)
}
}
// Swap the byte order on big endian systems and return
if !isLittleEndian() {
swap(cache)
}
}

// swap changes the byte order of the buffer assuming a uint32 representation.
func swap(buffer []byte) {
for i := 0; i < len(buffer); i += 4 {
binary.BigEndian.PutUint32(buffer[i:], binary.LittleEndian.Uint32(buffer[i:]))
}
}
```

```
// prepare converts an ethash cache or dataset from a byte stream into the internal
// int representation. All ethash methods work with ints to avoid constant byte to
// int conversions as well as to handle both little and big endian systems.
func prepare(dest []uint32, src []byte) {
for i := 0; i < len(dest); i++ {
dest[i] = binary.LittleEndian.Uint32(src[i*4:])
}
}

// fnv is an algorithm inspired by the FNV hash, which in some cases is used as
// a non-associative substitute for XOR. Note that we multiply the prime with
// the full 32-bit input, in contrast with the FNV-1 spec which multiplies the
// prime with one byte (octet) in turn.
func fnv(a, b uint32) uint32 {
return a*0x01000193 ^ b
}

// fnvHash mixes in data into mix using the ethash fnv method.
func fnvHash(mix []uint32, data []uint32) {
for i := 0; i < len(mix); i++ {
mix[i] = mix[i]*0x01000193 ^ data[i]
}
}

// generateDatasetItem combines data from 256 pseudorandomly selected cache nodes,
// and hashes that to compute a single dataset node.
func generateDatasetItem(cache []uint32, index uint32, keccak512 hasher) []byte {
// Calculate the number of thoretical rows (we use one buffer nonetheless)
rows := uint32(len(cache) / hashWords)

// Initialize the mix
mix := make([]byte, hashBytes)

binary.LittleEndian.PutUint32(mix, cache[(index%rows)*hashWords]^index)
for i := 1; i < hashWords; i++ {
binary.LittleEndian.PutUint32(mix[i*4:], cache[(index%rows)*hashWords+uint32(i)])
}
keccak512(mix, mix)

// Convert the mix to uint32s to avoid constant bit shifting
intMix := make([]uint32, hashWords)
for i := 0; i < len(intMix); i++ {
```

```
intMix[i] = binary.LittleEndian.Uint32(mix[i*4:])
}
// fnv it with a lot of random cache nodes based on index
for i := uint32(0); i < datasetParents; i++ {
parent := fnv(index^i, intMix[i%16]) % rows
fnvHash(intMix, cache[parent*hashWords:])
}
// Flatten the uint32 mix into a binary one and return
for i, val := range intMix {
binary.LittleEndian.PutUint32(mix[i*4:], val)
}
keccak512(mix, mix)
return mix
}

// generateDataset generates the entire ethash dataset for mining.
// This method places the result into dest in machine byte order.
func generateDataset(dest []uint32, epoch uint64, cache []uint32) {
// Print some debug logs to allow analysis on low end devices
logger := log.New("epoch", epoch)

start := time.Now()
defer func() {
elapsed := time.Since(start)

logFn := logger.Debug
if elapsed > 3*time.Second {
logFn = logger.Info
}
logFn("Generated ethash verification cache", "elapsed", common.PrettyDuration(elapsed))
}()

// Figure out whether the bytes need to be swapped for the machine
swapped := !isLittleEndian()

// Convert our destination slice to a byte buffer
header := *(*reflect.SliceHeader)(unsafe.Pointer(&dest))
header.Len *= 4
header.Cap *= 4
dataset := *(*[]byte)(unsafe.Pointer(&header))

// Generate the dataset on many goroutines since it takes a while
```

```
threads := runtime.NumCPU()
size := uint64(len(dataset))

var pend sync.WaitGroup
pend.Add(threads)

var progress uint32
for i := 0; i < threads; i++ {
go func(id int) {
defer pend.Done()

// Create a hasher to reuse between invocations
keccak512 := makeHasher(sha3.NewKeccak512())

// Calculate the data segment this thread should generate
batch := uint32((size + hashBytes*uint64(threads) - 1) / (hashBytes * uint64(threads)))
first := uint32(id) * batch
limit := first + batch
if limit > uint32(size/hashBytes) {
limit = uint32(size / hashBytes)
}
// Calculate the dataset segment
percent := uint32(size / hashBytes / 100)
for index := first; index < limit; index++ {
item := generateDatasetItem(cache, index, keccak512)
if swapped {
swap(item)
}
copy(dataset[index*hashBytes:], item)

if status := atomic.AddUint32(&progress, 1); status%percent == 0 {
logger.Info("Generating DAG in progress", "percentage", uint64(status*100)/(size/hashBytes),
"elapsed", common.PrettyDuration(time.Since(start)))
}
}
}(i)
}
// Wait for all the generators to finish and return
pend.Wait()
}

// hashimoto aggregates data from the full dataset in order to produce our final
```

```
// value for a particular header hash and nonce.
func hashimoto(hash []byte, nonce uint64, size uint64, lookup func(index uint32) []uint32) ([]byte,
[]byte) {
// Calculate the number of thoretical rows (we use one buffer nonetheless)
rows := uint32(size / mixBytes)

// Combine header+nonce into a 64 byte seed
seed := make([]byte, 40)
copy(seed, hash)
binary.LittleEndian.PutUint64(seed[32:], nonce)

seed = crypto.Keccak512(seed)
seedHead := binary.LittleEndian.Uint32(seed)

// Start the mix with replicated seed
mix := make([]uint32, mixBytes/4)
for i := 0; i < len(mix); i++ {
mix[i] = binary.LittleEndian.Uint32(seed[i%16*4:])
}
// Mix in random dataset nodes
temp := make([]uint32, len(mix))

for i := 0; i < loopAccesses; i++ {
parent := fnv(uint32(i)^seedHead, mix[i%len(mix)]) % rows
for j := uint32(0); j < mixBytes/hashBytes; j++ {
copy(temp[j*hashWords:], lookup(2*parent+j))
}
fnvHash(mix, temp)
}
// Compress mix
for i := 0; i < len(mix); i += 4 {
mix[i/4] = fnv(fnv(fnv(mix[i], mix[i+1]), mix[i+2]), mix[i+3])
}
mix = mix[:len(mix)/4]

digest := make([]byte, common.HashLength)
for i, val := range mix {
binary.LittleEndian.PutUint32(digest[i*4:], val)
}
return digest, crypto.Keccak256(append(seed, digest...))
}
```

```
// hashimotoLight aggregates data from the full dataset (using only a small
// in-memory cache) in order to produce our final value for a particular header
// hash and nonce.
func hashimotoLight(size uint64, cache []uint32, hash []byte, nonce uint64) ([]byte, []byte) {
keccak512 := makeHasher(sha3.NewKeccak512())

lookup := func(index uint32) []uint32 {
rawData := generateDatasetItem(cache, index, keccak512)

data := make([]uint32, len(rawData)/4)
for i := 0; i < len(data); i++ {
data[i] = binary.LittleEndian.Uint32(rawData[i*4:])
}
return data
}
return hashimoto(hash, nonce, size, lookup)
}

// hashimotoFull aggregates data from the full dataset (using the full in-memory
// dataset) in order to produce our final value for a particular header hash and
// nonce.
func hashimotoFull(dataset []uint32, hash []byte, nonce uint64) ([]byte, []byte) {
lookup := func(index uint32) []uint32 {
offset := index * hashWords
return dataset[offset : offset+hashWords]
}
return hashimoto(hash, nonce, uint64(len(dataset))*4, lookup)
}

// datasetSizes is a lookup table for the ethash dataset size for the first 2048
// epochs (i.e. 61440000 blocks).
var datasetSizes = []uint64{
1073739904, 1082130304, 1090514816, 1098906752, 1107293056,
1115684224, 1124070016, 1132461952, 1140849536, 1149232768,
1157627776, 1166013824, 1174404736, 1182786944, 1191180416,
1199568512, 1207958912, 1216345216, 1224732032, 1233124736,
1241513344, 1249902464, 1258290304, 1266673792, 1275067264,
1283453312, 1291844992, 1300234112, 1308619904, 1317010048,
1325397376, 1333787776, 1342176128, 1350561664, 1358954368,
1367339392, 1375731584, 1384118144, 1392507008, 1400897408,
1409284736, 1417673344, 1426062464, 1434451072, 1442839168,
1451229056, 1459615616, 1468006016, 1476394112, 1484782976,
```

1493171584, 1501559168, 1509948032, 1518337664, 1526726528,
1535114624, 1543503488, 1551892096, 1560278656, 1568669056,
1577056384, 1585446272, 1593831296, 1602219392, 1610610304,
1619000192, 1627386752, 1635773824, 1644164224, 1652555648,
1660943488, 1669332608, 1677721216, 1686109312, 1694497664,
1702886272, 1711274624, 1719661184, 1728047744, 1736434816,
1744829056, 1753218944, 1761606272, 1769995904, 1778382464,
1786772864, 1795157888, 1803550592, 1811937664, 1820327552,
1828711552, 1837102976, 1845488768, 1853879936, 1862269312,
1870656896, 1879048064, 1887431552, 1895825024, 1904212096,
1912601216, 1920988544, 1929379456, 1937765504, 1946156672,
1954543232, 1962932096, 1971321728, 1979707264, 1988093056,
1996487552, 2004874624, 2013262208, 2021653888, 2030039936,
2038430848, 2046819968, 2055208576, 2063596672, 2071981952,
2080373632, 2088762752, 2097149056, 2105539712, 2113928576,
2122315136, 2130700672, 2139092608, 2147483264, 2155872128,
2164257664, 2172642176, 2181035392, 2189426048, 2197814912,
2206203008, 2214587264, 2222979712, 2231367808, 2239758208,
2248145024, 2256527744, 2264922752, 2273312128, 2281701248,
2290086272, 2298476672, 2306867072, 2315251072, 2323639168,
2332032128, 2340420224, 2348808064, 2357196416, 2365580416,
2373966976, 2382363008, 2390748544, 2399139968, 2407530368,
2415918976, 2424307328, 2432695424, 2441084288, 2449472384,
2457861248, 2466247808, 2474637184, 2483026816, 2491414144,
2499803776, 2508191872, 2516582272, 2524970368, 2533359232,
2541743488, 2550134144, 2558525056, 2566913408, 2575301504,
2583686528, 2592073856, 2600467328, 2608856192, 2617240448,
2625631616, 2634022016, 2642407552, 2650796416, 2659188352,
2667574912, 2675965312, 2684352896, 2692738688, 2701130624,
2709518464, 2717907328, 2726293376, 2734685056, 2743073152,
2751462016, 2759851648, 2768232832, 2776625536, 2785017728,
2793401984, 2801794432, 2810182016, 2818571648, 2826959488,
2835349376, 2843734144, 2852121472, 2860514432, 2868900992,
2877286784, 2885676928, 2894069632, 2902451584, 2910843008,
2919234688, 2927622784, 2936011648, 2944400768, 2952789376,
2961177728, 2969565568, 2977951616, 2986338944, 2994731392,
3003120256, 3011508352, 3019895936, 3028287104, 3036675968,
3045063808, 3053452928, 3061837696, 3070228352, 3078615424,
3087003776, 3095394944, 3103782272, 3112173184, 3120562048,
3128944768, 3137339264, 3145725056, 3154109312, 3162505088,
3170893184, 3179280256, 3187669376, 3196056704, 3204445568,
3212836736, 3221224064, 3229612928, 3238002304, 3246391168,

3254778496, 3263165824, 3271556224, 3279944576, 3288332416,
3296719232, 3305110912, 3313500032, 3321887104, 3330273152,
3338658944, 3347053184, 3355440512, 3363827072, 3372220288,
3380608384, 3388997504, 3397384576, 3405774208, 3414163072,
3422551936, 3430937984, 3439328384, 3447714176, 3456104576,
3464493952, 3472883584, 3481268864, 3489655168, 3498048896,
3506434432, 3514826368, 3523213952, 3531603584, 3539987072,
3548380288, 3556763264, 3565157248, 3573545344, 3581934464,
3590324096, 3598712704, 3607098752, 3615488384, 3623877248,
3632265856, 3640646528, 3649043584, 3657430144, 3665821568,
3674207872, 3682597504, 3690984832, 3699367808, 3707764352,
3716152448, 3724541056, 3732925568, 3741318016, 3749706368,
3758091136, 3766481536, 3774872704, 3783260032, 3791650432,
3800036224, 3808427648, 3816815488, 3825204608, 3833592704,
3841981568, 3850370432, 3858755968, 3867147904, 3875536256,
3883920512, 3892313728, 3900702592, 3909087872, 3917478784,
3925868416, 3934256512, 3942645376, 3951032192, 3959422336,
3967809152, 3976200064, 3984588416, 3992974976, 4001363584,
4009751168, 4018141312, 4026530432, 4034911616, 4043308928,
4051695488, 4060084352, 4068472448, 4076862848, 4085249408,
4093640576, 4102028416, 4110413696, 4118805632, 4127194496,
4135583104, 4143971968, 4152360832, 4160746112, 4169135744,
4177525888, 4185912704, 4194303616, 4202691968, 4211076736,
4219463552, 4227855488, 4236246656, 4244633728, 4253022848,
4261412224, 4269799808, 4278184832, 4286578048, 4294962304,
4303349632, 4311743104, 4320130432, 4328521088, 4336909184,
4345295488, 4353687424, 4362073472, 4370458496, 4378852736,
4387238528, 4395630208, 4404019072, 4412407424, 4420790656,
4429182848, 4437571456, 4445962112, 4454344064, 4462738048,
4471119232, 4479516544, 4487904128, 4496289664, 4504682368,
4513068416, 4521459584, 4529846144, 4538232704, 4546619776,
4555010176, 4563402112, 4571790208, 4580174464, 4588567936,
4596957056, 4605344896, 4613734016, 4622119808, 4630511488,
4638898816, 4647287936, 4655675264, 4664065664, 4672451968,
4680842624, 4689231488, 4697620352, 4706007424, 4714397056,
4722786176, 4731173248, 4739562368, 4747951744, 4756340608,
4764727936, 4773114496, 4781504384, 4789894784, 4798283648,
4806667648, 4815059584, 4823449472, 4831835776, 4840226176,
4848612224, 4857003392, 4865391488, 4873780096, 4882169728,
4890557312, 4898946944, 4907333248, 4915722368, 4924110976,
4932499328, 4940889728, 4949276032, 4957666432, 4966054784,
4974438016, 4982831488, 4991221376, 4999607168, 5007998848,

5016386432, 5024763776, 5033164672, 5041544576, 5049941888,
5058329728, 5066717056, 5075107456, 5083494272, 5091883904,
5100273536, 5108662144, 5117048192, 5125436032, 5133827456,
5142215296, 5150605184, 5158993024, 5167382144, 5175769472,
5184157568, 5192543872, 5200936064, 5209324928, 5217711232,
5226102656, 5234490496, 5242877312, 5251263872, 5259654016,
5268040832, 5276434304, 5284819328, 5293209728, 5301598592,
5309986688, 5318374784, 5326764416, 5335151488, 5343542144,
5351929472, 5360319872, 5368706944, 5377096576, 5385484928,
5393871232, 5402263424, 5410650496, 5419040384, 5427426944,
5435816576, 5444205952, 5452594816, 5460981376, 5469367936,
5477760896, 5486148736, 5494536832, 5502925952, 5511315328,
5519703424, 5528089984, 5536481152, 5544869504, 5553256064,
5561645696, 5570032768, 5578423936, 5586811264, 5595193216,
5603585408, 5611972736, 5620366208, 5628750464, 5637143936,
5645528192, 5653921408, 5662310272, 5670694784, 5679082624,
5687474048, 5695864448, 5704251008, 5712641408, 5721030272,
5729416832, 5737806208, 5746194304, 5754583936, 5762969984,
5771358592, 5779748224, 5788137856, 5796527488, 5804911232,
5813300608, 5821692544, 5830082176, 5838468992, 5846855552,
5855247488, 5863636096, 5872024448, 5880411008, 5888799872,
5897186432, 5905576832, 5913966976, 5922352768, 5930744704,
5939132288, 5947522432, 5955911296, 5964299392, 5972688256,
5981074304, 5989465472, 5997851008, 6006241408, 6014627968,
6023015552, 6031408256, 6039796096, 6048185216, 6056574848,
6064963456, 6073351808, 6081736064, 6090128768, 6098517632,
6106906496, 6115289216, 6123680896, 6132070016, 6140459648,
6148849024, 6157237376, 6165624704, 6174009728, 6182403712,
6190792064, 6199176064, 6207569792, 6215952256, 6224345216,
6232732544, 6241124224, 6249510272, 6257899136, 6266287744,
6274676864, 6283065728, 6291454336, 6299843456, 6308232064,
6316620928, 6325006208, 6333395584, 6341784704, 6350174848,
6358562176, 6366951296, 6375337856, 6383729536, 6392119168,
6400504192, 6408895616, 6417283456, 6425673344, 6434059136,
6442444672, 6450837376, 6459223424, 6467613056, 6476004224,
6484393088, 6492781952, 6501170048, 6509555072, 6517947008,
6526336384, 6534725504, 6543112832, 6551500672, 6559888768,
6568278656, 6576662912, 6585055616, 6593443456, 6601834112,
6610219648, 6618610304, 6626999168, 6635385472, 6643777408,
6652164224, 6660552832, 6668941952, 6677330048, 6685719424,
6694107776, 6702493568, 6710882176, 6719274112, 6727662976,
6736052096, 6744437632, 6752825984, 6761213824, 6769604224,

6777993856, 6786383488, 6794770816, 6803158144, 6811549312,
6819937664, 6828326528, 6836706176, 6845101696, 6853491328,
6861880448, 6870269312, 6878655104, 6887046272, 6895433344,
6903822208, 6912212864, 6920596864, 6928988288, 6937377152,
6945764992, 6954149248, 6962544256, 6970928768, 6979317376,
6987709312, 6996093824, 7004487296, 7012875392, 7021258624,
7029652352, 7038038912, 7046427776, 7054818944, 7063207808,
7071595136, 7079980928, 7088372608, 7096759424, 7105149824,
7113536896, 7121928064, 7130315392, 7138699648, 7147092352,
7155479168, 7163865728, 7172249984, 7180648064, 7189036672,
7197424768, 7205810816, 7214196608, 7222589824, 7230975104,
7239367552, 7247755904, 7256145536, 7264533376, 7272921472,
7281308032, 7289694848, 7298088832, 7306471808, 7314864512,
7323253888, 7331643008, 7340029568, 7348419712, 7356808832,
7365196672, 7373585792, 7381973888, 7390362752, 7398750592,
7407138944, 7415528576, 7423915648, 7432302208, 7440690304,
7449080192, 7457472128, 7465860992, 7474249088, 7482635648,
7491023744, 7499412608, 7507803008, 7516192384, 7524579968,
7532967296, 7541358464, 7549745792, 7558134656, 7566524032,
7574912896, 7583300992, 7591690112, 7600075136, 7608466816,
7616854912, 7625244544, 7633629824, 7642020992, 7650410368,
7658794112, 7667187328, 7675574912, 7683961984, 7692349568,
7700739712, 7709130368, 7717519232, 7725905536, 7734295424,
7742683264, 7751069056, 7759457408, 7767849088, 7776238208,
7784626816, 7793014912, 7801405312, 7809792128, 7818179968,
7826571136, 7834957184, 7843347328, 7851732352, 7860124544,
7868512384, 7876902016, 7885287808, 7893679744, 7902067072,
7910455936, 7918844288, 7927230848, 7935622784, 7944009344,
7952400256, 7960786048, 7969176704, 7977565312, 7985953408,
7994339968, 8002730368, 8011119488, 8019508096, 8027896192,
8036285056, 8044674688, 8053062272, 8061448832, 8069838464,
8078227328, 8086616704, 8095006592, 8103393664, 8111783552,
8120171392, 8128560256, 8136949376, 8145336704, 8153726848,
8162114944, 8170503296, 8178891904, 8187280768, 8195669632,
8204058496, 8212444544, 8220834176, 8229222272, 8237612672,
8246000768, 8254389376, 8262775168, 8271167104, 8279553664,
8287944064, 8296333184, 8304715136, 8313108352, 8321497984,
8329885568, 8338274432, 8346663296, 8355052928, 8363441536,
8371828352, 8380217984, 8388606592, 8396996224, 8405384576,
8413772672, 8422161536, 8430549376, 8438939008, 8447326592,
8455715456, 8464104832, 8472492928, 8480882048, 8489270656,
8497659776, 8506045312, 8514434944, 8522823808, 8531208832,

8539602304, 8547990656, 8556378752, 8564768384, 8573154176,
8581542784, 8589933952, 8598322816, 8606705024, 8615099264,
8623487872, 8631876992, 8640264064, 8648653952, 8657040256,
8665430656, 8673820544, 8682209152, 8690592128, 8698977152,
8707374464, 8715763328, 8724151424, 8732540032, 8740928384,
8749315712, 8757704576, 8766089344, 8774480768, 8782871936,
8791260032, 8799645824, 8808034432, 8816426368, 8824812928,
8833199488, 8841591424, 8849976448, 8858366336, 8866757248,
8875147136, 8883532928, 8891923328, 8900306816, 8908700288,
8917088384, 8925478784, 8933867392, 8942250368, 8950644608,
8959032704, 8967420544, 8975809664, 8984197504, 8992584064,
9000976256, 9009362048, 9017752448, 9026141312, 9034530688,
9042917504, 9051307904, 9059694208, 9068084864, 9076471424,
9084861824, 9093250688, 9101638528, 9110027648, 9118416512,
9126803584, 9135188096, 9143581312, 9151969664, 9160356224,
9168747136, 9177134464, 9185525632, 9193910144, 9202302848,
9210690688, 9219079552, 9227465344, 9235854464, 9244244864,
9252633472, 9261021824, 9269411456, 9277799296, 9286188928,
9294574208, 9302965888, 9311351936, 9319740032, 9328131968,
9336516736, 9344907392, 9353296768, 9361685888, 9370074752,
9378463616, 9386849408, 9395239808, 9403629184, 9412016512,
9420405376, 9428795008, 9437181568, 9445570688, 9453960832,
9462346624, 9470738048, 9479121536, 9487515008, 9495903616,
9504289664, 9512678528, 9521067904, 9529456256, 9537843584,
9546233728, 9554621312, 9563011456, 9571398784, 9579788672,
9588178304, 9596567168, 9604954496, 9613343104, 9621732992,
9630121856, 9638508416, 9646898816, 9655283584, 9663675776,
9672061312, 9680449664, 9688840064, 9697230464, 9705617536,
9714003584, 9722393984, 9730772608, 9739172224, 9747561088,
9755945344, 9764338816, 9772726144, 9781116544, 9789503872,
9797892992, 9806282624, 9814670464, 9823056512, 9831439232,
9839833984, 9848224384, 9856613504, 9865000576, 9873391232,
9881772416, 9890162816, 9898556288, 9906940544, 9915333248,
9923721088, 9932108672, 9940496512, 9948888448, 9957276544,
9965666176, 9974048384, 9982441088, 9990830464, 9999219584,
10007602816, 10015996544, 10024385152, 10032774016, 10041163648,
10049548928, 10057940096, 10066329472, 10074717824, 10083105152,
10091495296, 10099878784, 10108272256, 10116660608, 10125049216,
10133437312, 10141825664, 10150213504, 10158601088, 10166991232,
10175378816, 10183766144, 10192157312, 10200545408, 10208935552,
10217322112, 10225712768, 10234099328, 10242489472, 10250876032,
10259264896, 10267656064, 10276042624, 10284429184, 10292820352,

10301209472, 10309598848, 10317987712, 10326375296, 10334763392,
10343153536, 10351541632, 10359930752, 10368318592, 10376707456,
10385096576, 10393484672, 10401867136, 10410262144, 10418647424,
10427039104, 10435425664, 10443810176, 10452203648, 10460589952,
10468982144, 10477369472, 10485759104, 10494147712, 10502533504,
10510923392, 10519313536, 10527702656, 10536091264, 10544478592,
10552867712, 10561255808, 10569642368, 10578032768, 10586423168,
10594805632, 10603200128, 10611588992, 10619976064, 10628361344,
10636754048, 10645143424, 10653531776, 10661920384, 10670307968,
10678696832, 10687086464, 10695475072, 10703863168, 10712246144,
10720639616, 10729026688, 10737414784, 10745806208, 10754190976,
10762581376, 10770971264, 10779356288, 10787747456, 10796135552,
10804525184, 10812915584, 10821301888, 10829692288, 10838078336,
10846469248, 10854858368, 10863247232, 10871631488, 10880023424,
10888412032, 10896799616, 10905188992, 10913574016, 10921964672,
10930352768, 10938742912, 10947132544, 10955518592, 10963909504,
10972298368, 10980687488, 10989074816, 10997462912, 11005851776,
11014241152, 11022627712, 11031017344, 11039403904, 11047793024,
11056184704, 11064570752, 11072960896, 11081343872, 11089737856,
11098128256, 11106514816, 11114904448, 11123293568, 11131680128,
11140065152, 11148458368, 11156845696, 11165236864, 11173624192,
11182013824, 11190402688, 11198790784, 11207179136, 11215568768,
11223957376, 11232345728, 11240734592, 11249122688, 11257511296,
11265899648, 11274285952, 11282675584, 11291065472, 11299452544,
11307842432, 11316231296, 11324616832, 11333009024, 11341395584,
11349782656, 11358172288, 11366560384, 11374950016, 11383339648,
11391721856, 11400117376, 11408504192, 11416893568, 11425283456,
11433671552, 11442061184, 11450444672, 11458837888, 11467226752,
11475611776, 11484003968, 11492392064, 11500780672, 11509169024,
11517550976, 11525944448, 11534335616, 11542724224, 11551111808,
11559500672, 11567890304, 11576277376, 11584667008, 11593056128,
11601443456, 11609830016, 11618221952, 11626607488, 11634995072,
11643387776, 11651775104, 11660161664, 11668552576, 11676940928,
11685330304, 11693718656, 11702106496, 11710496128, 11718882688,
11727273088, 11735660416, 11744050048, 11752437376, 11760824704,
11769216128, 11777604736, 11785991296, 11794381952, 11802770048,
11811157888, 11819548544, 11827932544, 11836324736, 11844713344,
11853100928, 11861486464, 11869879936, 11878268032, 11886656896,
11895044992, 11903433088, 11911822976, 11920210816, 11928600448,
11936987264, 11945375872, 11953761152, 11962151296, 11970543488,
11978928512, 11987320448, 11995708288, 12004095104, 12012486272,
12020875136, 12029255552, 12037652096, 12046039168, 12054429568,

12062813824, 12071206528, 12079594624, 12087983744, 12096371072,
12104759936, 12113147264, 12121534592, 12129924992, 12138314624,
12146703232, 12155091584, 12163481216, 12171864704, 12180255872,
12188643968, 12197034112, 12205424512, 12213811328, 12222199424,
12230590336, 12238977664, 12247365248, 12255755392, 12264143488,
12272531584, 12280920448, 12289309568, 12297694592, 12306086528,
12314475392, 12322865024, 12331253632, 12339640448, 12348029312,
12356418944, 12364805248, 12373196672, 12381580928, 12389969024,
12398357632, 12406750592, 12415138432, 12423527552, 12431916416,
12440304512, 12448692352, 12457081216, 12465467776, 12473859968,
12482245504, 12490636672, 12499025536, 12507411584, 12515801728,
12524190592, 12532577152, 12540966272, 12549354368, 12557743232,
12566129536, 12574523264, 12582911872, 12591299456, 12599688064,
12608074624, 12616463488, 12624845696, 12633239936, 12641631616,
12650019968, 12658407296, 12666795136, 12675183232, 12683574656,
12691960192, 12700350592, 12708740224, 12717128576, 12725515904,
12733906816, 12742295168, 12750680192, 12759071872, 12767460736,
12775848832, 12784236928, 12792626816, 12801014656, 12809404288,
12817789312, 12826181504, 12834568832, 12842954624, 12851345792,
12859732352, 12868122496, 12876512128, 12884901248, 12893289088,
12901672832, 12910067584, 12918455168, 12926842496, 12935232896,
12943620736, 12952009856, 12960396928, 12968786816, 12977176192,
12985563776, 12993951104, 13002341504, 13010730368, 13019115392,
13027506304, 13035895168, 13044272512, 13052673152, 13061062528,
13069446272, 13077838976, 13086227072, 13094613632, 13103000192,
13111393664, 13119782528, 13128157568, 13136559232, 13144945024,
13153329536, 13161724288, 13170111872, 13178502784, 13186884736,
13195279744, 13203667072, 13212057472, 13220445824, 13228832128,
13237221248, 13245610624, 13254000512, 13262388352, 13270777472,
13279166336, 13287553408, 13295943296, 13304331904, 13312719488,
13321108096, 13329494656, 13337885824, 13346274944, 13354663808,
13363051136, 13371439232, 13379825024, 13388210816, 13396605056,
13404995456, 13413380224, 13421771392, 13430159744, 13438546048,
13446937216, 13455326848, 13463708288, 13472103808, 13480492672,
13488875648, 13497269888, 13505657728, 13514045312, 13522435712,
13530824576, 13539210112, 13547599232, 13555989376, 13564379008,
13572766336, 13581154432, 13589544832, 13597932928, 13606320512,
13614710656, 13623097472, 13631477632, 13639874944, 13648264064,
13656652928, 13665041792, 13673430656, 13681818496, 13690207616,
13698595712, 13706982272, 13715373184, 13723762048, 13732150144,
13740536704, 13748926592, 13757316224, 13765700992, 13774090112,
13782477952, 13790869376, 13799259008, 13807647872, 13816036736,

13824425344, 13832814208, 13841202304, 13849591424, 13857978752,
13866368896, 13874754688, 13883145344, 13891533184, 13899919232,
13908311168, 13916692096, 13925085056, 13933473152, 13941866368,
13950253696, 13958643584, 13967032192, 13975417216, 13983807616,
13992197504, 14000582272, 14008973696, 14017363072, 14025752192,
14034137984, 14042528384, 14050918016, 14059301504, 14067691648,
14076083584, 14084470144, 14092852352, 14101249664, 14109635968,
14118024832, 14126407552, 14134804352, 14143188608, 14151577984,
14159968384, 14168357248, 14176741504, 14185127296, 14193521024,
14201911424, 14210301824, 14218685056, 14227067264, 14235467392,
14243855488, 14252243072, 14260630144, 14269021568, 14277409408,
14285799296, 14294187904, 14302571392, 14310961792, 14319353728,
14327738752, 14336130944, 14344518784, 14352906368, 14361296512,
14369685376, 14378071424, 14386462592, 14394848128, 14403230848,
14411627392, 14420013952, 14428402304, 14436793472, 14445181568,
14453569664, 14461959808, 14470347904, 14478737024, 14487122816,
14495511424, 14503901824, 14512291712, 14520677504, 14529064832,
14537456768, 14545845632, 14554234496, 14562618496, 14571011456,
14579398784, 14587789184, 14596172672, 14604564608, 14612953984,
14621341312, 14629724288, 14638120832, 14646503296, 14654897536,
14663284864, 14671675264, 14680061056, 14688447616, 14696835968,
14705228416, 14713616768, 14722003328, 14730392192, 14738784128,
14747172736, 14755561088, 14763947648, 14772336512, 14780725376,
14789110144, 14797499776, 14805892736, 14814276992, 14822670208,
14831056256, 14839444352, 14847836032, 14856222848, 14864612992,
14872997504, 14881388672, 14889775744, 14898165376, 14906553472,
14914944896, 14923329664, 14931721856, 14940109696, 14948497024,
14956887424, 14965276544, 14973663616, 14982053248, 14990439808,
14998830976, 15007216768, 15015605888, 15023995264, 15032385152,
15040768384, 15049154944, 15057549184, 15065939072, 15074328448,
15082715008, 15091104128, 15099493504, 15107879296, 15116269184,
15124659584, 15133042304, 15141431936, 15149824384, 15158214272,
15166602368, 15174991232, 15183378304, 15191760512, 15200154496,
15208542592, 15216931712, 15225323392, 15233708416, 15242098048,
15250489216, 15258875264, 15267265408, 15275654528, 15284043136,
15292431488, 15300819584, 15309208192, 15317596544, 15325986176,
15334374784, 15342763648, 15351151744, 15359540608, 15367929728,
15376318336, 15384706432, 15393092992, 15401481856, 15409869952,
15418258816, 15426649984, 15435037568, 15443425664, 15451815296,
15460203392, 15468589184, 15476979328, 15485369216, 15493755776,
15502146944, 15510534272, 15518924416, 15527311232, 15535699072,
15544089472, 15552478336, 15560866688, 15569254528, 15577642624,

15586031488, 15594419072, 15602809472, 15611199104, 15619586432,
15627975296, 15636364928, 15644753792, 15653141888, 15661529216,
15669918848, 15678305152, 15686696576, 15695083136, 15703474048,
15711861632, 15720251264, 15728636288, 15737027456, 15745417088,
15753804928, 15762194048, 15770582656, 15778971008, 15787358336,
15795747712, 15804132224, 15812523392, 15820909696, 15829300096,
15837691264, 15846071936, 15854466944, 15862855808, 15871244672,
15879634816, 15888020608, 15896409728, 15904799104, 15913185152,
15921577088, 15929966464, 15938354816, 15946743424, 15955129472,
15963519872, 15971907968, 15980296064, 15988684928, 15997073024,
16005460864, 16013851264, 16022241152, 16030629248, 16039012736,
16047406976, 16055794816, 16064181376, 16072571264, 16080957824,
16089346688, 16097737856, 16106125184, 16114514816, 16122904192,
16131292544, 16139678848, 16148066944, 16156453504, 16164839552,
16173236096, 16181623424, 16190012032, 16198401152, 16206790528,
16215177344, 16223567744, 16231956352, 16240344704, 16248731008,
16257117824, 16265504384, 16273898624, 16282281856, 16290668672,
16299064192, 16307449216, 16315842176, 16324230016, 16332613504,
16341006464, 16349394304, 16357783168, 16366172288, 16374561664,
16382951296, 16391337856, 16399726208, 16408116352, 16416505472,
16424892032, 16433282176, 16441668224, 16450058624, 16458448768,
16466836864, 16475224448, 16483613056, 16492001408, 16500391808,
16508779648, 16517166976, 16525555328, 16533944192, 16542330752,
16550719616, 16559110528, 16567497088, 16575888512, 16584274816,
16592665472, 16601051008, 16609442944, 16617832064, 16626218624,
16634607488, 16642996096, 16651385728, 16659773824, 16668163712,
16676552576, 16684938112, 16693328768, 16701718144, 16710095488,
16718492288, 16726883968, 16735272832, 16743661184, 16752049792,
16760436608, 16768827008, 16777214336, 16785599104, 16793992832,
16802381696, 16810768768, 16819151744, 16827542656, 16835934848,
16844323712, 16852711552, 16861101952, 16869489536, 16877876864,
16886265728, 16894653056, 16903044736, 16911431296, 16919821696,
16928207488, 16936592768, 16944987776, 16953375616, 16961763968,
16970152832, 16978540928, 16986929536, 16995319168, 17003704448,
17012096896, 17020481152, 17028870784, 17037262208, 17045649536,
17054039936, 17062426496, 17070814336, 17079205504, 17087592064,
17095978112, 17104369024, 17112759424, 17121147776, 17129536384,
17137926016, 17146314368, 17154700928, 17163089792, 17171480192,
17179864192, 17188256896, 17196644992, 17205033856, 17213423488,
17221811072, 17230198912, 17238588032, 17246976896, 17255360384,
17263754624, 17272143232, 17280530048, 17288918912, 17297309312,
17305696384, 17314085504, 17322475136, 17330863744, 17339252096,

17347640192, 17356026496, 17364413824, 17372796544, 17381190016,
17389583488, 17397972608, 17406360704, 17414748544, 17423135872,
17431527296, 17439915904, 17448303232, 17456691584, 17465081728,
17473468288, 17481857408, 17490247552, 17498635904, 17507022464,
17515409024, 17523801728, 17532189824, 17540577664, 17548966016,
17557353344, 17565741184, 17574131584, 17582519168, 17590907008,
17599296128, 17607687808, 17616076672, 17624455808, 17632852352,
17641238656, 17649630848, 17658018944, 17666403968, 17674794112,
17683178368, 17691573376, 17699962496, 17708350592, 17716739968,
17725126528, 17733517184, 17741898112, 17750293888, 17758673024,
17767070336, 17775458432, 17783848832, 17792236928, 17800625536,
17809012352, 17817402752, 17825785984, 17834178944, 17842563968,
17850955648, 17859344512, 17867732864, 17876119424, 17884511872,
17892900224, 17901287296, 17909677696, 17918058112, 17926451072,
17934843776, 17943230848, 17951609216, 17960008576, 17968397696,
17976784256, 17985175424, 17993564032, 18001952128, 18010339712,
18018728576, 18027116672, 18035503232, 18043894144, 18052283264,
18060672128, 18069056384, 18077449856, 18085837184, 18094225792,
18102613376, 18111004544, 18119388544, 18127781248, 18136170368,
18144558976, 18152947328, 18161336192, 18169724288, 18178108544,
18186498944, 18194886784, 18203275648, 18211666048, 18220048768,
18228444544, 18236833408, 18245220736}

// cacheSizes is a lookup table for the ethash verification cache size for the
// first 2048 epochs (i.e. 61440000 blocks).
var cacheSizes = []uint64{
16776896, 16907456, 17039296, 17170112, 17301056, 17432512, 17563072,
17693888, 17824192, 17955904, 18087488, 18218176, 18349504, 18481088,
18611392, 18742336, 18874304, 19004224, 19135936, 19267264, 19398208,
19529408, 19660096, 19791424, 19922752, 20053952, 20184896, 20315968,
20446912, 20576576, 20709184, 20840384, 20971072, 21102272, 21233216,
21364544, 21494848, 21626816, 21757376, 21887552, 22019392, 22151104,
22281536, 22412224, 22543936, 22675264, 22806464, 22935872, 23068096,
23198272, 23330752, 23459008, 23592512, 23723968, 23854912, 23986112,
24116672, 24247616, 24378688, 24509504, 24640832, 24772544, 24903488,
25034432, 25165376, 25296704, 25427392, 25558592, 25690048, 25820096,
25951936, 26081728, 26214208, 26345024, 26476096, 26606656, 26737472,
26869184, 26998208, 27131584, 27262528, 27393728, 27523904, 27655744,
27786688, 27917888, 28049344, 28179904, 28311488, 28441792, 28573504,
28700864, 28835648, 28966208, 29096768, 29228608, 29359808, 29490752,
29621824, 29752256, 29882816, 30014912, 30144448, 30273728, 30406976,
30538432, 30670784, 30799936, 30932672, 31063744, 31195072, 31325248,

31456192, 31588288, 31719232, 31850432, 31981504, 32110784, 32243392,
32372672, 32505664, 32636608, 32767808, 32897344, 33029824, 33160768,
33289664, 33423296, 33554368, 33683648, 33816512, 33947456, 34076992,
34208704, 34340032, 34471744, 34600256, 34734016, 34864576, 34993984,
35127104, 35258176, 35386688, 35518528, 35650624, 35782336, 35910976,
36044608, 36175808, 36305728, 36436672, 36568384, 36699968, 36830656,
36961984, 37093312, 37223488, 37355072, 37486528, 37617472, 37747904,
37879232, 38009792, 38141888, 38272448, 38403392, 38535104, 38660672,
38795584, 38925632, 39059264, 39190336, 39320768, 39452096, 39581632,
39713984, 39844928, 39974848, 40107968, 40238144, 40367168, 40500032,
40631744, 40762816, 40894144, 41023552, 41155904, 41286208, 41418304,
41547712, 41680448, 41811904, 41942848, 42073792, 42204992, 42334912,
42467008, 42597824, 42729152, 42860096, 42991552, 43122368, 43253696,
43382848, 43515712, 43646912, 43777088, 43907648, 44039104, 44170432,
44302144, 44433344, 44564288, 44694976, 44825152, 44956864, 45088448,
45219008, 45350464, 45481024, 45612608, 45744064, 45874496, 46006208,
46136768, 46267712, 46399424, 46529344, 46660672, 46791488, 46923328,
47053504, 47185856, 47316928, 47447872, 47579072, 47710144, 47839936,
47971648, 48103232, 48234176, 48365248, 48496192, 48627136, 48757312,
48889664, 49020736, 49149248, 49283008, 49413824, 49545152, 49675712,
49807168, 49938368, 50069056, 50200256, 50331584, 50462656, 50593472,
50724032, 50853952, 50986048, 51117632, 51248576, 51379904, 51510848,
51641792, 51773248, 51903296, 52035136, 52164032, 52297664, 52427968,
52557376, 52690112, 52821952, 52952896, 53081536, 53213504, 53344576,
53475776, 53608384, 53738816, 53870528, 54000832, 54131776, 54263744,
54394688, 54525248, 54655936, 54787904, 54918592, 55049152, 55181248,
55312064, 55442752, 55574336, 55705024, 55836224, 55967168, 56097856,
56228672, 56358592, 56490176, 56621888, 56753728, 56884928, 57015488,
57146816, 57278272, 57409216, 57540416, 57671104, 57802432, 57933632,
58064576, 58195264, 58326976, 58457408, 58588864, 58720192, 58849984,
58981696, 59113024, 59243456, 59375552, 59506624, 59637568, 59768512,
59897792, 60030016, 60161984, 60293056, 60423872, 60554432, 60683968,
60817216, 60948032, 61079488, 61209664, 61341376, 61471936, 61602752,
61733696, 61865792, 61996736, 62127808, 62259136, 62389568, 62520512,
62651584, 62781632, 62910784, 63045056, 63176128, 63307072, 63438656,
63569216, 63700928, 63831616, 63960896, 64093888, 64225088, 64355392,
64486976, 64617664, 64748608, 64879424, 65009216, 65142464, 65273792,
65402816, 65535424, 65666752, 65797696, 65927744, 66060224, 66191296,
66321344, 66453056, 66584384, 66715328, 66846656, 66977728, 67108672,
67239104, 67370432, 67501888, 67631296, 67763776, 67895104, 68026304,
68157248, 68287936, 68419264, 68548288, 68681408, 68811968, 68942912,
69074624, 69205568, 69337024, 69467584, 69599168, 69729472, 69861184,

69989824, 70122944, 70253888, 70385344, 70515904, 70647232, 70778816,
70907968, 71040832, 71171648, 71303104, 71432512, 71564992, 71695168,
71826368, 71958464, 72089536, 72219712, 72350144, 72482624, 72613568,
72744512, 72875584, 73006144, 73138112, 73268672, 73400128, 73530944,
73662272, 73793344, 73924544, 74055104, 74185792, 74316992, 74448832,
74579392, 74710976, 74841664, 74972864, 75102784, 75233344, 75364544,
75497024, 75627584, 75759296, 75890624, 76021696, 76152256, 76283072,
76414144, 76545856, 76676672, 76806976, 76937792, 77070016, 77200832,
77331392, 77462464, 77593664, 77725376, 77856448, 77987776, 78118336,
78249664, 78380992, 78511424, 78642496, 78773056, 78905152, 79033664,
79166656, 79297472, 79429568, 79560512, 79690816, 79822784, 79953472,
80084672, 80214208, 80346944, 80477632, 80608576, 80740288, 80870848,
81002048, 81133504, 81264448, 81395648, 81525952, 81657536, 81786304,
81919808, 82050112, 82181312, 82311616, 82443968, 82573376, 82705984,
82835776, 82967744, 83096768, 83230528, 83359552, 83491264, 83622464,
83753536, 83886016, 84015296, 84147776, 84277184, 84409792, 84540608,
84672064, 84803008, 84934336, 85065152, 85193792, 85326784, 85458496,
85589312, 85721024, 85851968, 85982656, 86112448, 86244416, 86370112,
86506688, 86637632, 86769344, 86900672, 87031744, 87162304, 87293632,
87424576, 87555392, 87687104, 87816896, 87947968, 88079168, 88211264,
88341824, 88473152, 88603712, 88735424, 88862912, 88996672, 89128384,
89259712, 89390272, 89521984, 89652544, 89783872, 89914816, 90045376,
90177088, 90307904, 90438848, 90569152, 90700096, 90832832, 90963776,
91093696, 91223744, 91356992, 91486784, 91618496, 91749824, 91880384,
92012224, 92143552, 92273344, 92405696, 92536768, 92666432, 92798912,
92926016, 93060544, 93192128, 93322816, 93453632, 93583936, 93715136,
93845056, 93977792, 94109504, 94240448, 94371776, 94501184, 94632896,
94764224, 94895552, 95023424, 95158208, 95287744, 95420224, 95550016,
95681216, 95811904, 95943872, 96075328, 96203584, 96337856, 96468544,
96599744, 96731072, 96860992, 96992576, 97124288, 97254848, 97385536,
97517248, 97647808, 97779392, 97910464, 98041408, 98172608, 98303168,
98434496, 98565568, 98696768, 98827328, 98958784, 99089728, 99220928,
99352384, 99482816, 99614272, 99745472, 99876416, 100007104,
100138048, 100267072, 100401088, 100529984, 100662592, 100791872,
100925248, 101056064, 101187392, 101317952, 101449408, 101580608,
101711296, 101841728, 101973824, 102104896, 102235712, 102366016,
102498112, 102628672, 102760384, 102890432, 103021888, 103153472,
103284032, 103415744, 103545152, 103677248, 103808576, 103939648,
104070976, 104201792, 104332736, 104462528, 104594752, 104725952,
104854592, 104988608, 105118912, 105247808, 105381184, 105511232,
105643072, 105774784, 105903296, 106037056, 106167872, 106298944,
106429504, 106561472, 106691392, 106822592, 106954304, 107085376,

107216576, 107346368, 107478464, 107609792, 107739712, 107872192,
108003136, 108131392, 108265408, 108396224, 108527168, 108657344,
108789568, 108920384, 109049792, 109182272, 109312576, 109444928,
109572928, 109706944, 109837888, 109969088, 110099648, 110230976,
110362432, 110492992, 110624704, 110755264, 110886208, 111017408,
111148864, 111279296, 111410752, 111541952, 111673024, 111803456,
111933632, 112066496, 112196416, 112328512, 112457792, 112590784,
112715968, 112852672, 112983616, 113114944, 113244224, 113376448,
113505472, 113639104, 113770304, 113901376, 114031552, 114163264,
114294592, 114425536, 114556864, 114687424, 114818624, 114948544,
115080512, 115212224, 115343296, 115473472, 115605184, 115736128,
115867072, 115997248, 116128576, 116260288, 116391488, 116522944,
116652992, 116784704, 116915648, 117046208, 117178304, 117308608,
117440192, 117569728, 117701824, 117833024, 117964096, 118094656,
118225984, 118357312, 118489024, 118617536, 118749632, 118882112,
119012416, 119144384, 119275328, 119406016, 119537344, 119668672,
119798464, 119928896, 120061376, 120192832, 120321728, 120454336,
120584512, 120716608, 120848192, 120979136, 121109056, 121241408,
121372352, 121502912, 121634752, 121764416, 121895744, 122027072,
122157632, 122289088, 122421184, 122550592, 122682944, 122813888,
122945344, 123075776, 123207488, 123338048, 123468736, 123600704,
123731264, 123861952, 123993664, 124124608, 124256192, 124386368,
124518208, 124649024, 124778048, 124911296, 125041088, 125173696,
125303744, 125432896, 125566912, 125696576, 125829056, 125958592,
126090304, 126221248, 126352832, 126483776, 126615232, 126746432,
126876608, 127008704, 127139392, 127270336, 127401152, 127532224,
127663552, 127794752, 127925696, 128055232, 128188096, 128319424,
128449856, 128581312, 128712256, 128843584, 128973632, 129103808,
129236288, 129365696, 129498944, 129629888, 129760832, 129892288,
130023104, 130154048, 130283968, 130416448, 130547008, 130678336,
130807616, 130939456, 131071552, 131202112, 131331776, 131464384,
131594048, 131727296, 131858368, 131987392, 132120256, 132250816,
132382528, 132513728, 132644672, 132774976, 132905792, 133038016,
133168832, 133299392, 133429312, 133562048, 133692992, 133823296,
133954624, 134086336, 134217152, 134348608, 134479808, 134607296,
134741056, 134872384, 135002944, 135134144, 135265472, 135396544,
135527872, 135659072, 135787712, 135921472, 136052416, 136182848,
136313792, 136444864, 136576448, 136707904, 136837952, 136970048,
137099584, 137232064, 137363392, 137494208, 137625536, 137755712,
137887424, 138018368, 138149824, 138280256, 138411584, 138539584,
138672832, 138804928, 138936128, 139066688, 139196864, 139328704,
139460032, 139590208, 139721024, 139852864, 139984576, 140115776,

140245696, 140376512, 140508352, 140640064, 140769856, 140902336,
141032768, 141162688, 141294016, 141426496, 141556544, 141687488,
141819584, 141949888, 142080448, 142212544, 142342336, 142474432,
142606144, 142736192, 142868288, 142997824, 143129408, 143258944,
143392448, 143523136, 143653696, 143785024, 143916992, 144045632,
144177856, 144309184, 144440768, 144570688, 144701888, 144832448,
144965056, 145096384, 145227584, 145358656, 145489856, 145620928,
145751488, 145883072, 146011456, 146144704, 146275264, 146407232,
146538176, 146668736, 146800448, 146931392, 147062336, 147193664,
147324224, 147455936, 147586624, 147717056, 147848768, 147979456,
148110784, 148242368, 148373312, 148503232, 148635584, 148766144,
148897088, 149028416, 149159488, 149290688, 149420224, 149551552,
149683136, 149814976, 149943616, 150076352, 150208064, 150338624,
150470464, 150600256, 150732224, 150862784, 150993088, 151125952,
151254976, 151388096, 151519168, 151649728, 151778752, 151911104,
152042944, 152174144, 152304704, 152435648, 152567488, 152698816,
152828992, 152960576, 153091648, 153222976, 153353792, 153484096,
153616192, 153747008, 153878336, 154008256, 154139968, 154270912,
154402624, 154533824, 154663616, 154795712, 154926272, 155057984,
155188928, 155319872, 155450816, 155580608, 155712064, 155843392,
155971136, 156106688, 156237376, 156367424, 156499264, 156630976,
156761536, 156892352, 157024064, 157155008, 157284416, 157415872,
157545536, 157677248, 157810496, 157938112, 158071744, 158203328,
158334656, 158464832, 158596288, 158727616, 158858048, 158988992,
159121216, 159252416, 159381568, 159513152, 159645632, 159776192,
159906496, 160038464, 160169536, 160300352, 160430656, 160563008,
160693952, 160822208, 160956352, 161086784, 161217344, 161349184,
161480512, 161611456, 161742272, 161873216, 162002752, 162135872,
162266432, 162397888, 162529216, 162660032, 162790976, 162922048,
163052096, 163184576, 163314752, 163446592, 163577408, 163707968,
163839296, 163969984, 164100928, 164233024, 164364224, 164494912,
164625856, 164756672, 164887616, 165019072, 165150016, 165280064,
165412672, 165543104, 165674944, 165805888, 165936832, 166067648,
166198336, 166330048, 166461248, 166591552, 166722496, 166854208,
166985408, 167116736, 167246656, 167378368, 167508416, 167641024,
167771584, 167903168, 168034112, 168164032, 168295744, 168427456,
168557632, 168688448, 168819136, 168951616, 169082176, 169213504,
169344832, 169475648, 169605952, 169738048, 169866304, 169999552,
170131264, 170262464, 170393536, 170524352, 170655424, 170782016,
170917696, 171048896, 171179072, 171310784, 171439936, 171573184,
171702976, 171835072, 171966272, 172097216, 172228288, 172359232,
172489664, 172621376, 172747712, 172883264, 173014208, 173144512,

173275072, 173407424, 173539136, 173669696, 173800768, 173931712,
174063424, 174193472, 174325696, 174455744, 174586816, 174718912,
174849728, 174977728, 175109696, 175242688, 175374272, 175504832,
175636288, 175765696, 175898432, 176028992, 176159936, 176291264,
176422592, 176552512, 176684864, 176815424, 176946496, 177076544,
177209152, 177340096, 177470528, 177600704, 177731648, 177864256,
177994816, 178126528, 178257472, 178387648, 178518464, 178650176,
178781888, 178912064, 179044288, 179174848, 179305024, 179436736,
179568448, 179698496, 179830208, 179960512, 180092608, 180223808,
180354752, 180485696, 180617152, 180748096, 180877504, 181009984,
181139264, 181272512, 181402688, 181532608, 181663168, 181795136,
181926592, 182057536, 182190016, 182320192, 182451904, 182582336,
182713792, 182843072, 182976064, 183107264, 183237056, 183368384,
183494848, 183631424, 183762752, 183893824, 184024768, 184154816,
184286656, 184417984, 184548928, 184680128, 184810816, 184941248,
185072704, 185203904, 185335616, 185465408, 185596352, 185727296,
185859904, 185989696, 186121664, 186252992, 186383552, 186514112,
186645952, 186777152, 186907328, 187037504, 187170112, 187301824,
187429184, 187562048, 187693504, 187825472, 187957184, 188087104,
188218304, 188349376, 188481344, 188609728, 188743616, 188874304,
189005248, 189136448, 189265088, 189396544, 189528128, 189660992,
189791936, 189923264, 190054208, 190182848, 190315072, 190447424,
190577984, 190709312, 190840768, 190971328, 191102656, 191233472,
191364032, 191495872, 191626816, 191758016, 191888192, 192020288,
192148928, 192282176, 192413504, 192542528, 192674752, 192805952,
192937792, 193068608, 193198912, 193330496, 193462208, 193592384,
193723456, 193854272, 193985984, 194116672, 194247232, 194379712,
194508352, 194641856, 194772544, 194900672, 195035072, 195166016,
195296704, 195428032, 195558592, 195690304, 195818176, 195952576,
196083392, 196214336, 196345792, 196476736, 196607552, 196739008,
196869952, 197000768, 197130688, 197262784, 197394368, 197523904,
197656384, 197787584, 197916608, 198049472, 198180544, 198310208,
198442432, 198573632, 198705088, 198834368, 198967232, 199097792,
199228352, 199360192, 199491392, 199621696, 199751744, 199883968,
200014016, 200146624, 200276672, 200408128, 200540096, 200671168,
200801984, 200933312, 201062464, 201194944, 201326144, 201457472,
201588544, 201719744, 201850816, 201981632, 202111552, 202244032,
202374464, 202505152, 202636352, 202767808, 202898368, 203030336,
203159872, 203292608, 203423296, 203553472, 203685824, 203816896,
203947712, 204078272, 204208192, 204341056, 204472256, 204603328,
204733888, 204864448, 204996544, 205125568, 205258304, 205388864,
205517632, 205650112, 205782208, 205913536, 206044736, 206176192,

206307008, 206434496, 206569024, 206700224, 206831168, 206961856,
207093056, 207223616, 207355328, 207486784, 207616832, 207749056,
207879104, 208010048, 208141888, 208273216, 208404032, 208534336,
208666048, 208796864, 208927424, 209059264, 209189824, 209321792,
209451584, 209582656, 209715136, 209845568, 209976896, 210106432,
210239296, 210370112, 210501568, 210630976, 210763712, 210894272,
211024832, 211156672, 211287616, 211418176, 211549376, 211679296,
211812032, 211942592, 212074432, 212204864, 212334016, 212467648,
212597824, 212727616, 212860352, 212991424, 213120832, 213253952,
213385024, 213515584, 213645632, 213777728, 213909184, 214040128,
214170688, 214302656, 214433728, 214564544, 214695232, 214826048,
214956992, 215089088, 215219776, 215350592, 215482304, 215613248,
215743552, 215874752, 216005312, 216137024, 216267328, 216399296,
216530752, 216661696, 216790592, 216923968, 217054528, 217183168,
217316672, 217448128, 217579072, 217709504, 217838912, 217972672,
218102848, 218233024, 218364736, 218496832, 218627776, 218759104,
218888896, 219021248, 219151936, 219281728, 219413056, 219545024,
219675968, 219807296, 219938624, 220069312, 220200128, 220331456,
220461632, 220592704, 220725184, 220855744, 220987072, 221117888,
221249216, 221378368, 221510336, 221642048, 221772736, 221904832,
222031808, 222166976, 222297536, 222428992, 222559936, 222690368,
222820672, 222953152, 223083968, 223213376, 223345984, 223476928,
223608512, 223738688, 223869376, 224001472, 224132672, 224262848,
224394944, 224524864, 224657344, 224788288, 224919488, 225050432,
225181504, 225312704, 225443776, 225574592, 225704768, 225834176,
225966784, 226097216, 226229824, 226360384, 226491712, 226623424,
226754368, 226885312, 227015104, 227147456, 227278528, 227409472,
227539904, 227669696, 227802944, 227932352, 228065216, 228196288,
228326464, 228457792, 228588736, 228720064, 228850112, 228981056,
229113152, 229243328, 229375936, 229505344, 229636928, 229769152,
229894976, 230030272, 230162368, 230292416, 230424512, 230553152,
230684864, 230816704, 230948416, 231079616, 231210944, 231342016,
231472448, 231603776, 231733952, 231866176, 231996736, 232127296,
232259392, 232388672, 232521664, 232652608, 232782272, 232914496,
233043904, 233175616, 233306816, 233438528, 233569984, 233699776,
233830592, 233962688, 234092224, 234221888, 234353984, 234485312,
234618304, 234749888, 234880832, 235011776, 235142464, 235274048,
235403456, 235535936, 235667392, 235797568, 235928768, 236057152,
236190272, 236322752, 236453312, 236583616, 236715712, 236846528,
236976448, 237108544, 237239104, 237371072, 237501632, 237630784,
237764416, 237895232, 238026688, 238157632, 238286912, 238419392,
238548032, 238681024, 238812608, 238941632, 239075008, 239206336,

239335232, 239466944, 239599168, 239730496, 239861312, 239992384,
240122816, 240254656, 240385856, 240516928, 240647872, 240779072,
240909632, 241040704, 241171904, 241302848, 241433408, 241565248,
241696192, 241825984, 241958848, 242088256, 242220224, 242352064,
242481856, 242611648, 242744896, 242876224, 243005632, 243138496,
243268672, 243400384, 243531712, 243662656, 243793856, 243924544,
244054592, 244187072, 244316608, 244448704, 244580032, 244710976,
244841536, 244972864, 245104448, 245233984, 245365312, 245497792,
245628736, 245759936, 245889856, 246021056, 246152512, 246284224,
246415168, 246545344, 246675904, 246808384, 246939584, 247070144,
247199552, 247331648, 247463872, 247593536, 247726016, 247857088,
247987648, 248116928, 248249536, 248380736, 248512064, 248643008,
248773312, 248901056, 249036608, 249167552, 249298624, 249429184,
249560512, 249692096, 249822784, 249954112, 250085312, 250215488,
250345792, 250478528, 250608704, 250739264, 250870976, 251002816,
251133632, 251263552, 251395136, 251523904, 251657792, 251789248,
251919424, 252051392, 252182464, 252313408, 252444224, 252575552,
252706624, 252836032, 252968512, 253099712, 253227584, 253361728,
253493056, 253623488, 253754432, 253885504, 254017216, 254148032,
254279488, 254410432, 254541376, 254672576, 254803264, 254933824,
255065792, 255196736, 255326528, 255458752, 255589952, 255721408,
255851072, 255983296, 256114624, 256244416, 256374208, 256507712,
256636096, 256768832, 256900544, 257031616, 257162176, 257294272,
257424448, 257555776, 257686976, 257818432, 257949632, 258079552,
258211136, 258342464, 258473408, 258603712, 258734656, 258867008,
258996544, 259127744, 259260224, 259391296, 259522112, 259651904,
259784384, 259915328, 260045888, 260175424, 260308544, 260438336,
260570944, 260700992, 260832448, 260963776, 261092672, 261226304,
261356864, 261487936, 261619648, 261750592, 261879872, 262011968,
262143424, 262274752, 262404416, 262537024, 262667968, 262799296,
262928704, 263061184, 263191744, 263322944, 263454656, 263585216,
263716672, 263847872, 263978944, 264108608, 264241088, 264371648,
264501184, 264632768, 264764096, 264895936, 265024576, 265158464,
265287488, 265418432, 265550528, 265681216, 265813312, 265943488,
266075968, 266206144, 266337728, 266468032, 266600384, 266731072,
266862272, 266993344, 267124288, 267255616, 267386432, 267516992,
267648704, 267777728, 267910592, 268040512, 268172096, 268302784,
268435264, 268566208, 268696256, 268828096, 268959296, 269090368,
269221312, 269352256, 269482688, 269614784, 269745856, 269876416,
270007616, 270139328, 270270272, 270401216, 270531904, 270663616,
270791744, 270924736, 271056832, 271186112, 271317184, 271449536,
271580992, 271711936, 271843136, 271973056, 272105408, 272236352,

272367296, 272498368, 272629568, 272759488, 272891456, 273022784,
273153856, 273284672, 273415616, 273547072, 273677632, 273808448,
273937088, 274071488, 274200896, 274332992, 274463296, 274595392,
274726208, 274857536, 274988992, 275118656, 275250496, 275382208,
275513024, 275643968, 275775296, 275906368, 276037184, 276167872,
276297664, 276429376, 276560576, 276692672, 276822976, 276955072,
277085632, 277216832, 277347008, 277478848, 277609664, 277740992,
277868608, 278002624, 278134336, 278265536, 278395328, 278526784,
278657728, 278789824, 278921152, 279052096, 279182912, 279313088,
279443776, 279576256, 279706048, 279838528, 279969728, 280099648,
280230976, 280361408, 280493632, 280622528, 280755392, 280887104,
281018176, 281147968, 281278912, 281411392, 281542592, 281673152,
281803712, 281935552, 282066496, 282197312, 282329024, 282458816,
282590272, 282720832, 282853184, 282983744, 283115072, 283246144,
283377344, 283508416, 283639744, 283770304, 283901504, 284032576,
284163136, 284294848, 284426176, 284556992, 284687296, 284819264,
284950208, 285081536}

29:F:\git\coin\ethereum\go-ethereum\consensus\ethash\algorithm_go1.7.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// +build !go1.8

package ethash

// cacheSize calculates and returns the size of the ethash verification cache that
// belongs to a certain block number. The cache size grows linearly, however, we
// always take the highest prime below the linearly growing threshold in order to
// reduce the risk of accidental regularities leading to cyclic behavior.
func cacheSize(block uint64) uint64 {
// If we have a pre-generated value, use that
epoch := int(block / epochLength)
if epoch < len(cacheSizes) {
return cacheSizes[epoch]
}
// We don't have a way to verify primes fast before Go 1.8
panic("fast prime testing unsupported in Go < 1.8")
}

// datasetSize calculates and returns the size of the ethash mining dataset that
// belongs to a certain block number. The dataset size grows linearly, however, we
// always take the highest prime below the linearly growing threshold in order to

```
// reduce the risk of accidental regularities leading to cyclic behavior.
func datasetSize(block uint64) uint64 {
// If we have a pre-generated value, use that
epoch := int(block / epochLength)
if epoch < len(datasetSizes) {
return datasetSizes[epoch]
}
// We don't have a way to verify primes fast before Go 1.8
panic("fast prime testing unsupported in Go < 1.8")
}

30:F:\git\coin\ethereum\go-ethereum\consensus\ethash\algorithm_go1.8.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// +build go1.8

package ethash

import "math/big"

// cacheSize calculates and returns the size of the ethash verification cache that
// belongs to a certain block number. The cache size grows linearly, however, we
// always take the highest prime below the linearly growing threshold in order to
// reduce the risk of accidental regularities leading to cyclic behavior.
func cacheSize(block uint64) uint64 {
// If we have a pre-generated value, use that
epoch := int(block / epochLength)
if epoch < len(cacheSizes) {
return cacheSizes[epoch]
}
// No known cache size, calculate manually (sanity branch only)
size := uint64(cacheInitBytes + cacheGrowthBytes*uint64(epoch) - hashBytes)
for !new(big.Int).SetUint64(size / hashBytes).ProbablyPrime(1) { // Always accurate for n < 2^64
size -= 2 * hashBytes
}
return size
}

// datasetSize calculates and returns the size of the ethash mining dataset that
// belongs to a certain block number. The dataset size grows linearly, however, we
// always take the highest prime below the linearly growing threshold in order to
// reduce the risk of accidental regularities leading to cyclic behavior.
```

```
func datasetSize(block uint64) uint64 {
// If we have a pre-generated value, use that
epoch := int(block / epochLength)
if epoch < len(datasetSizes) {
return datasetSizes[epoch]
}
// No known dataset size, calculate manually (sanity branch only)
size := uint64(datasetInitBytes + datasetGrowthBytes*uint64(epoch) - mixBytes)
for !new(big.Int).SetUint64(size / mixBytes).ProbablyPrime(1) { // Always accurate for n < 2^64
size -= 2 * mixBytes
}
return size
}

31:F:\git\coin\ethereum\go-ethereum\consensus\ethash\algorithm_go1.8_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// +build go1.8

package ethash

import "testing"

// Tests whether the dataset size calculator works correctly by cross checking the
// hard coded lookup table with the value generated by it.
func TestSizeCalculations(t *testing.T) {
var tests []uint64

// Verify all the cache sizes from the lookup table
defer func(sizes []uint64) { cacheSizes = sizes }(cacheSizes)
tests, cacheSizes = cacheSizes, []uint64{}

for i, test := range tests {
if size := cacheSize(uint64(i*epochLength) + 1); size != test {
t.Errorf("cache %d: cache size mismatch: have %d, want %d", i, size, test)
}
}
// Verify all the dataset sizes from the lookup table
defer func(sizes []uint64) { datasetSizes = sizes }(datasetSizes)
tests, datasetSizes = datasetSizes, []uint64{}

for i, test := range tests {
```

```
if size := datasetSize(uint64(i*epochLength) + 1); size != test {
t.Errorf("dataset %d: dataset size mismatch: have %d, want %d", i, size, test)
}
}
}
```

32:F:\git\coin\ethereum\go-ethereum\consensus\ethash\algorithm_test.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ethash

import (
"bytes"
"io/ioutil"
"math/big"
"os"
"reflect"
"sync"
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/hexutil"
"github.com/ethereum/go-ethereum/core/types"
)

// Tests that verification caches can be correctly generated.
func TestCacheGeneration(t *testing.T) {
tests := []struct {
size  uint64
epoch uint64
cache []byte
}{
{
size:  1024,
epoch: 0,
cache: hexutil.MustDecode("0x" +
"7ce2991c951f7bf4c4c1bb119887ee07871eb5339d7b97b8588e85c742de90e5bafd5bbe6ce93a1
34fb6be9ad3e30db99d9528a2ea7846833f52e9ca119b6b54" +
"8979480c46e19972bd0738779c932c1b43e665a2fd3122fc3ddb2691f353ceb0ed3e38b8f51fd55b
6940290743563c9f8fa8822e611924657501a12aafab8a8d" +
"88fb5fbae3a99d14792406672e783a06940a42799b1c38bc28715db6d37cb11f9f6b24e386dc52dd
8c286bd8c36fa813dffe4448a9f56ebcbeea866b42f68d22" +
```

"6c32aae4d695a23cab28fd74af53b0c2efcc180ceaaccc0b2e280103d097a03c1d1b0f0f26ce5f32a
90238f9bc49f645db001ef9cd3d13d44743f841fad11a37" +
"fa290c62c16042f703578921f30b9951465aae2af4a5dad43a7341d7b4a62750954965a47a1c3af6
38dc3495c4d62a9bab843168c9fc0114e79cffd1b2827b01" +
"75d30ba054658f214e946cf24c43b40d3383fbb0493408e5c5392434ca21bbcf43200dfb876c713d
201813934fa485f48767c5915745cf0986b1dc0f33e57748" +
"bf483ee2aff4248dfe461ec0504a13628401020fc22638584a8f2f5206a13b2f233898c78359b21c82
26024d0a7a93df5eb6c282bdbf005a4aab497e096f2847" +
"76c71cee57932a8fb89f6d6b8743b60a4ea374899a94a2e0f218d5c55818cefb1790c8529a76dba3
1ebb0f4592d709b49587d2317970d39c086f18dd244291d9" +
"eedb16705e53e3350591bd4ff4566a3595ac0f0ce24b5e112a3d033bc51b6fea0a92296dea7f5e20
bf6ee6bc347d868fda193c395b9bb147e55e5a9f67cfe741" +
"7eea7d699b155bd13804204df7ea91fa9249e4474dddf35188f77019c67d201e4c10d7079c5ad492
a71afff9a23ca7e900ba7d1bdeaf3270514d8eb35eab8a0a" +
"718bb7273aeb37768fa589ed8ab01fbf4027f4ebdbbae128d21e485f061c20183a9bc2e31edbda07
27442e9d58eb0fe198440fe199e02e77c0f7b99973f1f74c" +
"c9089a51ab96c94a84d66e6aa48b2d0a4543adb5a789039a2aa7b335ca85c91026c7d3c894da53
ae364188c3fd92f78e01d080399884a47385aa792e38150cda" +
"a8620b2ebeca41fbc773bb837b5e724d6eb2de570d99858df0d7d97067fb8103b21757873b73509
7b35d3bea8fd1c359a9e8a63c1540c76c9784cf8d975e995c" +
"778401b94a2e66e6993ad67ad3ecdc2acb17779f1ea8606827ec92b11c728f8c3b6d3f04a3e6ed0
5ff81dd76d5dc5695a50377bc135aaf1671cf68b750315493" +
"6c64510164d53312bf3c41740c7a237b05faf4a191bd8a95dafa068dbcf370255c725900ce5c934f3
6feadcfe55b687c440574c1f06f39d207a8553d39156a24" +
"845f64fd8324bb85312979dead74f764c9677aab89801ad4f927f1c00f12e28f22422bb44200d1969
d9ab377dd6b099dc6dbc3222e9321b2c1e84f8e2f07731c"),
},
{
size: 1024,
epoch: 1,
cache: hexutil.MustDecode("0x" +
"1f56855d59cc5a085720899b4377a0198f1abe948d85fe5820dc0e346b7c0931b9cde8e541d751d
e3b2b3275d0aabfae316209d5879297d8bd99f8a033c9d4df" +
"35add1029f4e6404a022d504fb8023e42989aba985a65933b0109c7218854356f9284983c9e7de9
7de591828ae348b63d1fc78d8db58157344d4e06530ffd422" +
"5c7f6080d451ff94961ec2dd9e28e6d81b49102451676dbdcb6ef1094c1e8b29e7e808d47b2ba5ae
b52dabf00d5f0ee08c116289cbf56d8132e5ca557c3d6220" +
"5ba3a48539acabfd4ca3c89e3aaa668e24ffeaeb9eb0136a9fc5a8a676b6d5ad76175eeda0a1fa44
b5ff5591079e4b7f581569b6c82416adcb82d7e92980df67" +
"2248c4024013e7be52cf91a82491627d9e6d80eda2770ab82badc5e120cd33a4c84495f718b5739
6a8f397e797087fad81fa50f0e2f5da71e40816a85de35a96" +
"3cd351364905c45b3116ff25851d43a2ca1d2aa5cdb408440dabef8c57778fc18608bf431d0c7ffd3

```
7649a21a7bb9d90def39c821669dbaf165c0262434dfb08" +
"5d057a12de4a7a59fd2dfc931c29c20371abf748b69b618a9bd485b3fb3166cad4d3d27edf0197aa
beceb28b96670bdf020f26d1bb9b564aaf82d866bdffd6d4" +
"1aea89e20b15a5d1264ab01d1556bfc2a266081609d60928216bd9646038f07de9fedcc9f2b86ab1
b07d7bd88ba1df08b3d89b2ac789001b48a723f217debcb7" +
"090303a3ef50c1d5d99a75c640ec2b401ab149e06511753d8c49cafdde2929ae61e09cc0f0319d2
62869d21ead9e0cf5ff2de3dbedfb994f32432d2e4aa44c82" +
"7c42781d1477fe03ea0772998e776d63363c6c3edd2d52c89b4d2c9d89cdd90fa33b2b41c8e3f78
ef06fe90bcf5cc5756d33a032f16b744141aaa8852bb4cb3a" +
"40792b93489c6d6e56c235ec4aa36c263e9b766a4daaff34b2ea709f9f811aef498a65bfbc1deffd36
fcc4d1a123345fac7bf57a1fb50394843cd28976a6c7ff" +
"fe70f7b8d8f384aa06e2c9964c92a8788cef397fffdd35181b42a35d5d98cd7244bbd09e802888d7ef
c0311ae58e0961e3656205df4bdc553f317df4b6ede4ca" +
"846294a32aec830ab1aa5aac4e78b821c35c70fd752fec353e373bf9be656e775a0111bcbeffdfebd
3bd5251d27b9f6971aa561a2bd27a99d61b2ce3965c3726" +
"1e114353e6a31b09340f4078b8a8c6ce6ff4213067a8f21020f78aff4f8b472b701ef730aacb8ce780
6ea31b14abe8f8efdd6357ca299d339abc4e43ba324ad1" +
"efe6eb1a5a6e137daa6ec9f6be30931ca368a944cfcf2a0a29f9a9664188f0466e6f078c347f9fe26a
9a89d2029462b19245f24ace47aecace6ef85a4e96b31b" +
"5f470eb0165c6375eb8f245d50a25d521d1e569e3b2dccce626752bb26eae624a24511e831a81fa
b6898a791579f462574ca4851e6588116493dbccc3072e0c5"),
},
}
for i, tt := range tests {
cache := make([]uint32, tt.size/4)
generateCache(cache, tt.epoch, seedHash(tt.epoch*epochLength+1))

want := make([]uint32, tt.size/4)
prepare(want, tt.cache)

if !reflect.DeepEqual(cache, want) {
t.Errorf("cache %d: content mismatch: have %x, want %x", i, cache, want)
}
}
}

func TestDatasetGeneration(t *testing.T) {
tests := []struct {
epoch       uint64
cacheSize   uint64
datasetSize uint64
dataset     []byte
```

```
}{
{
epoch:       0,
cacheSize:   1024,
datasetSize: 32 * 1024,
dataset: hexutil.MustDecode("0x" +
"4bc09fbd530a041dd2ec296110a29e8f130f179c59d223f51ecce3126e8b0c74326abc2f32ccd9d7f
976bd0944e3ccf8479db39343cbbffa467046ca97e2da63" +
"da5f9d9688c7c33ab7b8aace570e422fa48b24659b72fc534669209d66389ca15b099c5604601e7
581488e3bd6925cec0f12d465f8004d4fa84793f8e1e46a1b" +
"31b7298991c6142f4f0b6e6b296728ae5fa63ccb667b61fbb1b078003d18d97b906af157debed5e6
c55d5a61cae90c85f9e97d565314a2f9fd9e0c08430547d0" +
"7cfcee3271f921b95c32a11596219abaa30abc62c2c72c6725078c436c677320594df6bcb92134c1
b114fffec982a1f68f13a9f812f074b9fb9c78f2cd4c1c90" +
"7ebf1e447f7a422b06303921e3d54f430584d849eaa4b7d652e92a5d659bdfc462adcdd7991e8c66
a19da4ddb5390463d073941491859397f135ebbbdbdf5801" +
"cafb873c383893390141ae385515504d74a33608273310c312ba468046d2e20c271a38cc0e3920
b39705050e752f34f244fc23ddd17ff18677756a87671d4145" +
"3aebf97e4890da1d645f41eb20da92a8537c787ce419580073c46aa3bb154952993142ec5b4fb6e
8f108fd15fc618cd5c27b45a37ee6dcd52a4ce656c0f58604" +
"717ec55f5e592355f1f20e8316f8fd77243734a8b0f50ad93c1d95b5b0482afb22cd0667d935bd605
3d7198b54974e10d100df7ca3ec2e0bb5ccce5807b266e0" +
"8429d5fec2ae6ae1cc7c5efc27f19c89d4b4a6c5c0b9397886dac635ba37446ff528b582457a4fe7f8
03f1a47903574f8982d4a679b627396a4e97aaa12fa179" +
"0d31ba52e9010bc3c26ace81f702f86649fe9eeda9ec03b74a8a5cf540d82e22af33ab893564397df
c4edd8b1677350df5b82ab61d24db95f58fd2d78afb49c7" +
"2d2b1fefa8ff6606b8623829cc752ea37d663b945f3f1d48ad07b1416af252f81b55acd8f164da4faa9
d9453721b3b795041ce7df7c77edc13865dbe04fee331" +
"47daebe18c183c4a6594a6df3a4d2dc5e3811d805102c9c49286e3d12b38927fa49a7b0cdcb1d79
9f57118953e31c560aae213a1799d59a78ae68f0590347061" +
"fc2668caf08f860452f6b7d3ebc1efecc2e1227d33296b1f1850360dee7236e85274eaede4d18a58b
4261ce1f6a7d283dcf64e6d021813f82a566354445327e5" +
"6217279b2393fe5aa0f9eb149d4866e1105106bcc221810ceaf053f2ec733d8a22f409c1baf955e50
184005c5d55de907de97f5f713b62ae10937e1a7af6267b" +
"d2a239e8589017197c343b81540bc26bc52bffd5336fb1da1202a511c7175014d2f500b9d9ce78e4
b9f2b158d0fb27af352b6f78c129cad642fe909612c9d658" +
"17a8d7f9195ee97201675a918e3cf520fdc19f92b7e6a3db806d4f3799361334082cc58a22ddb4e4f
5760bd1667c177b26be325166c6bbed669a158fc87acd43" +
"a2462e12578d72db6606f9e24ae659ff411ac9b31d696b8354fd08a591622967a14f8468eaaae390
7b7818154ba2d6e4581589354d178bb6ae1c03651c44bbf0" +
"e7fa52cb0da09508b5a444aed05a54f416841247a4fe36bd5529029e3adf78b105e22468ed775f4d
0954504dd55f2c9b9e6b3a086370b2c0b6fec7efd6914e07" +
```

"26627edb7a04869a874e31f448271077a7de3031cf81bdbc39848efee6075e0d65fa3a32640e9f03
95cf7ec12139992aff0a54e0a7dfe5048b3cc03246b56f7d" +
"3093538a7b87538d8792a665bc589373621b2f3cf47d2c1f8f580fe34d79c6b2a66323ce89808ce0
e5cf77700f5a4446c4be01a310e8f7c7ebefe756b0044886" +
"a0477c88ee8ea8c71503748a4cf9eb40ad5c1c8accf7c63c0f43a94ed2b8a5999df3ab9b11b80de7
3310e036ca88668e640015fcf9cd18eed05517d54896f43e" +
"25e7931b44872c4e4183500e0e8c5103292bca1c0d6b0b00c9acce25d31204bb3e4f255c03a0a09
16664e9c831b28b364078109a74411a11afb1e610c7d1c9d4" +
"ba5e10d0ee0da409654d9e7308395e17caeb9caebccb0192679866e6f2ecb5f10044333bb70d617
12adb6d74cdec6918ed9a71d9925da576a1e6f4e906a5cd5f" +
"0e94a25e48a4141e4e2770144b63e2449b0f84c82879f34d78440cc430196ba85a213fdac1bcf279
a46d7592fa29a876bb7a2efb7081365522a3f06fdceaedd3" +
"cc0335cef9ea570733fe8799bb1b918aa7732b4d175929d80c7844a78e19f2dc6a6febf648f49b403
20b0f7d784e7f84e45408d70b046bd01cbd8fdaf606fcd3" +
"02f4e5a48ab8d13e93a246adfcc94f3109e02a7a969986e75b6ced6bf2d11a55ab77488e131b65a0
6398fa8e384dc90d875584c9b17cdcf2da5dd72a461cd07c" +
"4a955c5fe48509b3284476c42247e086de7d63839b7358cf4ebd9edf9ac8b6fd0c096166405de19c
51e8785009d30feb67cdb8ff9ba55459dfdffba8c022e26c" +
"0ebd399e4b76ccb4d5491a862c2c4d8cdf1461a96c9b98150e170efacec980edc00a2c7f6d7c6bea
3075627e1eb386a7f1ede1059da81a4ac5cf35aa173c88c5" +
"1818dc0fbc688b68b82ddc225b6c87588e0c680e303e737c82a13e34be58df8b0cb336aeacc698c
79e7682ebb69e6cd6bdc5d11790c96afcfa9290f39515142f" +
"5f90b938216a1d14bc049ce3f0ac135722208b989d2557d3520c2186479f179e50fe5b125b8d6638
a65047729c6249b9b2c6381c9103c97d1b389cc9cdb31c21" +
"8a2eecbf4b9ad1dcfa57446cde88f96563a544c49d6f5303a84a1b7cf074fca78e67e72c9ffa0c542fb
646418c6434b16b771088140725cf2dc723c1a975c4ca" +
"8a80e633721274907353f51e95952c2b403b45750b42ad10961f60473eb54616f61f7b038c5b7eca
475d6a2b844994a9eeddce4f7bb49782e50ef78bc13b85d1" +
"9e956f47c60823f3d1981413cb78d309f63a844694861b11b5238961c71f61d82daef6795734f0961
e92b9167c57f48e91693e9656fcc6e88f9ce2d373da26bf" +
"45b3dff50211fec72387005a7e04828e4ae7ddd10fc2332acf5f1b0f67adcd863752573c2d2448885
7bfc58c41af45be7641f5cfff611f184612fc0d695866f4" +
"2b396b1d9881f442c4a995f4b500f02d4ab4b53ad6e01776ab0e244583f01301203a1515f3dbb739
06014e36c7143bf882b005f0228ca0562623893c8a24b7c6" +
"4c2c561912010c121b5c3a1e35e75c0b094731e9c0d6acf5a2b1e5b179355525a175640579705f8
98feb98bffa25633bc126613fa27d2ceb214812902ada23f4" +
"367a78655d0d2276095c9e83dfa79153730103963499c367c5621fecfd0888253df82b3d5716703e
f92594cf269310b9e6c892c488edb3bba1d0b216e92f622a" +
"7f8f7f00d2926d81a4c7ca6cef40d240576a8d5541ccf561c8e0e699925d20347ba7493ed6e182cfe
3b633e70b3ce3a0d90813574f6fe329c495d3cd46fd5d7e" +
"bdde58d7eafcb134a9a5d3e5d66136e8c9b5d9ecac195dcc44158941c9fe2d87db52a7ddcedc9f82
ec160901cc36a9c877af80ceae0563dfa75cabde5d7a7c94" +

"9f24bc190f7c2045368356474ff6eee284e7125d1c5f9a036fbde24cecfd3a30481ce077f20cbcb319
24368296abf66ce4834102cf7cb949d1b4c6faa6d006ef" +
"21379cead5d5a39324d41555c46e0b42a1e871143e47f8e6b3d794e75d7a43c282732766d856e0
4e666ea346657b157404b0fc8534a2dee8243d40a5e37609e6" +
"18bc1d52b91a7623aaf8214a97e4c8c5d860b31c3792b129354a121a7a7e42b50dfbe3ab6590769
401eb280545547a43c3a1455355d5d5fdedccb472abfe75b8" +
"f5e7d62b0b31553d8d55de0c3c71e6f5a2abba6fe81e9a42ec1968f235bc4296c1ac5df743091745
3384450ab56dafa7c7af764cefa3b0bc861c52ae27692365" +
"9d7d9ed7609958884147acca867909a75bb6a2c364debefaf98c7ff70c7f4acb5cdb81100fd79a48c
5139f8bbdc6553b509f1eb0f5d5d31886a602cd669b3f9f" +
"59195a1fa2bcff1170003ba1b2e5e9ad7f2bfcd0573d0f2be9d8fc1773c3a63a2b9292cdbf9b4515c0
b1d51772e5ee95303ff493d85314c989e269df4ec3a916" +
"40988a11c6a4ad96f7d0541a150edf444c2b1672aa6d37564453b835c2d39864c05c4366492fc91
64bf73795410e7aae8206430403357fec6389142b4976b218" +
"d70622b4098e322f73020a0d045f07668d1e512c6eeed6e2befbfc3a6ac64054396df96fd41f7aeefa
0ab1f66bb52ee1a1df066f365fc43ff0800b0398b621f9" +
"a415895268505a81517c44a56dc94e76580fd107dba034bab9f4f4b8a9f881ff34c60c406c47b6d4a
998894401006aa88f328393f9cd55a2b4d24db5abbcb05e" +
"20d392f3feab3ca12dac475eb3690f2bf9c699d7d90900d9a840068c8cdda2ca7a27bebd685a26eb
01a768259a65ab4d7efc1811c87a5a1f4e5038f6b3dc74a6" +
"b46d9ac58d31bfc22dac23645aeef819329c9419326f22e1c24c53457baf62ae9b92ab5f999d4ef0c
cfb5a21b7598340eb2d399ec81588b6a674c5a1e45aa238" +
"c55cae8e1af0f5d64ea378b8afeab263a3a2e5c71cdda4cdb824ae55df2b0260aadf386275ef57781
d46f6da3d0b300ea68c14a620c25b5df738c54aef04d63b" +
"7dee06cd225e9ed00e78abcddca5a133d8b5e0d9b287e6436014c5da729442239bddb7ecd3fe34
e6f6e530134a03ef45de4ae4fe3bf507f16cdfb9bab1fc90e8" +
"dc565e4a7ead95352c5a894661e5d82c6d0fc47843d5cab12c4013db76c90734cbff34c73d0d873a
c9b27b417665f4948469865f33179624860604a9aba2ceb1" +
"68e74b6af3d1ad0bfcac4180ea844339a034b6b2c3e2f61f0c7afbaa76c1ebe93727df1d3db27d59a
5cf51b2baaf637b6eb8a20302ef9af0b25dbe3a5e74331c" +
"6b0c8a0cf2a2ad72d2e19797983e09468ea95270dc229f2fa084dd2aa96e722016504f6d82508572
d9c30711c3ef41ae3ae2f36cc6f5dddbcb0b40d9499b24c5" +
"4cd36d2927a6b9d57e335e4fca7f0f16887711a8c1ffa0b48bda46c506ca444b7c23e2c8dd086c2a8
7283d5fc0d58e9a384106837318dc84ffe65b52d4cb9141" +
"2672adfe139c3327350fe3cf355a08c0ca43598a253833e114243c5253077d65643323f5d69b3c79
02d91bab7a0928754e7d80afab8d48539fcbe0d9ab83b4db" +
"43a6594c4071df2ef35acd1f53006a570f09104f1776b26a303e2aec93a00d2fd8c952d1ca0e54504
cd9b469be7c1e71557ec31467ecc773ee817b17c4418712" +
"163ae86646b20b80c85860e828c48e88f1309c9ff018e6a95f4c1178de6a4f9f5860039511845da7d
8727b5d824ba2502d0a3d76ce74372db77c2050c728dd65" +
"b3a15da4f1e1e41c3c2acfebc5618e5e923d503c43a3421d2628ac037c5ce13c74c4ee14d47af023
23872f6bf2e8bf09d017ea6e8ec4d3f9fc4fb203ac4e1663" +

"756b11629224c676713a42b1f43dfd6362876be1c4865928688765589e26c8dd8bc04ca18d76ced
7f786cdb0fa5028ae53991d5b7b45f93bbd50aeb97300f04e" +
"69c6736f270907f6a7ad76dde0a365183a961bc8385511e0f22ce0cb8f3c42c5d3928621841e3028
5fb625294865409267dbb0cf91730ba2fb1088fb79789a54" +
"a856311bdca5b0ac0e95fbc79b11c561dc03ea82db182808031e86ec327097143ee761bb62dae8
a9f4101fabcac1fc87b3c2080820582dc8a7a8287364550013" +
"08053c781b3eb279c89e817fe97103b6930fef2dbf7728def389403a4283f63ec04ae953784b749f0
ea6f08749781cd17fadfd15bb197afd2f4e0a8aade2b1ad" +
"5100cbdce49ed59658993c00e06bf57c0026b97beadc30cd25f586ff03ab40fcd731535c9a1ccb2c9
9dc7f8815feab767e1237cb069981f28d8fe26bdec24218" +
"488e6086c0ab0efc5d4211fa0726b3a11387df9bb62b863a7b154ca390a268f5e49f50dec45d24be
ce2a06575cc07a24bfff017d7445024739efb050ace5f345" +
"98dacda843d4ef5bfb2c931dc16ee3dd8b61a6f01d9a7de8bbb6d89ca8695f8ef8bd1cc6e0455848f
ac7691e6789218790270aef40fba114557fd88ff74fe8fc" +
"476d9b9665d7e45582540710ce92c8dcad1ad8c05642a23a0d58c02db37ae1a0e70fbc5f71b1300
fe398c74cbad37fd57379f58dd3e2d3de6860a17acf3c9321" +
"02eb4f9d596497bd849c5bfaf59a83113ef389b6896aa4d4665504a22486299993a9987b2bbdb47
d59b3f6ce5d2c9f9ba33b5f0760388ca7f8d8af07c1cd28f5" +
"67a417a59ebde4bb9867d4e7b7b79dd8665602c029e9a16a7718efde3d034f13f7f0b9af1702c335
893526cb87afc2100e874b25c37fd666bf34bf6a653c7cf5" +
"44e1fe0286a6723c7d33461dea380b392dad68f79a78fe1b785d7833ca0d1cd68cff472991a625e3
099f3ad2cdc99bd37eae35353cecf424098389dbaf1885fd" +
"7db54909a92ee879609eb2e9ef4de1f4338f0df53dbde486ede944ae69869fac701d4f1f48c83757b
470ea28c9de2ae5f1ef5d1c91118d16ca0d80b1baf3d314" +
"056949df27a09eff70c9ac50b54feff67a165ce5e22ba2222defedc7c39e02356c3553e97524c1506
441527da4f5de121142ccd494f83114b3ca2dc37e15c752" +
"e2faed7d50254124d68f67e26f4f50c9f0edf6e58b916ca830c4e33801dc11039b18292b87b08f4f2e
dbaaacddcdab78ff3a0004f86034080f2ca4394b14aed4" +
"31e38e3605e6b257bd772954d2f4b846a17df7ed6e5dafa33964d9e56a07a19898fb4dfe8b2ddbd1
1fa0013e6ebc0e429a5166a43d1ec45557cd1fc1bddbec4b" +
"2e9ca26395394c96395ff8f557bd0f7f805c09f0c18534585b7c7fc1d07f145372983ad77fa804fbb77
65934e72beae0929a87cc6bf7f6c242ff5db2d4d5541c" +
"8c366d22e24e1da5379836fc0eb484683285f99f178b98ca170464bdff60ee04584c12c65408102a
c6dc7d10bf58a7d770bf1b3c636a48f934f6f4bbdbcc75d3" +
"fc551de3ebaf77006707f6120b3804f2bef9b4bd59f5996610c09ba3953994d1b78a9f3bc3bafeb522
66f10755ea842e5b4370c937c09afd34a092ff9b98b4d3" +
"518bc2480d4b132455b7f03774ad76b83b254742117921c31cebeab5f39c145f7f373a5603d17dd9
5217ba1af37a0aa95b2992efcd02d0bb4ad08ebafb31440f" +
"1ccfce45882b547ee4bf6ec7ecae11ed79fc63b03636c8a14ec4e0f6877eb658d839be2eac0f10a89
48e74203f46078ce66aad2764ff05590e2ac7a8dd8b3036" +
"901fcb7ff3369ee989a28e34b9b62e1e607d14da3049ded1a4ee50257195eaaef995bed79ec8511
1abb522aba1fb306869a1ab381e82943f35345bb5502bb90a" +

"e2a0af77526a84754ee4d600ba7f8ac98705ee687bab949a081849889d7b83a21a3dd34af84dc2b
9458230ef0ff44c6398d3c6e48e5c09c399ac4d4c7b285549" +
"e0bcab7fd96de42f072f1cb633e3e250745321049d0d7ecdef4636e70e94c8414e76ecaedd6ee079
2e97de11e7dc1e1e1801ad68f9147278e268d7ad76c5bbb7" +
"98386fdc13ca8c77569d96e0debba8ea3b751352136c8f1c8d611a69f1baa9aa4b9d0a476ebd5dd
21339ef7f97f09aa86b69a7b114cebe17a6b0e58bf52803d6" +
"fd47d9eac3a988b51e9bca95c546d49367a3126bf8ee44fbd0e77611473a1d3d2de0ce4ea54f9bb7
f9dd0d0c065f613a623fad43a445eba294fd00037492914f" +
"b74d10d0b97a0cf9bd3151c3cade89521f36b6fe1aca7f352e79a77d063da5337a7c88d90e9e566b
cd97732baa4459305967c2f65adf1a4a4c7991cbc99df3b7" +
"14335a107a97a4ab104bc94fecd1d003fe6d2f22e717853c449881c4ccaa7e7a1e44961a14a47a0
d0aa1b1493dd02760ff4d31fbddf5941f93c8e5925d1886e2" +
"8761baef8610fa6be016c8f4fe65bb0f335152d5e94893e274f2ab90118e4c07957d252963755b4b
638ffc0a734fbe6e32c2e304b10a46a4eed330d101c4f0ae" +
"011e7f94b89bc0eb9d358a6548b3f0c47ccc3c2d986d381437c49041629c6cbf61bdf0825efe17e4a
bef128003681450ceeff0e28842895d8e338c247abf81cb" +
"7260fd45042c1f6c630a4b195579721392e577fbfdb9f5b003a8b9a6bc15ae754f6255131a0be600c
7b07e2cee1ffe32aad4687f9a429998ed9059a99fd879ea" +
"c4dcb55f4551bbb70c187cf1b162e2ca4a929edd6ec9260877df652622ae073fc63c0d8522d3882b
a888ac50a67a68fb6530193f93165093a1d8132e87d8887e" +
"ff2fdab0fbae6ab9506dae61fada4023133d166bcf1956aedc3237c77d1c81dcc84ae957d89367b0f
c950c78e58f2cb9c4fd93e16b94421fdecd46c3ff55592e" +
"4374a7f7d8ede9923115770cb416071e8f102d4ad78b891464ffd14f589c238c8e13a4e2a81744d1
79e7d3ae36cffee75ceb99633face85d077d0c15b3970930" +
"075dc08b420e0a545200895207c5a746a18ce9ab64a50d3dcf44da857fb65e4efc29b2b4d532dc6
a03b699dcfd77030a4945e6431273e25f06ad8f913c2a9eb7" +
"59d8d3049868d337e451726d95c4cf8baf381096fc9b62679175dc8f14e52f8b99f212cab6544414c
62f17c8323256cce95356dcd351e34c7a1576b17c1406d7" +
"5b8bcca8099a1993df1541ded61b876ae83396b191b719c4b1cbe50d73fc13da352d827ba09aa7f
dfef3e4e0273c31ef4fd38b93cf64199c3969a7c09dd5e0f3" +
"ff93a5a7db9c2c1ec25e3060bcb5481c6802e1eca78f31862842ea08e8f92b8e52856c4c9fedd0bf2
0e386cfdf926425f7756ff41fd3567c5bf334e96e3f492f" +
"74bd0519d8d98efa0b427ba681b8b1be8fab041ff084dc5f8c4d5d48f481115d7e407ad8a6034f481
c2be86f8451980c3aa83a3fff245d90d13801a54527e97b" +
"e392b25867882d43e3819f4a8aa380db63954ec23d2f0c11a7aa5bc7a3aedc43ecd3b024280ed88
43399e28deb954bfc11a3197fb14a9c9a895859e390e9586e" +
"2ad21da39bb9ba79a62222d228a0fc96a24e801f00afc3f98d2168a8a253f24deffe461f6313de9b4
33e1d2e307239c0e3fd5d9fe4c8352c2c6797b1737e93fc" +
"14d411bc69bbc9d78cf91734052b8aa1dab348e4c243b8e6d623865c135f807de8d5fd88f3921327
affa37066dd538351bc4ec52eece88856de0a424a87d062a" +
"f68cf24db37dbaa8e8e96a812fbf32ccafdf1b9d27f11bea23df02143bd09061a881c010819a315a5b
6ee44b3c60979b3f7b41f488b2429d49377d6542fb0e22" +

"d09a0ef5b81aa7c8134c0aecdc7a4f9228559d0bb826d30fd77fe0f834212647ce61e22fef0a1c10e
b4177de81c31c12054a15f81b605619f3045646110673d0" +
"b2d79d80577fa43284266fd2ed54f9a3b9df3509f79559c5bc51a58521bfeb2f95d8851527b7ea47b
92a694f6ea2b67dc2d4f506d11d2db32c2929cdf5c8816b" +
"7f0c310cceb7ede08d5965ed2c7be6c0a317251c7d31cc4a15f6d7976a8a1e6a2f386fe0071d43a5
0bd0ce5e864a8e449fe9600c6e4a84866879c490de9f9d46" +
"3f22708abf34d3e180dbb6005484a6afad373838cdac335f05c034e3090b2fbaaa53fa2db1f96bbe1
41d570f17363ff98672500e16994b79be74634755b09e66" +
"f1b37e338c946bf85e06c97e31dbddf257d58fd10468278648d86f38710c2ca0b6ea7cac4ea0e2c4
9b96bf1998bde1b3d38aa853736308e12b4a0d467fdb8a73" +
"0d810ce45518614bd5845f58a9835a5cfbe745f45ef59ce9a677d10d8c9f6294f1a0565301efb3c66
10afda35167150bd326c77057e530c213da63af3e6a600a" +
"d87b16ec5cbf76a13764f71b3e7e0c867086ebd9fad02e1d747030064e071a13da4758cd0fa20872
b3dc350f4cbfcde1b68a97aca41e32207b40beddec412c0e" +
"c75d87c6671ed94bda5170aa2866509161c28d550190675f60139a7b460469f3d4829b3c65f5d18
5936582629160522fcfdcc53fd0dcc8fc46de11d52bfcc5e6" +
"3407ecbbb682cc1693d6543756fa4e068e92ae1a94924a1ff6891361e5f262b7d3c3a3bc2866f54e
6d03ebd5479afa3f424077d51668cc60e23b35fb0222ae22" +
"5223ba8a8c416b68c8853022d150c951f06f8f85c2078d3035b8ac3ae984ffcfb024431acaae8bfbe
b981870f9ad6bbb88d7d5ff34ba21a44cbffd0aeaa435ba" +
"7d40d22602e807ac9a69db514ab13248133142cf03fac999a2b185f34d83fdb495ef042d4a5e92f26
24193c88858d91c0812b18fd67046cf50635e6ab1ea9ade" +
"7b1fe783dc5147f14f9194cfa92c03a0456f4171f9e5c156fee1c607a1e9e06535f2dac49b92ddf5fda
cbf88a062bd7ca5439bae645100121e598deee6043baa" +
"85cc0d727f08d37a766a55a9ca21ffb6594fb73f9aad15be4a64bafddb6c85d00f7bb5705d9e56b41
0dd80df8b087b6d67c7ca84eff2ad699f901415fab21343" +
"6351a9bdf83b440e29f3950c7e4c49963ab109686d78fce629e9207db2e17eb5f02f01db6441002d
72c06c6c6bbcdc0a7443589ba29909a5a78864ad51e1dfda" +
"14782d869e4989ac3c5ef0aa1eabe540e9e7cd4e8eabe25b07f300a134a92718186f085d5c10a71
1ed0e574bf7550f6bccfc3c094d6e59619bde9fd892af8ef2" +
"50e1cc3afdcd9c84ccb97344542843028b00064b0c3d18ac0f0703fe6f9683d40813abbb883e164c
5797bc1555338566cf8cdd358e9fcb0e93f08f7ae06a5121" +
"c67a231106ad8fd42f0798d7185c2de78b8b76c10e82272a405212ce3b904f90236eeea02054953
b967cb614e8f8ac49b977152a52df981c86fa4a92f7f70eb6" +
"cd4eb65986564039b0d77f8bafedb4fcbf9c34b8fe9c5fa87b0785c118a8624498fb0184a0dbbfb167
77579e1964330c12e494449f6aa5cf69ec4a32054be553" +
"6027e0d27c7044abd4c0b8e43db703209037efcfd08944647a90a1ab0c71011753354990cac5a47
2fae44dc370aac8131ebdf31456a8484e7fdefd268cbf5cb5" +
"85ac615039d3655b1348fc0b3b078ac41cbcaf6aaedcc1153bb8d55c307f45405ad6a959abb37bf8
891c8dec79a9d7ccd9b791cb60361d4a28f33ec0dfd13fa8" +
"e0b9b29e14bf36f5047e51a39c2efcefcc156bd08e46c5c1000a3cdc2bb20713e19d6f492c40e51eb
93628cf85d07041ae5353e7decc824cbb1db8ab3a7a7fca" +

"ff04c2af423bcfb1864ddc864624b827ddcff2a2f8fdb7a3d86d76e72b4f850ec1262d8fc89e7b12e4c
c618afe6a2bdf205075c2008f93b7281d80180199409c" +
"de850d1f14ca0ff960f69772385cf0f0a0f47cafd5489ea4fd8b68ec7aa539b942379139756c95bb90
818842cd43511edbb7577ae469f46728b13a61e6eede06" +
"3a4cdfab5ed590feb807d55d76e518d1d74bfa6704f7c8ccc672824b4d5ef5fa5b3ab8fdf2b6c17534
04ba35b76aaa931a4e0e5ca7e440524166b23e9a8be9e8" +
"635381f6c9086802d428fece81395dada6b3b866e905ec00ccc4fb9b8415dd15e443f84b7220e3b2
8700ce3d88f9c6df2afea39e0ead537a50ee11f0c247ee86" +
"d4b3074e8761de4de611c409c6d4c369c2c11742a7763f6550edfaae49afeec33353a14d2ae6068
7dbeefd2fe29689da6ae79d7b06042dfd25a68bde9182fba4" +
"1ac53706a8b96535057fc2f99ac84a9cfc6549920c3e2cab44e48a08e77207b6a95b2f6179d6dfd6c
2d9e3c91106a7a687e40bb2a1c5ccf566c0e31a0fdbd0a4" +
"f270f9812208f939efd9698a8b28ce9c5633f18ace7ab0a7550d9e7e26cf62eef49200331e19a64be
d648b5d18ceb389bafbcb3f280ba78e4cf03b053f2a5f08" +
"3c852452837138004073cf6726143179386279f1a8f15d44876c19bf6c2e2992ce6056191bb1a386
f0e1f6f249495cff126991c6560e3f613e56525c0c49b5cc" +
"2ea4e736d83480f2b45d7dc840b849887f54a2aa072e72e3fd0db34e5cddb02221fdf2a40fb6ec271
ba3a09de8dc73c24328c5d9a33ae0adc9874902f25d5bef" +
"4d85914557e2983c93fba16cdd4bd929e878b5d51b142b6e9aa0ce84871b7b03ee6cc13251e175
47c2d20a7d4e948760e207e29de58a7ccb71b87f99d79837db" +
"d0f293ad3d33ffe91435598e8a4584b7b7ef5b1a895a2827b4976f81d335e4aa6feda35396908996
19a4cb34fdcbbecf1b8b38cec2ec7c07ce84ec3044f49656" +
"28fdba8971585afb509526640d36425777b6ddf5b2a49d795fdcf71e57fd35f29fff37890541b6e152f
14fb6ea4c70a1b9f159d02ed895a68dcc276f5d5ae83e" +
"47c021392ee22a398c8c73b3446d61562b3ec596036959aa645a65e5d24f733e142ec0e184b72a
2adcbe3913932b2c9503c856a7e989d24f306e01e99268188d" +
"f858694e297803effeb8e28bf8fb63ed6787acc2c61f509e19099607512d40928a08e649474a43728
b63523175fad12ad088aade0c1e20815c7c12773bc959e8" +
"640ee23eef2b1653ae8918615b45158a01be5a5f39a75a7c6cd8f1f6b463516539771ad251d5c2d4
0c5049877765512c44e58bd3b9ac3a0ac281771097880fe2" +
"c9516dcd6f1373e1e8a52fc485d104004dcc839fe3d120f1432b213388dd37980ce8238c87a70d5a
be95d78d696d2436eb23a8f620ce74335d5e47f6524b11c3" +
"e22288644b539e3ab664dd5fd6bafb02897aab35adaef204f82d9318b22f45b787f5bacd74b01d23
537973060868a47f2e3a45c1d8805a1d657f2332af8170e2" +
"9435d7540e70e92a8c8794bf22d3e11d54ff2d48cbc7a1ac3cecfc48f80fe521f6852f97aafa0605f3e
7084b15e61a74869512c9c2d84180686ea07b562cf35b" +
"5a0ca529481ddbdba9c60729f821dc7a5a8b5c7eaef1ea7927d455a702aab538e7441933c4fff2d2
7de5659d6fa41f0ee72bb13a829839267f3a7b51a81a85b0" +
"d737194d94e1bf8173248cb057cee19eb5e2cdda38c529298f3c4d3b95400198063c5b27e9262f9
c66425c65568a09035bed9cd55c1f2ec4becb6b9c59445398" +
"ad5b7c85142e713b6dd32493dcb817c8bcdbd728e325c25c5a14d764b63f960d1e48a0bc7f4d2bf
51060f83b1d1f2591c6a9b79182e686b887a2c1461442e2f9" +

"16e8582e298f87ca95a8052df33af20ebded7bb1c528920233d1aca3b3789494d97084890fa3db0e
a7eb561b0087c4a90000db41ea072613f91ebba82790f33c" +
"fd52cdd92d2ef1246911ef1dd82ad083881b72a08a40ee55884380dd136a7c0724cded69c6abf1f1
56b14ecd7284abcbf66522264145ee78ab0ef0d2a74eb390" +
"10946d5efefb7175164e96621d3f158de8b57956b8b1155c35b32007e47d915cb61dabd556a3705
37737574741fcf9a8a23f7155bf1f0e3d3c0d2088d1191d9c" +
"9c974139303f3dda55a70ab4810fddca3561114969d370f4e6bad60a53815eab1c4613854d04ba8
b049dd7ab1a935c728299d1502ff9aa3fbb356f87f2a52b6e" +
"947dc79b5fd211ed31dee722d3fd857f43aad973fbfacb7cbfe1b2553bdc76142ccae5b4021a4647b
8d8087925dd3191a57198792b6f918de87a92705ce57905" +
"f2dcfb67a20f8c77e700933432d60a4536d0959415f15f3eb8a788f1b19c497d3b68194e27ee73623
1835469d8bf0ce1717ecf533ab77dd97b35881d8eda959f" +
"54a7935b1bc11d7f2e472757734afaf0463da3fad9804eb948e8d6444e8394b33f1c187618c7c023
71ee6d378ebb7a20b6049a5504daa71999d15944ee82650a" +
"2388f374f3ec3afd4ca58ef3f2588997d194a2741252cf6562e00cd6b5c5fe4066454d2b315031769
4052b4dafb40c2f04c850e4062cd8f0af2da75280046850" +
"77990788b27fa457ae9d0b622d18fc070f1d2661ecab529b5cb82f30a29610dc6a9e93ca9a2617ab
0109957a45c1204e5eedb8860c6f4d57122060f39a4194fc" +
"a285f1e9e7a75cc3511b8cb4865719c2260a630845051876e7795bba59573b6ce5faf7e5708eda7
be25dd49c8cace4c04c541074d703e6601e043f6c63a0a371" +
"1a381f0ff83d136f4aa29de266169ce5b3105cbfeffba370fa306a93830e3c0519a495b8b9f4b72078
e2c45421b4b0667f903676a1339c70ddd1a90dbd21853b" +
"2826ac3fa5add5073c634d4c5e87db0efe18638ee93c460257e52aacb8600ff36739818056110b2e
974a1959e3784903aa97b0fcd9264f7d8f6bb5d8b7d9f03c" +
"4b643955bf7966250936d4e7d651712db5e695a6a36b5e6f56c651ff737042b5bb73638e21ca6ce9
a3e63fbb1906675d97001d7ee240d277d62df18acb169677" +
"963d231c5276bdf5767ec35fbedb062e61c23d759aefd287b2dd62a0d6f0518d90b3c1756fde50afd
33cab395ddf3cd538b9ad8862a199141331c63110c9ddaf" +
"fa3d6c63a1fb1b45529eace826cc29a1df5df327bb782e573c41864c18e6d31401d19719326e5c35
bb50de7fdc67177a6a6015b4264fecba2360ab72ae8b060a" +
"6c66c5a05782a15fe3c1833b47e3495d29f2cfa579fcb08f02fd064e9ef2ef5564ac6a43cfbcae7d79e
9f87ebc2176611823c6624db11892f8c47f8c96a49539" +
"1c18f821ecdefb343eae3fd98dae1ef96fa3527788543c0d06d9793579cc62d91dc4d25312901c63
68ba81c8536c6287230e8f97d25f6c77366609580cf26a27" +
"88502a9aada84a794d3674ae11cd1742cf245e9d9502dbb5b340c2a6c79e3607f6b47666e1ea991
ccfbdf6cc41ede46d043bc4d3e5e6882414dc65d62f9f47b9" +
"fb7b828a89afd6361ae458c2cdc82f459c54977072702ee5a4c22955b8019d8b8d91f558897c4b66
1f8e5412ccdc10c40521303c0ffd353a0c04cebca5622a71" +
"192b144d0f9c5c0706a130df887526b7b6e0f358ad9f7d0fd4d87c5fdb29a7453388c0d009da0d4c4
7a5d6cf8363892ac42b6ce3388771f698802b4dbfd66aa3" +
"5fa6a6f8b42dd8446324501c807b6e72cdd35cfe08956a52f86bb4709fe2980f62152dba3571f18fcc
4c1cf7a25384c4b5174e93e5afc9b9f12db2bd505ddade" +

"d670d0d71b9548f9a07ef98521961cd96e8f363cf3222336bc4baa284b5305aab47dace615c1b3f3f
b1ee23ad9ca3f58b086d9169ee5b2d3c2831e1db4f905da" +
"11e1fe79e3d48c01bd9879ed68391e4d24d6db8d6774cb8747e7ea368aba3bbf355386408af4a59
b23fce74a5e673a1044db66ed8529a65462269480736cdaa5" +
"0784fbd77e1c41197335b4c517af8a67eef5b7165c5fd6022cceed0396089c3985c36595497db0a0
fcae478e4e4d68c57b93f466aae86dd4244633beaa8116a0" +
"de25d2a54353b7ee85fee58ad4780a2957d69816585a64f65e75f332614aa6786d1a1432f6acde3
85d3d6e870bc968c60c81401726a958f0caae336c83a9523a" +
"c174faed43ec67473dcd151506e334a6aaf1731dd3aaa831f934be83beaefafa11810e7eb140f4fe8
0cfba574e6106c1bfe9f0b20173a4ec2663ce0580df6daa" +
"7966a3a8906677ab680025782c61b95cec6a73b5deb16599e6521f9c6c4cae0d9286566388d518
1d6ba11c51a25c62b510d9b1793f3ce9f73ff0c9226c8aae69" +
"5d014287df074a244014720ee38e3968557db00aa63dab71854b8573c42c65116e3d88bf040d53
ef3165a5827c717179e2939e310be5eaf6fb75447ba98ce925" +
"98e83a32a90eea848500a30eaaaceb307d37b1201b83a744468a1a52632ce5525c1fce5f702421e
42e7cc4c61caed539dc09001cd31a8a2b48a783c36c56a3a2" +
"d50de42c63981c86642cc92bcceeec8a66b4afad3c1be1df4bcb8beedd442c281080c94692bf4531
96ed1a66a074d56a8e7f60238ce18358373efc173e70c691" +
"f832e1139bc04e6258d77cf7529af7ce5eca28ca5cda818625c0bb5beca96d99fc9b6689a7771434
aa96e23c55a41cff7b7b718df58260b3bc91762034debf49" +
"7d8ca8d5764c52bc9665bf86db5407ee1b786d90f8d7772597eceb98f0121e3996e771d951568a1
62f6b71042998db8208ece5b8b0c68107b8e2079765b0d8c3" +
"2747597072756208b0d84415a5334a88d916bda390e26ccf3046b860e7ccbe22c48cd3d3f51bb65
a98ace74d52613f782db726babd02780b8d620655bf9d551c" +
"ae9ef3056e3d24f5e7c3105c4857492fedd244ac2b8c30a874c1446630b042d819bc6b6d2d96829
de903db22af706e93c5ae876d72c633600222443d1765bc62" +
"a8a20c458ae55bce8cbbef753cccc5e7d929408d6a3709467373651f0163128aba4142ecc56ef11ff
1fabf5eaf6e955b4252d1350e9002300a1236ab2fa0ed34" +
"c9cc7dc1d4f09bd31296cec1493e725b57cc496fdac4e8d26197376bda7f74c0965c4352bc9d5c73
1df04f9908899cce6ec3afe15210d115992b2d95308dd032" +
"13c557ac527424c7db02475a2fc78b88d022d212c3d02d5ee490e2436e6e572e8a1330465b9052f
8a3de01aab76662d18fa3d076fb77103fe432d549bc861fcf" +
"f63f3401cda31673ee48826b68b387802fea4471deb1fc928586f1b1614c16311c9820b563ab0112
c28af5c1af5121818540c4b7d7f549b33906c1b86c6674ad" +
"799acee7342e4a79d9295493b2430fd08f373338795764621bca444868f3f42b0e40abd4b8e148ca
d2861fb4980b83bb58d40eeecd8d8cb1ef1ece17b0ab72e5" +
"06c6e650a3a43081f545acbac51ed7e121df51edb75120cce30ef7dbf41fad331120e537fb35be45d
93de4fac0cadc7e5f644e2b767a285facd5f12845559785" +
"57f4afc276e21d77f6162062430dc8918435f035f435ea419ae9f1ddb6afd46b243f8bd6a3a33e797
0e7e76fab9ba6afa72a4806189462f9d0f231a23e3ee1cc" +
"51cd10cb9043a27deecaca866751f971254fbe3084c243ef5f516bb652988b770896ae5abfa12db2
eb2abe404cf694e9f60d47e734e260ae668b750e11b26001" +

"0d2bee5ca555a44523742fb069e484f7a69c12d4bad026c03ed7af10ebc9cf2f54d143fbe4de83448df80668217a11f5a1187f35ff306e6c685cfc2417c14aa7" +
"aeba1fb7dab05c913fbcbb8e677dd0f89324048862220ab6f5340c38b70804f625f5a526d6675a49fdc22ea6ceed477097fc723a7b6eaffd65c48dbee13df566" +
"f8f3449d91abb367cf37a8460fc8072c4ac75f88be8b9c840ef438cbf12a2e7d55799f641316e3381f72265425f3e90fbeaa9919533d8f9262da27f1f933d4f9" +
"a83e07aeb968016fed89e7b16babf0b6af3800a27c9c3d330b6bf8be447d31bedcc526b1bb53ecb10c3ea098bfa7d014d93274bec70b6e82bd5c443e860835f0" +
"ae82b7be7c78cd996e0990e3cac8c1c431481c8159ae1dbc40c03f4ac543e5758f347e12715822d86c881030de83a76ba1c49e4d4836bab7b5287122ccf523d2" +
"33935d802d2bca303cf57b36a5ff17e7c611f1cf99699881ae464da2911d77580587a7228db8325f204adb14413a13fe318e995d60e35c88bb47b99ba9ee8daa" +
"3e40ce5818876a3911107a159125dcf768ba04074e5771334e0de430c439070422508577e474e9532f7dfbc489d0c87d37103920415b6c116a422ac15e0736a8" +
"1e1e317adc87005f868815950882fc7497794c5eaf76f9def434d198304ff495bd2f9f4026aea330450741fb969700b953ab265aabf1fe146d861ba2aedc53d4" +
"f929abec2dee710aed8fa605fbb9bba914eaff01fdc113836d34d855383e4a311b4ec6ef6e80dfe32bc8035d84ddc4e2c305c112b93560112c1f3dff800d6043" +
"7eab01991f924075b4dea4db01c377ee1ca374d383ff1fbb463bf7078f6cc7509a0ecf536871abe7c95bf89f29c71f72f1a2002854113cb0d6d2192c00123010" +
"8dc9477808a218f84afb81f0274718c024393d5be66edaac7406e520b0c8e2c02ab98ee7b290db261f2122ea68bd79f2cc6dc64936af5064cce2b4d1b7078703" +
"951b6b81b9b60b99da4c2d12bbb50351a5b7713541db0958740910ff69e748c71bc7470a3c05489febefd384e06d267371935f652736bbcaacb20c34bd50144c" +
"71923b5a521ac4b1ba694d024ba51b4bef3ffcff74d5dc63810b2c0f529073e13ec3232d8647ad124b21ff73402d371c0db39d46cf4d2d4cf7ad43fd8dd365f6" +
"9b6b7bcdf664df0e62ba58f3ca0c62ad6fdcc9b091fb4926cb47b5ff8de7d3b12bd8709a46e5c3d5f0d22934c7a0574ee70b87af97d0fa46f7d9673915fed1d5" +
"a6c57197524ec9978d1bdf65633721ea2ccc25626dcb5e7f5e090b00e413c10a6d20b45fb8e98c22928de6dda184e856c86792c7cd09d38e4333a76882d363f1" +
"7f4d773ba104b2d04fd81027da087258fb175bfa8005c035a4719bac5b9630ae57889fb3b52a0fd47ec4060137b0f95fa5d5684172d07ca91e91eaf20dbfdea8" +
"a3e23937f33d8774f30c7e8e5d4b2d5371e5ea5e8d290970904c4c1ff33baf675ed79599653808f652ec4fd0088877f7dd7973023ccc8377d1ada2b80c07d077" +
"d7208686354f511925a3514c9e93c13525353b3d9528ab678e3e783c290ead88c2c3d6230bd4cb3bf79fce6dc3e95bfebda41e5d994e61ab083d73408ff6b627" +
"6996a263d2920170fff6869c2311441837a2fc190bee104328591b402defa38b421b972b01d020bd20b1b6a6ae884b23eb829fdf032a81d4f199a87ef125d4cc" +
"8662e24deb93700980e6ebc6882bcbaaa0283492e81f81e76bbe2ce18df4fb6654363106589l8ee217b5da262f1a1adbd59eb3c555cfebb12280058c75b5b33f" +
"8aa8c2d7cebf12ce46c5f49ecec5a865a9f0b65476793884f0021f8731b1bd288f55dfa1665776b2aee1007bcaa6d92a76a2ba9925bcfa68db7cc727b2a07ebc" +

"e24c0314c96ee4d6164c699e585461388dd73476a1e0519d92f51b64eb2842a7b17bb55d512d52da802df63206ee926f6a6a8c32de7b30e7cd3f23e37e0fd82a" +
"556323736ecd9de77494a2f8702463f40fb837c2a99270b9050b0cbbc2c305a32380ff5fa94bf9c101c667f36293c12ff9aaf6e0a810b75230caf915135cbe6e" +
"63ffb2a0e8632d32f72a65aa965fc556e10ddf6d5e40be919066eebda09d581a32156e1675300f52c8b355e88696fc2a67dd8e350a6e902e082af28a9809ba11" +
"ae0a5fd9c6627fb808d757147e5d59cffd9c45874478ab226e72909ccba6592a54391d072c7eb0221f1ff7be9924b9d037e4f8c31e94fdc814a8c4cc7ad4c9f6" +
"eacd5af66dd76bb6222b2fd3ea50a828fc3a91ef8b084214bfdcca56348517be18ca472166dd7f18c8e444e3641486e7dada626ced8710fc73a2b09b6e9395b0" +
"31ee2c48c9183851357d230204c911b345457de602824273193b795fc21e90a0c1cdaaba36787424b23ce73e2116947f143f9641d39a4c07c2e40e02f3bd7c68" +
"6899fd57e3eb23c6f5615c9dbc279fca0d4218bc79d928e70018533a85b4646bdc78015149b4d41d77ec7b46900e7fd5250116ce978f825569bd887bf3fd0365" +
"e1259a7514116fdcdd6da3ffdf432bbe8e59b9bca9222c5dca1eaa61caf29b8461ddced6f312838fe490f742db696fadddd19bab8de6bedaade878be07aca4ac" +
"76d69b81a6890e66dccd702720c3bd5601c6abdab95fbe4ccde6e35385b75e1977d5085ace928adfa382ea2890889017b9c4c81d9ba4629771f84cced6280db7" +
"a6cd83ff9375ffb0a75a6bebba9a209f048788ba39127c1036e4bd0aad9be40754fd75295611e455909a818a3541af32eae98df7222353a4405da0e7be9f1cf1" +
"bcb823fdea7976a810e8a3c7bf93fd947f961a344a93aa1ba99bf2df48ec82769d8c08e7b14191050d5706a9467c9122f34e27f060dd4d6e936c414c4e551b9e" +
"5d6b5b58347ed0012a8a323f41b43bf5e960b2806de59da85b998affdb490fbc965d569114223db3ca65df69a617f6808bea23017327ddaf32990070aaf5f444" +
"a9db44a57b5c92bc27bc71c5f8a2b6929edfed8e182bf5942564ef045c75448450eb1a4e4e09a1875e8a4a74f229879ccb7a2f2cd0359abd91a782c2ec1f68bb" +
"40ce0a63bcc014b198adc222fc957eec0483f5b93f0db91b7ab3b3e3c59841dae057eec97abb55fc42b2de124946e66ed2a7fe8cd047cb79051b55f82594ab45" +
"711c92364f932a5fd274fe184c85583ac7cfaf258c57e296f9c18fd181308565315e27272cbad3b21cb4490ca0e5f675365caac42f299e22d8a74ca51a9d0883" +
"bb376804e234502db66067e7a434d38c3dc075346e888e4558b1745d00458df99db02f0e4c37702fb0989387f74d002a924790a6b7351ee0f41684bef079be26" +
"ee9d70b560c006cff4b08b9578afb5019c21ab9418ae4ecaa7a1cfed2d880a06a03c2c7711b601a2cb3d9193e1577b4f1d0e614c0be1f69205fa6524fee80bf1" +
"e1f1906b50e75fea2d19b8a83071a460145e1730581e5e9538888d2e797ee3cbd3b31399ecb4d6244ee44362493802b142ea397c2e7a3c1bc86f0ea0546a38ce" +
"574e1df0c27ad8a28dca70f659ae6a1369d8b3aee7d0dd24ea370cc2bc1b1a4dc9f63911b63e60fe4ed8552bbca10e01c82d11b0ddf748d234b4aa3b31683c09" +
"86358fad680dd2178902beadc4646b3eceff572631ff9e6b64d8a622ad9f0308cc46b7d422ce792fe5573e9b9480e1ae9fedf31edaaac3b08c5a2c6c27d6b033" +
"6b92a3da7b838bb0a2916ebb6ee72bf33a7fa70630491f49c67031ce4b9dec2315088d0a5cbf7473fd121e0ef5f4e92d43114014c9f8c6e671086a446eb1f66f" +

"70f0cb0c668998ed96ee0ad2687946681fe40dc46cbd170e0cabb6f6216be61221f171fb2f4273f58
c10d5c4eccafd1df62fdc8ac2c5c8f6d5eb637b71fa89e3" +
"f8347343f89667a4450c5c6e3791034d2dc3a593185b55bb95d8f8f2984ef981e4b692c1383ace4c
b2c4adb80d5d582857b5d0e3ccb12845a59587b47232ad20" +
"926efa78e05a57b136e284401c516296b6b194d541ec165d11ef94f166cb52f45145d745ff3deaf64
3b5c45573ed0e69a22f0e0c9c5367f6d1398105516729b6" +
"3f2edf1b01ad9633edf80efbba6555d4253fd99b45a36f16ba98ea0bb0d80533aed806544a084a39
8a692f698c78b9bcfc9b4d3328dd869dbf7085893b8dafe1" +
"59e0517c2f6a3ddfd4a8c670072b30c96b90f81fcc08523e4fd75919752bfa52a1db7c374debbd83c
a8e311b98b0d8275bedad215847fa8984cb50e108f69550" +
"f6517d719dbb5dade1d3c283357e14b6d9e85d61e33813546517e1262a7cbac814d79cf6b7e21b0
fbbee9b6314f02b2d4e6995d2231670884c78cfd86a2acbcf" +
"0a178ba64de2f13f022e22b9b968ceefaff374aff02b703811f3dc541a69a21d6e1c5d1aca48889b12
5ff1274e65413f61e42bb0194b60b65a3454c696033cc8" +
"e3cc3613a52850296a0154bde0e2a81b7a6489bfce505dbe1bc44e0e1052f678297bb19cbdf7970
bfa5268af8a54eee004063f9894118ddce7fae8bbba53a428" +
"678cec8a2bf6cca2b1a5f4a2e95562437e4eae41167f39d2a150f7c46c1eb6da35587f7234d870b1
6ed91c7db548ddc99967381b4bb4f3a2b0a5ebcbc7ab1b06" +
"7d5418768eaf7d526ca116e239ceb3ab393c45f3b32b713c11fa8e5ae8d7611e6008fa08d1305d56
55315a72c85a04dc853da3e8ea9d46674194e15226f126c1" +
"a233c26dd7d3cc04ae572320d0c351911b6fcdbc0b8450523e96022f4b964d4e479b6cb1c40a6d2
7699b57ed2952ef7fb3172c69ba7beb8c8633a01070ac4344" +
"d4c401acf8ca7fcafeaa59e1d4c2ff251bb67dbe10a862103df1b416fd2097fe412b3da9d4095b48ea
094fc3bbf2ca41e4452af3a179580e3bc11a7d97ba050c" +
"ae1d6b8075da267b3ae2231a1fcfce0c976402f34963c007d4f85d9ca95646990d1bb09691ceea3b
34211dc58409e052d0acf8c2296a7e8fb52d7c673506d89b" +
"847c369daec7909da8657e8976f59f2ef4c8a049b46fdf30d6d223ea4175e4d60e469bcea0eb3bdc
aa4d6024f2b43cf6de9bb40efa9172381291079dd82ac5b2" +
"39f2051a7f1aabcb8d50333e8c160de19ce1c76ced8056a0724ac630dd45ec4e315437391158a63
3c179a3d1f364b475454fd29c1e539077b9d5f7227786a5d9" +
"d8ec78e5615c25e517e9fcaf07611b85dff2c131a1b11a901a431a601854e5cb627cf7b8b0c5e66a
d6cf60b7ffd6c6441f9ecd58f414013279e9de533d8d797b" +
"936cfdbfcc78342b7ab586457541df5f3b7d1873612df200896e2929f44c6fe10d24f7e6dbe52b6c42
c0a40c947c1cbda2a41437079eebcdc29716d80957c159" +
"627e7366cc16df92cdedfa9f52edc848335f1c7152652fe24661a469fd503393229063c7ab20d8d89
5139a2f580dceac9f6dd4c4ac652b1d60c2b8a1b0b2923a" +
"86c31742807549e6d523b3c88d31e8534b9e05a6c63f6c8fb8a1eb4dad733d92e7071e410f0087c
a3074f4a2df511ae89cefe9ed09a8df603d61f23754e43cc2" +
"e42bcdcbe58b0587aba9a62f32c7507116fdc8a9db3d65d6c0097c8f473eb7f3bcd11ab81d5b636b
0812b7982201a63d0b8d40f2c38f65ffd953668eaa5751b3" +
"dab7f038aa7adbcd1f1102267c9d55d43649f9b4f65f1851546c5a9ef2c7ef56e84b16f12641e9d5dd
aa78ec778b5f113b2e06bad5821e1a5203b006a774e36f" +

"56c9336d92c8cd8bddcf014b6d58c394e2a93554af6361fc1bbd13c359fed98bb5adfa4dd1266e27
44e126e1bc029ab28fd68b648a2ab26ac23252171b298641" +
"2621f2a8697a00ab3fdc1b3b04921390ee16d213601ab249a51830661051d34eb777f690fc2d8dfb
8e0898567e388830bac8b0bc896f43003feadf34256a927e" +
"b4d9293e32ca135351a19d1246cda30551c87de1e148ff5ea576b67e19e1a0389b88a5548b3b1a8
cbee19eecc7de5c2333264c711d50d688a1c57eebc28dd6f3" +
"3dc0e4cb857973c3d0f28683a6f3c09db9f54b8fabeca9e4f9b86d794ca55d6611858f0d48736adc1
0dd6763ed7199bad81369ab1f3de30f521d43382bcccb7b" +
"be0178f716d5c3cb87488cebd7d9e2bbe671dfcf2512f1b815075777ea92a867f35e09ff0110e61db
24423d0598eb6fd078dde0dc2b5d7f5e0bb6fee207da109" +
"2e656b5c982866d5fe01e6db79809646559a6f2b9088e977789aac74435dc625b54296b25788bfb
bda9bbb25247d428f5141b03172fa11f12339b91ca96c92e7" +
"ea5a128c8046087dc7a7eba63e3bdb200565d8a103e7b3c292b088eb06aa27b43688c8516bbffcf
123499574f00908ff43d66b79106cebaf16725f1dee600a29" +
"7b3a3da878940867f9549e65c73ea798ca923b012fb8a7ef3e2ded1d2c4e85635219f627dc4feb90f
884ae6436e7b44f9159f9889d8e194828e079cd2ee60a7a" +
"6fbb6b8fc1f7355d7322709fabddd76e4283ddda3018b7882ad79b32bac133da415453eecd5bb1f0
deb4f3b987a71a2f2e60194cde63a42b91b39bfe51b4aa8c" +
"20952b601df11d170c65a7fe935915890849a367936e97bd242edf305eaf2f4f4fb9e5ee1464c51a8
99ba5cc69cf56731502c1b75d0d565b1dce15440b0de0f5" +
"58bd4f810bf058af99c158a2be0dd02a01bb5317f55675f4d42c6766fc61271954b6988c33a84518b
cedbac8de305946d060d19c4691c026953ebd680a4c9012" +
"0e8bd54675d6c33cc86e65f5cd3c34cb1e6fd47784a64f39e95a1945b5c21df2b3288f963863b333
66908b05c2bd499dd25c1b8e97329d7e435899afeaed174d" +
"2a2471b6e8d6ad7a0b1b6a8b19fbd976362283e5abffcbd2cd310245092749b23e0d114e7276229
53487f373c833281a74a1b97742ca99e49cac14d9102e3680" +
"404509889ace009c47d075ba9891e7f67b89aca3e213150f3c715cbab1869135601612d7dffda3cc
104b6508f56eb8b7e7f379b21e1ce290ce5fb96f53e3a7eb" +
"c7f7bddcbdfc266f23b775602d8d12527d30446cb4144df7fe3c2756e232a8ffca625d7b6ea2c8c0a9
2e6425ba67ab75160623c39f01fd96856b582e257a6930" +
"224c6da90a6eac4249214c3b85aef52835d904a8a5e224d59eae0c80a33b3141ffb31a7d8e62833f
a4c850fa6be135558fff5434777df45feed00316c475759f" +
"ac6e014e9d3cf23e7322281ed75623ed69a81d6f05ee7de193f6b44ede4a94ced27aef5ab9056144
593a836da80f5297875e7bd84d8ca6df95de8650b00b3528" +
"123132f26aabf755d00450648e44f3beafa4dc746775958c6dd88bee825c29112a3af582bb2ebe62
8d70364fe9ad01b8a9961d5b7101869044015148611 4af1a" +
"d85679bcd3eca510c6d6887e70e0d04b04fc2db5ab1eb21fff925b66f08f4fcbf31be3d743154056ba
137727b63576e72f1756029c86bbcf9452fc6cfd89f3b5" +
"9f243d84c410253ba7c9284191a0ed87b2513901a93606f1aeb736c90dfe40c0a343d45e9a992ea
894b22ee5d49e0f7d55d9bddaa6c74bde8ca5839db67b77a9" +
"ef740f9a47241f05e5dc1b9c95c459cc9db560b1db090daa3f4c6de46f695a158baaf357a1fc63ebc0
d9db8144137ec4bd69c5af89cdf9cfa66e06bff6339d62" +

"2c372fbe5a855d14fa7ff3726512f966e4da0556b29ca6d7517803f897d0e1911f9b46a291002a8320091aa7016cb7ac993e35c8b0f5aed3c94ff0b5dadd8b77" +
"056d06d1bed59aaf7bca8516c3bba6b33e12df2e5ca4aa40664b3bf48c4dc2c57cfd74c765fe9f794f55b5df6ac6dd2b3592bbc71354c8dd9ae41b0a05e1c7c0" +
"d3bd1a0ac6b671c48c01d4a0fec7a01ad11040f213461759f9e029c835ca1d22f9a661b69d72bc46e34b1be7ed85a21830fb87baa74d7ab145ac1647f5f2df68" +
"671100d4d9e41082d3c81f3b5a6e603bb33fd56c1dbcbdce5e213c651da45d9d1dd7532d9a955202338387af6315137dc458fed62920a0e721aa7ff1660981c0" +
"e4c3de0a4863f6f660a7c1b9745ea26036a25cfa37e1337ded405ebb0401d7041a7938800a97a032fcababcc06391a77a580b1a61de014db9d7e280ffa6b2381" +
"ab6969ae5cfcca00a47ac2fa05be02aae7beb806d2afcc11dc0642d2a12ecde2d2926efd9fe790e1bee19f9114d22ca42f438ef656a1311e4931ab7fac93ed17" +
"3f68ea0abed18cc2c8905bb2d599780690eabe4996e38872a3190fee361df9fecd5906f664106de4835f8fbb657366327871a2d38cbb671df04e0d14fe97e260" +
"c42eb07bd1d70514913c7a64a51e405cc92e06845e5a78981fef9822fc79e9937ce0513138f6bbf247f5c457da708cf84e30d083b4ba48d2d43d70e7c31e9482" +
"4472617910a3de4369217b4daf892c2c3250d1de0457e88b3bcb5c4568f9b26aa675c551a9a730fe9ea8145ce7f8e23ec825be9be3b9edd588c391295fe31ac5" +
"bfc97d2e438ca9bf6551728b3be6d6c6ac064baca763e0eaa24f754f4bbc84a4377de45fb6a8f37150865df18749df1af4ea911b62f616dd4dd4b25b27c7b6fd" +
"99d8c00ce8a53fde3ced091891e8daf43cade10086be046ee5607003de24101db49b1a4fb0ac270d05bab12583e263e903e94dab8bba7c785e40499ab01ff92b" +
"b82c2e5342dce84881adedf77cab593f541e4c963f4f9ffc80a16bd4eb7f20ef4bf3f57abc7cbd86332d8be80f0794fc82767d13c71d8ee20468ee35c13308b0" +
"dc29ebe8c6a81e02ee9a21807ff57e4d932edcaf59ae9e76f7cdad46b32f94a74982f0887d7083c90ff54058e873b10cec67fba1b717deba5356e170dec1a40d" +
"36c57674ad8d43c5c98022b553fe060251b994271585f702de3e71fb1c8e36293dd44a4b99a1baf33f6205e9fbc9acdfe8cfdf007224f93a7104e7803454fdc0" +
"9fc5a20be59f600ee734847257a5ad62c599a7fa836d1174a6291e61c1be4b310bd4d7b7cb9be976dbdfdd2b99340a9863c8c0e5009165d7097317e6c3a29cfc" +
"dc84b19bc68f38694998f626567b80ce6699124b12bae4bb9e661c2484f5109517318341287e142a849d61d0d7b11d4996547e7325f28842dcaed26367f7a888" +
"e58c24c857da2f48a9fb91c78cf351a23e82ae443223580a9fe15a6a778f6c13be66888219e3e15971170712b6c356520cc15e4e75167993b66e6f125799cd40" +
"86c72588a85f68361f1c2f09e87f9a4de95ef9a3b92c3313664a706cb72916b96a9cb50771f6917ddcf696ce8d7f2525745fb6edc30bf3fdaad66ca5b013300a" +
"7ec7cd274327b1b9cd931c068d8fa9fd6336d59f6ac79b84a24b34c47e408b3bcb8ead49428c123922e54bfcaec7e39c4d6ef79e5645a35f715d151e679ef5c6" +
"6f86cd013fdaab978ee4e52eea5e2753e693271344a1f215e1c690de06f29c856c469ccb484d445bacb16694f4def1537cdb32260705e8a50fd65e98a24967a2" +
"456af6cf90643638999389a35de6e192068fd2e2ec29aced58611560c792ea5c7fa37583ebd5452a8d94cbf1898937dd8aa6656047e6e03f84dfd0bded514a6c" +

"b47ca71c2cf1e76f606c04374663712fd96925eecf0ac1c38392390c8cb095f39e1814252ded78b55ebeb9915dc5e2ec14fe99e3a075bd389ac601681f154286" +
"885289e568a8646d94abc806b4637492e3a407cde582d42764eef0d56ab14b00e9aa1f64d8fdd533d4314145c8255c44d0c746af6da844d285eb044d57e8cafb" +
"ab6c3b962e0177f11a839f4a5c0d2c2e8d5f76375ac115e0a89f460ea1be238f974a68e0693d15790117106c1a65ab5f7aa08e738aa888d5b56be39d2078837d" +
"fb2357d86f5be85a9af41aed611b231495564493e46acc90c6a3e67d5b055320290aef508aa6a1896f19cb5633edc0fec023216726e50960a44d81e0614ce748" +
"6ccfdaf620eaac0517e8cdeb1095d55f3a60d61dd27d967eb26128b84c9ea8418779e074cee8961c5dac811ce5ee8134d3910a47de7a1344293f5c64ae8f1b38" +
"9d6c457dc74e7005c339394f5f24630f5e40cf270640d1e4c27cb6a74fb440f3203026acfcd31f39cd4844ede7e785290878fac8770f930e96c3edf61748dc6f" +
"b7476832cf77ddfbe8eb8e12fd002038630301439ef8a7659bb10593a92cb84018e1ec78856f403e1eb9d6343aa0bbd77a63d776f1d12838f27f3cf6296ab0b3" +
"b4436f0ec545a5a1e92a5351fb273b3ed56a40e5a5d25e0057f4077bfeba2e2d8cb17a553b157609b20bfb5cd2699af9936f50d823bb59a950a24b8fd15ed705" +
"b1628663f0eb5b5c2b18f000ab039bc425ebafb2010e1a2264c38fa2bbd0f77e38eac8acd670565490fd60cac7fd28d988c8dc0658505dd98425f22c94647d44" +
"5d0236b97ea58b3c71feee90be0055ce1fabda5ebfced9d9bf5efaeac8408c4b6bcdb39851cfe038d88ada5211de2f0f69e9e3c62453106c366cf0c40971c0e8" +
"e8f2a790aa66999a0cb4cdb57a8c2d812e9e4a66df2f001a57e291864339257ec26c9bc2dc6cb2eb5c3301c167e1ed0387f9ce9f76c6759ebe5c68e8be378c42" +
"e0350b344acbc8b40c95cee9e82bb43cef5e91a32a6be8a727d5fbe089321ede3abee4da6b9f41775d7e9abc36f6a5d26ab88ba32978b5ea0ad63f0ce8a772be" +
"5aa51143bcd00d78bdcbd69beb652139ad658dc7ad242b2057eacee092aab4940d6ff993a8c7d8fbe93c08c93c45d5f3a01058f3c75c94be9da1a19a97754734" +
"b713e1ad6b7cd472619ec1abd4cf42f50b0648661c2b8dbe8976037c094c7176090ba94618e1918db44f5d2c367a0c7f911132d9a8b2398b9417542c7ad99b53" +
"a7ca48253bab8382a1a24d35b9b9818bda513f4b52fc576a71fa63e72aa8042ee1fc806c6fd3fc16e07ed2caf9f82bd3bb6b393b2708c051c24c2e05aaf72531" +
"d865888db06f719314d6094b2c4f0718c151c88958d2d6c8a6f49464f81cc46709dde026f4e05325ea4ca2dddf9a79bf98bff3aa5eb412434f0b7457b4ed47ab" +
"85a212e0c7720c78c961d56141bff0f964622d4d3984c1017de6f5846c72fae0c771a819ba6c111bd739fcf16f4b85f8101e7c3f0daefe753ec130a6f34c7697" +
"4dc531f83715ecae28bf2e55111778ae42aef17fa95340584cfae3d4599af9dbd10211baf3aafa8ac8a07edf8243daffd6a6486b1e3be4b60711194261e2b646" +
"e2667554cc0bb2fc07054b653231cede43154c9002890ca20b0ac81c4788847c6ecf7c174e528f36f8cfc53f3366fa9ce07b1843939cf6d318ed11f7ba6eb791" +
"ce25e75cbe37d2ee3d45bea487d969de041011959c0fed4e6c86802a7485fad70059ece14a29b03d4df41677acf71419ee63f1101060ca5e4ca0ab2edd71fe77" +
"46c6bd9f36bdbbf0a9956eaaf974f7bef982cd34881abd686fe77b536c85d042d77dadd00c5cb0130737e5318a025e6ae6af96ca28cbd41094d86a85765ff891" +

"af825793910c406470cc61be5d9282911d2faf82abfb309598fce0101ca64abe3920701a958c20ac
35927733466a23de809afa2bdf331f68c3ab0cfa08b0c549" +
"a20e9b50dbe85d22d215d0e5fef854ba271a4c0f95e6abca19018bdd4a042721887418136b4a60cf
291bf06ec47a5a4d2f9b29f988733c6bf6f65da5a95f8939" +
"fe0f2bab0bdce98569a81f861014e532f6a995542db02b6bdf3169191d300fb0429c1cae1d2dd4d29
e0b61751576e04b558d38d3afcce8326c2871e969c1492a" +
"8391c0becec29edcf7f038a8093471763db9f13b97114acf7a979f5ba3bf6f990317890ee0705850fb
97bfacf306a0ad621b2c3b633af01fc5aa059c0e22ed17" +
"23584dde6cf140bd1d0087ca9090ca9f07d3b93c60938af8df976555455bafbd8cc986ba32fc3f15b
5962dbb2d37b6ae55a7de0c0c6f2366be0278e26bf9a725" +
"f61f2bcb545d66f79261783f7f03395f2a5d27e56af62a01ffcf778c3c686e244bd9b7e5029d1d40dd2
250705c6825bf78e83730212640cb5ba54191b61fce33" +
"ce6df7721b15662162b631d99e6431efd24ec35639c2b97f10374fa5b9e2ca4231f523195206fb969
5ec7721c98d74f29533cf714866adae8edbe8ed2d0969c4" +
"9ed36200c4b8b75131e6d1efa913106bb0759aa8255bd6a9dc2b00407358f4523486575b111676
730094f46d0a7b95427df74f053c6611b4c465efa5310f760c" +
"5ff081e841e5f90c2de35855d45a7f35ce73d7c7f9f61fbfa953398e042c3946aaa4b7a2094d95410b
8a5ef76c8b57d49f77311192b3f4578f37bda1a426de7c" +
"7cc54b5400bd16bb30cd8d1b7b42ff31c5e3759e3c9a7668174c02bc5a08f1bcc7e3ba145fa5f5c41
e48877b41b0ef8fffd0f75c6547047c2e7b7c7e1aef2cda" +
"c4a778adbf71257618b4eb3c6dbd8211f829c1d6373415b969cc48f33d586d2678e7c1b441364a9f
e2bb426a33b2a132741fac547766d196df3505fdb17977de" +
"7853cdcd8d9932eb9452620aa4921b4416f65055d77573b132a40795bf142815b655e670bf2c446
4adb5d826a1744c8049d7a6cfbc8a4634e66eb32f0cb6fa17" +
"ffa8925131c3a253101733406a2a3a0dc61ec3ca1448623b6295791d4e2d65d303f78038e15d0ef7
5d823759bcb4b277b51410c37d5efbbb2e3a9e0cd78a8475" +
"05d44bb1fed7f72b1bf1a96ad0148e816d34c66b1b5faf172b8141ba007bf2e5dbbfee4b09ef66656
ea3cde54f086040d14116aa7f3584ab6773f6091a2fbcee" +
"f59d6ea115f88ef9fcb358c87c35caf7c1a6022e141a3c688beef17da5a619e733d854218b30d5edc
39b933b19dedd6750acabc52234934b08f930b608a18008" +
"838cb0fb73d4c78af0c468d9fa4fc5852135ae91ae00a99a6c603371d09b031ee37f87586cdc8389
7d8fd8ee2e07b9d0478a812d3f7eddca08860386e3ad9521" +
"98d5fc04fd0aba4b3da6ab8bdd9eca8e0399a2012d6158ed75ced5f432a223449b4e3db3fd4b19c4
94a69e9f2670833f8a88f7b2873319e9495f03fc69b6d098" +
"6006e3ffd8cdb9c1b98f72345848deea1b98ff6ef766f4398e642e5f2b217c1a87a608c1dc701bbb79
d75a4433ca1d600061836888a220ee262124d145d371f6" +
"576f04cf71701133787a97aaa615ca98138c2be1046604d885e2f274b0de8743af50ad5dfc4c3a09
164448e102be577eecf77ffaec1724f91f00f908ff6af41e" +
"57056dfa8f5dbcade85a66c10e524bae55922b4084407fb36ca8d6b7322f76a8139be9455a34440c
719d0db8a36385efa48841170c8d35046407b586f5bbd169" +
"7cbaf6819b663fb17d0f0ae89691a099a8ccf47fa61fb6dbb22b3298e5cf2465e4b93c49da70fa7692
4fdf29389194cc5c61cb4b3084d0851bc3018270d1a24c" +

"b4b04e8af927d9fec9ea1c9ce18d4dbe61f7aac0ffd4e7c2e9729b49ed9874b883ec644864c6d9ad0422c4d89f87df1dfb2c96314b6a3e19afd21783f365445f" +
"bef10562a26b48df42dc344ddd63fcc03220dbde98f1109cade221027c40f0f996f4beb29513c3979ba374c4c6a2c2dc6276ca6be66eecf1dcb245d6efe78aea" +
"e49ece37f87894bef3c0cb1b993d974685564e2476c12c8d8f63a1aaf142fe34a6840be340b64f96d441f4537dff434ddce630101ed9f78e807881f6b7590697" +
"bc97e60accd7a135d8915781f4fc22e437145154dad0a39e5e306c117b11deb10462ba74d58e81de7674ef0bcb20b38511991447f63ad906b11abd4ba88df3c9" +
"e6931f87fece49f48543fed0439c88ad78f82aabb32dea03d030bdd76efef6b737daac2de2db1cce10e2ec74565b0a609606cbb6aa259ba88715229b8176c874" +
"db3fc4f6db9f167e7b2d55b33a261f9eecb69a0d36ecf9ec4f8f9cee5b74bcdc5d77b02ada89f56259edeec0d9ea866ccf454b9abd29d5d21041179912a5c302" +
"1862d850c3ff483e09479957df5bde03a29504b4a43e1fd40af2b8a2653a37cae89c8d917aecdec3959fecd32b7fd313a61e134abc15ad008aa993aba9629a5f" +
"0af0ec713f742bee096e171729e70530b60f910ef83746a61580f0cc6d67723792c0e0e94775d5b1edf37864a50678d197bb34a97e84d7f764c0bfe05f4b2d0c" +
"dc431d1f4410500dbe2758eb05bb6b19b154707c255a97cedc6aec1841f1817f6bcba0b9a9c1d3aebf747bec4423c71309fb8b4ada90dd9f7adbcffebbc905de" +
"74ce531403df33457c4d0b970fea5df4f85732e3c33c5b8242b041141a8c51a62f0bb14dbe07b14d3f5ce646d76e87b258e9b62128f9c0c0a8014f2c5b3d3dbd" +
"a3a77be6222419cd3fbbd3b842c46c099f142bcd36442961e8205ec5d7fd159befdbff12693953307026f1e06fd57b6447dd3cb52df466f0352cc46f27d1fc56" +
"56e06f68ca2847d291421bc9e0af6bbcfb7b3ce07600827809506ba3f96f40ca22766f8cba32d4461488f6596082a52c11e9ac908922075a7b443c41e55b719d" +
"9cac9fb587cf02432e1accf3cb7a16de0d5bc3a1c0aeff5a1795680b4551316e3d7b5a9bc63a09c6f75b0f00eb69fb6ef5130c1ec40c7a7d5d6ccce364b74f63" +
"a836a4a711027e592d6a70e10e573cc6d730a0def4a7a2d4dfcf3b0aba37fa2060ae6935710191c023a0b8e123a67ee811ed43b5127a1c4cf82d52ad6c40fd66" +
"1160f77dc320bbed349c8b6d08b2a7a6234a8dc88e4744b51d2d7c56e02f1c3bea9e6c2c3d5522ca02ec7e0b8160555eaf28797ed30b5c931a73562791f5f0a1" +
"b7ce83824bae17de449cff41312bd441f34df62904f4a0265d6fb9b8a352895ac6f0025d6b2074570970b4e679c559d03ef40794708eae36567008d9c33f7fc3" +
"5f8df7e901c27f408cc7fcc52631f1178695ea660d07df541e5a32721d145a32e8d32f06301b5073149e8798371fdb1a2daa5e1d02c24da07682a2cbacf5af55" +
"64810e479e5966dc6bfe14b4472c42cb154e19f7b8659d42de5ac926224cc6b0d8f3fa797058fd6e21ea85146838c4612760d84e24341825b6931a6417327394" +
"0154125254d4e11ac80e475a178605e851e1be39695cdc0781da241f232cedb32a04b1cc7352882fb635162ec3f5fc5004cfa7d03780753c14173ae7b12a71cd" +
"b40d4835023a00a4803bdfb6916956ade9f687af567e6f29981120d306084ad061ca1585f0e9497fdb27f9d54cbac8fecff176145114ebfc17e3f346b246ab91" +
"094dac0e684a708b45dcea16378fc29683dd033310068339b13d995dce77a50f9ee9cb4cf564566b05ce352a21159ad21e720e05ce6069a5ef4e9fe8ffd28516" +

"8356a0b80c4d1da547776f486a117f6f7ff6557edae7d68834cd71973517cfe4af045ad0fecdead68e
dc8017000958b69410247a51bd9bd3152dd57389f25223" +
"d5e88c0d343ab3aeb89b763eeb7ee48b3966fee147a4614e436c9a1a398487c80a001700666251
b3dd6a2e5dc96814d21e6498e75811ba4c51160cbecb7d5510" +
"62697171a25a6abbc41fd806c3dfc83daaa10d7ce47f5a29ef0d85dda5a61429c637520e6a404862
4cbb25f53977254cf803848ad81f25eda07690fe7a0466e4" +
"d18a2fd145dda1c94a994bc4ba5ce1aa1b50c38151febee757afceeaa91c7b35e57b90ff7b62efa92
9dcb962d32dde5a0bc3159524728621a3d7487eb7c3edd8" +
"6df3f8a18e590039bfc84a22b23b11468c90dcdc8506187233d8a6b3dc9785ddf6f341709fefccde9
1a7a0925a8446b1896aabd6a6826ef88b756a9711cb3b78" +
"1ab1f4df4d0515070e41fd5b0c5483270307e60eaaaf0b3a48f6bb96eef6141075445285675bb12f2
ce38b42c91c1e063400d7bba9b322a0783e7d2f5d3f8874" +
"52ceb65bdedaa032336d969d2e0e3007d2ac07bcf054b9d0330f2e26c486c054bfa709fdabe283ed
9a4ae67cee24f40f2a5c4e70160e6ceead208ca400959922" +
"70bc35be104c9ad94cdbe288b1c599db1758331340c9e927bc9d688e4186d5badd463bd3ba116b
dc22a39c604778cd95503ce4ce642041e89bcdeb86fb19ab25" +
"e1f94ed2a2f857b228ed4a582ad411d7273a0d5189bf7a2b87a135753e03383033b989ea532041a
b9ed397ecb3ce61e01923b3729068f6828ffd12e2ab1d28db" +
"6ed7423d458decb00476657a0580b4af3aa5615bf07df55beaa2bec71447aeb39791477dd09349b
f573e29e9c4fd454b4bbf1e19591bf38dc47c83bf39babdc8" +
"737d69ce4b586cd10ed406426b88e686c11072f04c680e8b6275166e2dbe91f701642b1b4ed92d2
3d6fe14f39ff7f5a09401f3a398eb4bb742f6cf10aa35e767" +
"7e6e92aec791e94f8122e8c9cc9d0bc979e3eac6562ab614ff20330b00d9cdbd08e9deaeaf5cd67b
49164f550c5f5c2d7523fe5ad71a2bd03fe2a97329980cb3" +
"049ecf6d677d815e56f7cc27407cf73528549ea98265ef90277c14763d5acb3572f5a482432cd828
8972af580fdd3418889bfa6a373c4813c4c45e933ea4ec72" +
"cdd068089c2a30897dd57031445834de9f23faf506ad930d843b1cabad2c0aa8965d1b5e57032c9
69f9f55fe2a3049f4e63d5b5c6f5f760da5ba44e3bb9307e9" +
"ea39973d05a74a49e15bb71eaecf62373153ca316fdd40b1c64ce2896c95a7b5df970c2bec85edb
d5ed84fa7949c08d5ec4d987052fffe357d444e2408a22295" +
"6ac1fb272f5023740b381e00dae9f09751a33bc6ad673c4221ce3f932164deb99f1da3eb3581caf47
5e385bcc56d47a7a1615a9543403750f0121d5482c4ea5c" +
"94fa3428178f6a4deea08d754ba2abb3d1aa48c3e06f06ffcdf4571579d398cd991e60599e9633fae
6a0c07e10e538aebb7d33aa127c830f14b083728f6ad7eb" +
"c9a60a0ba222f47780eaa82a21393a49defee97aa8c3aa2fa53a2af86059a7587074128c2fee7060f
398ae70b156d53aba0bf1af4bff10a966ce7e6382cec4b6" +
"054a8f6b9ef0e8729ee182f86c862f9b7d5ea36ef7e15bed10ed41b25738c380e58cf42795e320274
9074fe5cb6e8fcb49a116f54d84734a834609a3443b8b42" +
"97c05ced428f5756ba59bfc1535bc7e16d370d81b72b1f3f78ba75c820b22e485dc042e4f38e93cc2
918a491750c92998f03aee571cbe9abce4d00fdf9801f9e" +
"8e0fa276822e1e5349945f1d337e656b431c48c1a2e9d4142ea14e9427881bf201ad8cd8effaa6fe6
a7e07c8112299db1b327a0cc34c9fbd35596f4ee25caeee" +

"221afad93ce7df64aa6ac766cf4fe1660446dcbafdfb86b4e0fea78c29c3e84ce42da4a503178bd250a6dbc4fc65e397062229001da05d5be118dea7ca5ce67f" +
"b4ee07a8b01e408aebef2c913434921df686a242b7d015a559f9efdc54ad62d7f31ceb72463041843d7fb60f948fed03ff143fe24ab81bd4ef6bdabb856ef1b7" +
"174cc987436322271bf48423114e05758a08cdbf300931fc7e950830b7ee920f7033541f1db9b0d2b91cad80d06c049b05fd0a76d6dc211bef2a08d53b1d16d4" +
"2232fb263941daac4e004542517807341bde98e9990a97739ef86d66c7a51324f1f6911cce4c3db37ebacb6e58eb02d8f7d6ea31338b56a99649c4e730a01bca" +
"deb6fc87cabe00addf1bf76b83927de26bc2bb3f0cd5945d863b0c31cfe8fd4b60462000a911755cbecdb6a98139041d52df498aa99aa3876836ce5b5bb426e7" +
"c22b5977902e0b3425fdbdb8f44e8758b207b469c3e5363f552c89fbf778e95e8b7ff6566ab591fb68a8bde38d8169c708a321b669c08d9ecf1a06c5321bb1cc" +
"9c8a585b6381645edfbd1ea4a2ad7e7eb8be6c431958add393c0a257aeb283644c6fe97580aef613f1b9d83e5b009f7a4d059025c11e0a0a67801be511dc097e" +
"4e7c065684effcafab83e0e716e2d0862e83b295f82089ed3ba4f6897c8d8eb2b358231f95eef840e1fe22e9065de2b3dfb3633e2968135756cd9c109e8acbb3" +
"172bbb6680c2e4fd69e179916a7849315c9f4dc86991d75cc6358617454694b3fcf2471ec7fca6ea2d99f704b9aa37a25a3b3183c5e32e3711346ba2336d6001" +
"489afb9cbd8822dbe4f0323ebf7cfa9367b6548213d473c0f07b1bb6d16e1c66fd2bfa1ca623e03149fc81eb6f71c12e7b4b76ca588548bb4872469687f334f9" +
"7e114a16a0a58ec70ed74ef69dd96666a85aa52d1ca812235796d90b9af4296247f4c1ab632effeaaef6acbb637f1aa9379195b3b668ca541bc6eb595bbc430b" +
"28adc5d1a26fd4cc2239516ac9ca9c0c028110926a2f88881a5886554c31539f4c8260e16364f4ac27710d2becdadf573f4a2b7b55d76ef059432c91c6f5beb1" +
"56686a620bdb4aea50df564cc0c5ccd8a93c454e06b8969a0f59d63ae5a29105149c08a5de65e87b0dc445dc5d86db8788ba77b83e22125c69621140d7f17906" +
"4ec0157a877cc51ce3c0d565bdf6c884f69b0ca631d6863770f6db30eff847e33c8b30d5714668a38a09f454ee44ff2b7f97207f10efcba74325378f6f272ef9" +
"9f09c501c12bd0a4155f559a604204b36751ce8d4c0af35a8b445a9290c305d5d3ea21f944e31df9a711ee90bd16a37850e2a87c3bd3fdecdc6e2f261a5d6d0d" +
"580990fcab9228cbb39f8c79608d821ce27c10b0ee0b5a96474759f67970cbe03cec9fe594765bc935abccf867b9717a4087465c8604eae89497c8ffae7e46f7" +
"ade2848916b54aa796809cb98a4364b7b44c17944dbc408909a92d4cbb24a514b72fe8de7d1cbba0a101973fa9b29d97dcf1f4ed8a05d5e0cb38849dc6e2d041" +
"16892ce649e0a553a727bfbb1d5794a059d6a411e43876e561d83bd22c054680cc8fa928f5f4be2d849f02ddf9c6d11ba35810b81553e1938ab013663f6ef35b" +
"08f06260932d7acf99f57967eec57a61f03d880c3225e53102a672f5842da21aaaee02444d372ab8ed7096235a4926e3288912d9c736c2c4dc49918abdfdd6d6" +
"d0df5be0133abd61b02a6f008909c5350f9077598ab2e612603431bddd3826e314feb280585b37eb89e597f7f0bdb738a9a93d9af224659d50c8f7479b240487" +
"76c2a960eb18923fa2d3b31b3d20ef538759cf22f5b415d19bdae689f2bab651d79ff99f77a721bc1b2233da12c12be0c9881ad82fc97a6343b3af8207dda8b6" +

"5c600644d741b8a16750964e341e060260c4de26f991f3a1f6a606d1153565f1c9cfee58eef327edc
0e9cfaa206ce930b191f521be2344949bc75d583a413a96" +
"ee4edac424cbf9bdad2883c96a1306b96ee059d8044e3b7af4e7138697f142774ed6409a86a3c6c
456600d4e405e6117ec759f4b22d7e5a185b0f9c67ad987bc" +
"58d2e8c929d4a487e5b77201d7c1416878e8d63258b2f58727cb631494cf1d68b99c28493b99b04
09ccc1f9c218a2b95c45ff36563f0045ae5c3098f641ea6a9" +
"b48a3e1489831b2d176a1e0cb2afe6bd8cc5e797de01391e47e798c1aa945d33d5e7dd607aa73c
9efe93f0646adcd7e211303ac7deea4d02c80370e8e867e2ad" +
"9942bfd5a66143560a1f59e5be1f3aeecd7eab689a4a481aec78045ae0604f69d9eea550152f6e2b
c692529357b509d60e5a497bd94e63dc698cdfa2a3a55976" +
"0b2d072a2fe9c1fd41f31518aae0edaab532591476a9c5a61cd76937575cef71ff5dd66e158e7820b
4b6bff4067cc26ee9aa66f41b80f078645b920512b5efd8" +
"88b3644601a72e3f665b9c8f0ee246593667379b8fa043718f2d75c21d2a11640c328971c32d5743
c11ada6c95cfabf1c6b66e0b09342afc899e1f272ec48a7f" +
"ba5a51943763bf969cbac879363e14dad1952517d8f4b463511adccf25e655bced7cd9666d01dd4f
2a0a21729ac4f44970c9c478a995d1c3b358a244110f1db9" +
"fe6335685701e0c2660ae69d33a93a75e44f5374b979a5af140200db43ff612be2728548192ebfd0
a3860a9e135b910fe3fb249926d334167622bf4123bdf0d5" +
"38e9ff2a3bb67a44f8407328e3c94b47d92e0401aa1db85459967699804df245a7808f972683afde
d9cad8fbce15c1be38fd10c62c7abc302eb0537d5cc573ec" +
"245513a87c1a8d386f7ef0c4a91ec3c602b14a14ae395da13284df3391b929c7379e181c5d3d459
7e3c955ef6e3dc2fc55890df04785bdd4e3fa35ac775f44ef" +
"9d7813cc036d6bcc316e869eeddf7b30e4b837e9285eb20397b4d7e0d12080c502c750268bcd6ffc
323cb094afbe8304ae840d37be833878697f2cf931faf06d" +
"28dc6c7e1b5df69327127b47eddd0237f1bb5942ee5903d8cbfe1b11484199e90fe7c8e7f2f725deb
2293630bd8c8a377d539736e2ccc2b90c08b97abd8c5ce4" +
"ea91a6219ab06c61c31eb48a35587b3c1719f387bd8c2063c5a79d041ca8a9ffac2e3c728f74efdb7
4ee0730f84cb3a8aefff7c8a1b570127cfc93eb6d3327f5" +
"ba7f886dee8be0548f710d6bfb18cbe5910bf61aed2c95028006f419241d968933aa00bb0760a41d
2693465827a00837a84cadaf8a8e804d175adc5915c6cb6e" +
"fefb2cf70db063f2f3812da17586436c176aa0a815dfc7983eb88bfb1b6d1db7ab119cd3058c0db4d
1910034f70f6eedfee8b742ea45af9780f415be2f851061" +
"313a218ad48e992b75afaa07c33ca47ee0155fe72e13d7e5736e512c5e5a45d351f7c7902d8b0fa3
1b34569a9aea31b018d63d572a9898c389d07caa427f114d" +
"251263d56cf5d6663159c2b32683b266fb909ba9d4caadaeda6700c03b25307cdea597a3287fd76
082dbf33f073482872fdb494b892112c594d7f265d2799b5e" +
"5ec46a30fbf1557fa344a664a7af457a4e8ce2c014a270215d3f95d47a42d8f86a61d6d6b363d04a9
9a0d8f06c5b15cd803d951aea0ca185a807ca4c677db789" +
"fca64f0c5ba95b8c64f930eda658f9f773a9e1c8669589a7d98ade8dbc2c2c4cbbaf6ea2bbc6e762d
4098f4db0d3f055958ae9da15ae57ee0b60fb9513dacf5a" +
"d65e34613570186acecf9e165bfa470aabcd35f22620497fbcabf220c53cff84eec12cf9965297b364f
0e9122895c175d213fc2a9c9cbf27ebe1cf96fdacaf1c" +

"1c79ede66cfaa5057d33e09b31b43869812e5a0ce730663c18c4333141ae9565e437d99ade6b2c
be005214e8b3392c55bf4d7b38ef16e7f84b4ba3c85e1dfd1a" +
"ca8da1a5c75fd190e7752926533327880aed1461c7e9de2443ba0a2d094f4a14d5fffd3b102ea78a
cd34d162e82ab78fbb82bfbc8a9708ab532aa861643c39cd" +
"2bc89f2be53c583f9930fb2da14f1c5d5f218384b1740a76bd8b7ddd2c9888c8d7e7e78cc7a3304fa
41995c7c1c3316894296caeeb9711f0e6bf16abc380bd41" +
"10448be3cb03cc3246ee7b9559c858307001033c84ecf89690526544c05c146f206d4a21e710597
bb57759d232154a1f9d88eb3f3440374bad1e901da7a154b8" +
"39a6d1b1b6b2ab0be872ff036a9f9f769a169fbf91bada732d8f28c453b9be49011b211155fa5c588
b43018775f99e3b92b322a4c41282326b79fd26541ccafc" +
"c0e2f09797e3217fb0e5785b72e654dbcde8ba14b2d56faa2218748c6789c158bb635d43c9a6469
0b004ab70f457e9fd959b2d90875966968c7ac44b103283e7" +
"50b60deeb1f89444aee25fbdb7fa3a96d70c3dce38246f111e466cdfa3b807c54ed584f5b1a64456e
923dcf37f45b36cea3d602ba3a55a4fd883ebb6dc198650" +
"b522461614656897b9b7d408d48b12e594af06c91f715b32a4ed65a379f0ab461acb9b8b20d1f1b
12e9f7fea422c0c7d545eff4152e06002cbd120fd74b483d3" +
"a0ee30cfd851c98e9aa8fb19b60528de4a75b412bed656933ae8ab600aeaef5befdcca4d35fa472e
d38ffb91a9017c19c5d500426f262ba379034c45cf5d1627" +
"48da223207721b4bc4504b79309f3d622c53dfe3c83ff8866dd7614a2e90a85c077b2e18bf1cb600
8f0d785d6a0ffd5f15a83a343036f3fdd25314bfe47b5a12" +
"58a7c89475f39a58a671d0a17f6fd100a8928181b94d8d53149316d5addf14bd398b538e2593273f
02cf296fd73ff92d02230de939dae94e03d44ce93dd4dfa1" +
"b9219fd369c854ec409d7bf94b316e5e9c16e1ba525a783e24bd3fc0ecc949be245c402efae8ea77
aaca74c78703506cfd5a5a614793e04c76652b4f344f79fd" +
"f2da1e34f650fc1094116ead723813d204ffe375d20707fd94d90f21c009194201c88d22afaee83a8a
6be7518dc915331b863664e033d397c64e1516c0fd9324" +
"11614a1bdf2feb86e0d0ae21e784a55086c596c7eedd44d3afd7295455450f507f1c1a33c9ba94d5
0931ec054d8740510ea23990c266f30678a74fdd485b482b" +
"cbfb4070e3f10b66c65a4210794a3137adabe887ffb9bcf2a30c625138f840b2666610e76e5a0abc1
83088a94930c025836653eddbc440621bbf94761c74e108" +
"3672c6a914a753fd452e8e7a02c54b21d7bab4b705b4509b9b5b27e2e5144289eece950c3634b4
10b5e3cf8c5a5f74d98b55d17d45d7014390cf696a7e693777" +
"4c028517062a69276910cf5f139078e8ef6e77aa8b35aa55fd4f53e48ae6b4875d1732b286ffe8bf8
52b73af7b964fdf1aa4c4f16d9f14485a2b1a704c2615ac" +
"8ac74eaaacec7e8e4e506e1b418d377e4d5a271dfab47b3d3c11a809beda596fdf37935dfe06c147
dfe7d5be696ffb2a0cff907d1eb2a88477c261d5a7aba06c" +
"d70dc52d00b9a9d851e849f86e1cba91b4c40d1ae3d4f21a2763369dde34d084adfc09d2a6cb5f09
114cd8d6fa26d15f1ec428adc245064e5b8e80f21b0b3ff2" +
"6690398d3080f5355fc082bc4bf3a38576c7da00efbc80839dc9a06fab2b998a152553c36fab42e03
e3e4b54456ed954e53bd63902d89e2617a263e70146d1eb" +
"71557baf43aeb0a681f600a784778c895afce26fe17e3ac33990c54cd96fcb2432de79d4f95ab2fb9
6effdd37f4e4247ae5b4c1fa461ca3269d45a90af090333" +

"fc3ab5152bd5aed4445eab93466701382ba76fc8745abd911bdb45a494e1c62647670380c04377b
cdb5e631318dfa79850469a988094acd48a4110bbc7289617" +
"ce436294ff242302d154ad75437ae2f551df5b84f884c87497de0bb2ef7bd41a8c758e4b158084c78
ef047389d88974faafa00ce7396e849509d39c403fdcca6" +
"8f47e1d0fc294e5510a07af24c165e1a4b4ba9498e7b333c4e8624c552801079775fc684b6e98b72
ff133164a2052c2aadcef168d9cdeab8a935c98f08e23b95" +
"859277381a2ce23ea61fbe9ec1439a489523161ed370b0069aa6a5c7981e4a80c04e304ff2fd85f8
0b51e3de3484b53084f376cc72a390aaefc49baddf4d2545" +
"93dcb5a49326c9c15c3d1c0e0709c9879d68bee07b956d018a995bf1e7f8fa03ef2079d01e0bec60
1519704cced98854c94f1f0ae837653f14c0221e12f2cbdb" +
"1564066062bc1d4dcf7ed8b2c980b90e8101842d5844375cb370f402d858dffd9eb52572f8420d4d
246462230ca0dbd567250e4f065730a6aecbd804b1acf949" +
"30e2890a39fdd4c1eb693f7e345504dbad5ac207f1a649968c3a7b416bd972b6a6bfde04375337a9
3b0ac08f6fae62c0fa7df8ae9deeee421f7ac62d8cf5ecf3" +
"b5ff39877ee4abaeb9db03d8a8f13f7925e54267a2651c55ecf580d5cbb24bf504fb01291e3e97ad1
696ed995608fceda79f2441ca67bfe3c31f4f4bf0fffcd6" +
"55408744524412cd4d3cbdbdd216694aa7477e88b25f7efeb34abf491a50695ff686829a8fea9e99
9877bcb37291b8dbeeddfd44465a2c28a215aa532590c487" +
"d4747b6ece4e1aeaef725cb305d11b965a9647bef36a5c2fb45cc334d35ff4e308cd8813b6de3953
b35a4ef6a3ae07794f8b54ef6365a573135320612bd1acfe" +
"6cac5524c0e98b6f2a33a790b94f5134f0cba075a6fa93c191f4176ca62ea2e365557d6b3363a17b
9ee52f3c347c82cd19f8432d16a934ae9c5d4d4505e7d20e" +
"1ae31bb64ccb084f7a59744b27d58c2388d449ff4b63604878ae858240348ecfcb51761678265bd6
0d5dd7d51e25e91668fdf80f6b726b29ef6c3f0f229d8af4" +
"b2cdadc3ca7fbadab49b28819b9c9b92b49cbe9a281e5891f4eae7616013777605a0623dd7a650b
af9a9dad66ca9aae3c76ef1e27db32bd9514a2776eb0c8d05" +
"65eec06fc4c8a69c417efa336842e248e5a51e3b5f3ba3227e3f78f1bd12d81595e03a01f4259c772
fd481ab5f3d7a945e1c95fe0dc3c4742eeb7e15c9426ec3" +
"ed4c90ee07d56acc78fecfd7c5ce1e04e7db1a970091f15c90f0aae2865d135395d27787aaf68c6a1
79064d82691e0b6c795f61875f317dc6d2e8feea55a28f2" +
"461d74e14e350351720b6f536adfe3addd4111f08e3a84da2656fd4bc83989b329b383da9f01cf23
92aa0b19577884d1281f2e6c106df451c078a472b36057d3" +
"065dfc4bbb47ce4e5dce4acf6da095bdd10322f3ae12bcdd1f805e73b303f1fc7a7e16cf3ffd822bd8
b25fbc93be065019e394113182713f1ad299ae6537f6bf" +
"57116e8dc9ec775519b797ab4107c2ac5129ba85188852c3bc5f116044bbd8985b6dc8b8da4659
589bf9d2351c4c3adaed87fe2ea20ef6bf62224c7af86fe8b4" +
"973e558f39465dff43bf23cf1f78957514e4e82a3009d40d97bf8d8442a11deabde806e2fa84c1ba7
5273da75ce8dad3b2a34786b2958ac4bfd248ebe604a173" +
"83c727b11dd922b1f72476af700b663fbd7033d0ac74b463d40a92d26c938b69f96fb4a9cb7a9ca2
bd9496251270c0c5fcae6b3c2eda5377b897891648a97125" +
"8ac71fed8dce8e02c30961a299cb7f3145845dbe8f4dfaaaf4baf0ca3fb730abdd258e98215f072a94
3d5aee8d8bc4c86023524f7b69186d99ad88ccdfc0b4bd" +

"7db422bbad7eaa0824ce24b5c186e172c8c584f1cc5c126c901a69ebee8dbd230a653a3643b7875
672d22fd86079daf8d834ba33664f5ad0b6eec767b4f58b45" +
"e67b776b90e0a5e130aa5365003eb7fd78b757b1cf9133f6a1d51064b293cb42c8c41b15b7e95e2
a39fa5dae19c6e20031d2bfa4632c37779163fdecc6b45624" +
"4d6bfd01a8877f6fe7739591917a86e7dd795ad85cc3f256cff5961e8b62e92a0754a51f2c6d59819
446eec8bdd08b87cf9f4fb5373e809d52240d2dd691cd50" +
"37fc79d35b61d63851917cfdba164868a3f79e2061bd4610c1f5216ed77df00baa75f949ad37142db
4fd282a5c7d2e2636ca590f92fc4781d4f51efa69f53947" +
"d4fca1dc7dd2429837b6d7e5c9528effdecf6f731f676587785e5c4096bdb6f1f44e72f5f77d9025813
e848881506f65bfb0f2b2d3ae6f9e00731929b5ac083d" +
"b1c9c324703e63fef6319e1d8150aa0ff7d9a2049961df9158f3e1f2e540a91feb742625d2a859a45
2186d2ccaa3ec2ba086ee0868a4dc24ae6818fc02f9c1df" +
"dc326cc31c46feefda97265238592f638968627ec24903b97513ab05ed58ce5b516decda0e2fbf01
a70e6cb2e53c3e7b8855f80cd7e007b78da727ef0893e099" +
"592ba684d62ae2d1f06ad148fa7f34cfc724d804149cda21aee7eac064ad20d29132b260c2c2867f
a6a2e747739fc30df2f002c2a99da6c7e64ee51e806af7d9" +
"768aec456b93a05002666cb61b2229c99f2cdef9afc9cea1c4ee3a85dd189823399781ee33cde2ab
edff09c47960c035e075a29156005d75845a11fa06abcc50" +
"5f7f849a0caaf683f334e9e7bbbef90fe6cf94cbf87767219440d31713daef6ad1e4a1cc720ce59fee4
cf7731e46bbba9ec1648908ea345030aa8f10ade10ffa" +
"3d2acfd480b0b11eadc4fb2b740596b204e911456cb2f35ad9993ab7dd6a48b35ba0c207625384b
b3c2ff24437810bd13c7ee96cd6f97f19ffd537ad182a3657" +
"b0e83d42fd6e2ebac6cbf5ea1bde97465b7cec6954ff5b5be049e59a49ea25ed6667dbace765401b
de12031e5cfabf2df7afb728d2a0b2a38b24d79bf23a313c" +
"40fe5adef97487641c6088dd8712c0c352708e474b02c08fd2d71b6d44f16d82f291ccd61c43a339
408379a8de54cfdbbae5e421e084112fbc17fb5561e084d1" +
"4149bf4bb06fd161878d8574f856867cff974d5898e161923d55bdac8699c9df6a220bcb6c800d3ae
7f107b8c4acab206d780aafaf6c2e2379de8c900700d9c9" +
"c87d464772514c5aa3e5f5bfc00fb54f2b74702838b4731c5ac8a070b50783e81dd97fa8d55c739d
026b607a2a78aba1bb79b1a7a3c22c78368672ac020061e6" +
"d9683d57d6989c6c6f08b8d5d74720f5cb25505fbe81d2bf53a68e972a54784705b20f83fd1ab5afff
30764ef89dba4465b56f48b325ab3810bf8dcbf4faa61f" +
"676e2043ac8540df9e3af4c0f51d816e89c09bf67253be45fc5f75f64be97f6c7dc0c6392af6fa8e75a
ab58eda976b36773cd37d315771400a2cb846fdef3d8a" +
"a15bce5dfda0379e526f87cf67767a2ab93d41c85b4ed016ed0a89d2f94737433a3ebade813def29
eaa18a1fb925fca7d08d1020f64caebc562cb4ad2fb241e2" +
"94923b2f2df5e6c4953c4b73be0f5568defe57ce49d16e2a205323e46cbb5a3e77fff1557671503bd
7b5de5320f1fb951fbe26400cfa854af2d12fab0215310c" +
"f070add34dc4565d1757d7e10a03e3bb73a607ed7e10861b1274ddee76183cf7e56c1de7162c805
c2dba0e0331d36f3a4e2019a2e0705ee2747ed1e52bc3a6fb" +
"3b061f784348204cdf8d643ff6c271fa72b56900edcc2f77201f3bd4fc296ad6534a7029ea66761bb9
a3589a1f6ef566504c70297b98fbb603214fed2e4b7ca1" +

"06e3f0e993118897fa641fb9722d4667fa98d07a6837e5ab2144e5ec1548a7dbca28c559f2a9a99d
54b8e55f56d0e59bcef1ac45e2046835b60579da0d2261e7" +
"30dab9009d138421c6458d146e870358b0b3fa20257e50b58f167c6b47edf7053513d58f33547d0
6ce52458baaa4dcf15f77b103565c66a81f183c827801b455" +
"b61b6287a46a37a96884075a7eada9ba7f0ddcc14654bf87a26d2e27a978b415257773796923a2
20e06b25af16fb5aaded9b2d081a4c64106df460ddce9c3b2a" +
"c8553e1521e501ad29a4b7f7681c9b60576a127087a5237c4c2bacf9b163dd590e63f2bc80e7f1e6
13773f87d0343130647104047 39d63363d204be7b14800c4" +
"d8c1b6a2a21da70223be51d281fee302ef806454f9d7d28244ba537c1d9f8f1bcc5d47038d986a8f9
5ca48437ffe94fd44a90bb03014a259112a97508adb3db4" +
"34f72a5268c1af6bc6d5801e579aab2228ca33600ebbf1a1959081c3a4ca99e444f97409f5e0ca47
79241c9aacad1f4ee7fb4369bd6ae076378e4f63000b9a5c" +
"849ba6e72e47e2454a44659149338ac0767cd25d8693c0d143e354bd600f1c1d3a44eeb024923e
a659060665d5cd9a4ca1ca86162819556535fd59b9fde90caa" +
"29920efe99479fe7e4b4e5371e13ccb43a1419cf023433239d840900d31bab37fabc3fd20e31bb7d
bcb3ae8df66f67e2844944bcf544b658364f9e3d0b6d84b4" +
"63ad4eb621644fd7d774b501407a1178814b15149345d551b474347188067db2ab4d7f4d1abd30
27133039e855e129f3f5649550da8c04fe2db57cb89bf1bf4f" +
"72eb35ccfe31afb92f6136d4c2a1c115b07b721b2da43151f11c356256230408979c5d9524371442
9e2c9500e7b043b20dac8a9763e5b487d1cbdb34ac379b9c" +
"6409454c79385b6e562459c4fdaad1b7f9297c1473e9b90fffe6d1c5390e241a187a4cefa2eb0cb0c
11f4ca6c5b961c18ceb57892295290dbc991692556bffa3" +
"b8c405cf285e6bcb8a90246ad0ac15122f4cf73adc129d23aa2240733404beb6d74bf698e5589288
a522573c774ce9f514b5d5c086382ea1dd4e89ff5facbf23" +
"d36bc3d203941e17747ede4b82820351f4df278ddb787ce8f6f1cc468ef953399efb072ce706e253f
1bab76444bb70be6443cd0db633e958dc57bd223e00418e" +
"915a7c2e4d94c0623f9788276480cdfe798387d35e2ea2d304d066aec7627794cdd4200a44208d6
c87f242c76e2d4a3f966b6fb96eaa63d892c1a177bef249b4" +
"fdd1a4c06c791f677dd9919f739ccf318bd77835330b0219786249c9c9736161dac771a838724f2d
d70afba46a6782fd27601cf8a7126ae95a66e526131a68d7" +
"7a809e513533ed8021eecdbc5851dfcf95e10f1bbe47b5c7f079275a1837836245266b66d89fab25
ac4bd6c1225560bea3259b67bf50a58ee056754d574da79e" +
"f9a1a0df3a5defed0f74fe74ce0bf65a04086f17e94a8451828c723c97932f26f9349f1a2c7866c617a
528602721de4f3cc8916bcfc66cdc106bafa26ea87a13" +
"94dfa37e396365fb7f92df007b46a50ff04c7f85bfa679230ebedf18c2fb876fc7098dd1c4328adf85de
71c31d94687a308053bfcdc158cfa7772170fbed63f5" +
"37dda41f65196dfacdd1186b5de0f3369d841ce6502192292d05a19ce7464f5bcab3015c721cac13
ddca561b92dc1ee25d3068dc1945a1b4e2bd1e6604c42e4c" +
"3c04b490f6365828957990007394557854a903e19feb06906e41cbc8766bf37bd7dece90f4cdc987
709b1129e84bfdc502543b5bfa887bf78553a5ec10ad68c5" +
"d10eff75f7aa495e7d934a55577fdc0aead31aee4522db0259d7d4ea8438a7996d80a787465a298
0457193d1c4bf1a0a1e01741d72e5fc4dfe58475c1c01026b" +

```
"5a3bc973b902280753e9c3226db9cc778e2506c56ee86ae85b4d54dbf05394107329b2d1ee5652
2cb1ce562fb1aa4e592199d9c29f64cc3ab1d757531e209eec" +
"aa138d8388169b5e28c45f5aba267eeaa57f69869f0b6855d82b0eafcde63895251f41e8e676a0ab
12ef3f569bb7de91b79fa46ad9637da01ca004f4d30259c1" +
"f5b00761f6ca9c17721a6718390624a10a11f7f52d7afb71ee5f8338828910e48f94a1347761abac8
7897b2dd0e23f1d325aec5031ef58f2972e8b402e05f8c1" +
"ae7053a90380a1ae0d4d06645548c23e13afa31aac8ff83b10f8341418af4114632f6406d6e33076
391696c9161d63c8bcfd1c625fc737f68198046212d1638a" +
"d2d2d42ff7029c1fcc682a046edc4d4f24862d82c600180b1e8f57ff6a3865dfe9274f9886d00efa523
a1b3b3757c4489200fec3dc5583854c955492336253dd" +
"767f2a60ce3d224afcff9cdc19e9b28830d33affda6af99942a8fe39562055f3e884fd6c1ebc1908ac1
59061f35e9b0da80434ce9673d9c6b87265170077c670" +
"743e37474d7605cd01c44af600f16d9ffaf24acf87fbe5ccf39bac41047a810d210051c87f06147a0b
b8f1427a406700483679638f1af23f1dafb7aa0c468669" +
"71c3a82f535c26cf6cb335e8e915fda393799d3dbe0e04b907ed3612d12ac95783a6876cd986d2a
13b82192532e02c250eaa42f891d2481655fa4494c723fe00" +
"87d224444245eb5b0eade5f741b025db1992a8ad0dce51b0c1af4a18a9e244f9f755891adf0f1917
9c7baa6c32bffc91e0b03c4ed3aaee1978b6a1f03b87ac6a" +
"fc3b9e7030bb212b17de198edfccde29d04224798c1204e47ea235f048724fac62d637d1ba0ee392
2048fcf79c746b6c0c036d882e3491fd72bad6e009c6403e" +
"55876f4d31330caa02aedd0b0c121c3c41e736853a08071f0dd4ddc7412db0bbe274a9ac2932552
bb37c40e72c2ef1d7cca8236942e480d709d3ea9d5ae0a1b7",
),
},
}
for i, tt := range tests {
cache := make([]uint32, tt.cacheSize/4)
generateCache(cache, tt.epoch, seedHash(tt.epoch*epochLength+1))

dataset := make([]uint32, tt.datasetSize/4)
generateDataset(dataset, tt.epoch, cache)

want := make([]uint32, tt.datasetSize/4)
prepare(want, tt.dataset)

if !reflect.DeepEqual(dataset, want) {
t.Errorf("dataset %d: content mismatch: have %x, want %x", i, dataset, want)
}
}
}

// Tests whether the hashimoto lookup works for both light as well as the full
```

```
// datasets.
func TestHashimoto(t *testing.T) {
// Create the verification cache and mining dataset
cache := make([]uint32, 1024/4)
generateCache(cache, 0, make([]byte, 32))

dataset := make([]uint32, 32*1024/4)
generateDataset(dataset, 0, cache)

// Create a block to verify
hash :=
hexutil.MustDecode("0xc9149cc0386e689d789a1c2f3d5d169a61a6218ed30e74414dc736e442ef
3d1f")
nonce := uint64(0)

wantDigest :=
hexutil.MustDecode("0xe4073cffaef931d37117cefd9afd27ea0f1cad6a981dd2605c4a1ac97c51980
0")
wantResult :=
hexutil.MustDecode("0xd3539235ee2e6f8db665c0a72169f55b7f6c605712330b778ec3944f0eb5a
557")

digest, result := hashimotoLight(32*1024, cache, hash, nonce)
if !bytes.Equal(digest, wantDigest) {
t.Errorf("light hashimoto digest mismatch: have %x, want %x", digest, wantDigest)
}
if !bytes.Equal(result, wantResult) {
t.Errorf("light hashimoto result mismatch: have %x, want %x", result, wantResult)
}
digest, result = hashimotoFull(dataset, hash, nonce)
if !bytes.Equal(digest, wantDigest) {
t.Errorf("full hashimoto digest mismatch: have %x, want %x", digest, wantDigest)
}
if !bytes.Equal(result, wantResult) {
t.Errorf("full hashimoto result mismatch: have %x, want %x", result, wantResult)
}
}

// Tests that caches generated on disk may be done concurrently.
func TestConcurrentDiskCacheGeneration(t *testing.T) {
// Create a temp folder to generate the caches into
cachedir, err := ioutil.TempDir("", "")
```

```
if err != nil {
t.Fatalf("Failed to create temporary cache dir: %v", err)
}
defer os.RemoveAll(cachedir)

// Define a heavy enough block, one from mainnet should do
block := types.NewBlockWithHeader(&types.Header{
Number:      big.NewInt(3311058),
ParentHash:
common.HexToHash("0xd783efa4d392943503f28438ad5830b2d5964696ffc285f338585e9fe0a37
a05"),
UncleHash:
common.HexToHash("0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d
49347"),
Coinbase:    common.HexToAddress("0xc0ea08a2d404d3172d2add29a45be56da40e2949"),
Root:
common.HexToHash("0x77d14e10470b5850332524f8cd6f69ad21f070ce92dca33ab2858300242e
f2f1"),
TxHash:
common.HexToHash("0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b
421"),
ReceiptHash:
common.HexToHash("0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b
421"),
Difficulty:  big.NewInt(167925187834220),
GasLimit:    big.NewInt(4015682),
GasUsed:     big.NewInt(0),
Time:        big.NewInt(1488928920),
Extra:       []byte("www.bw.com"),
MixDigest:
common.HexToHash("0x3e140b0784516af5e5ec6730f2fb20cca22f32be399b9e4ad77d32541f798
cd0"),
Nonce:       types.EncodeNonce(0xf400cd0006070c49),
})
// Simulate multiple processes sharing the same datadir
var pend sync.WaitGroup

for i := 0; i < 3; i++ {
pend.Add(1)

go func(idx int) {
defer pend.Done()
```

```
ethash := New(cachedir, 0, 1, "", 0, 0)
if err := ethash.VerifySeal(nil, block.Header()); err != nil {
t.Errorf("proc %d: block verification failed: %v", idx, err)
}
}(i)
}
pend.Wait()
}

// Benchmarks the cache generation performance.
func BenchmarkCacheGeneration(b *testing.B) {
for i := 0; i < b.N; i++ {
cache := make([]uint32, cacheSize(1)/4)
generateCache(cache, 0, make([]byte, 32))
}
}

// Benchmarks the dataset (small) generation performance.
func BenchmarkSmallDatasetGeneration(b *testing.B) {
cache := make([]uint32, 65536/4)
generateCache(cache, 0, make([]byte, 32))

b.ResetTimer()
for i := 0; i < b.N; i++ {
dataset := make([]uint32, 32*65536/4)
generateDataset(dataset, 0, cache)
}
}

// Benchmarks the light verification performance.
func BenchmarkHashimotoLight(b *testing.B) {
cache := make([]uint32, cacheSize(1)/4)
generateCache(cache, 0, make([]byte, 32))

hash :=
hexutil.MustDecode("0xc9149cc0386e689d789a1c2f3d5d169a61a6218ed30e74414dc736e442ef
3d1f")

b.ResetTimer()
for i := 0; i < b.N; i++ {
hashimotoLight(datasetSize(1), cache, hash, 0)
```

```
}
}

// Benchmarks the full (small) verification performance.
func BenchmarkHashimotoFullSmall(b *testing.B) {
cache := make([]uint32, 65536/4)
generateCache(cache, 0, make([]byte, 32))

dataset := make([]uint32, 32*65536/4)
generateDataset(dataset, 0, cache)

hash :=
hexutil.MustDecode("0xc9149cc0386e689d789a1c2f3d5d169a61a6218ed30e74414dc736e442ef
3d1f")

b.ResetTimer()
for i := 0; i < b.N; i++ {
hashimotoFull(dataset, hash, 0)
}
}

33:F:\git\coin\ethereum\go-ethereum\consensus\ethash\consensus.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ethash

import (
"bytes"
"errors"
"fmt"
"math/big"
"runtime"
"time"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/math"
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/consensus/misc"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/params"
set "gopkg.in/fatih/set.v0"
```

```
)

// Ethash proof-of-work protocol constants.
var (
blockReward *big.Int = big.NewInt(5e+18) // Block reward in wei for successfully mining a block
maxUncles            = 2                 // Maximum number of uncles allowed in a single block
)

// Various error messages to mark blocks invalid. These should be private to
// prevent engine specific errors from being referenced in the remainder of the
// codebase, inherently breaking if the engine is swapped out. Please put common
// error types into the consensus package.
var (
errLargeBlockTime    = errors.New("timestamp too big")
errZeroBlockTime     = errors.New("timestamp equals parent's")
errTooManyUncles     = errors.New("too many uncles")
errDuplicateUncle    = errors.New("duplicate uncle")
errUncleIsAncestor   = errors.New("uncle is ancestor")
errDanglingUncle     = errors.New("uncle's parent is not ancestor")
errNonceOutOfRange   = errors.New("nonce out of range")
errInvalidDifficulty = errors.New("non-positive difficulty")
errInvalidMixDigest  = errors.New("invalid mix digest")
errInvalidPoW        = errors.New("invalid proof-of-work")
)

// Author implements consensus.Engine, returning the header's coinbase as the
// proof-of-work verified author of the block.
func (ethash *Ethash) Author(header *types.Header) (common.Address, error) {
return header.Coinbase, nil
}

// VerifyHeader checks whether a header conforms to the consensus rules of the
// stock Ethereum ethash engine.
func (ethash *Ethash) VerifyHeader(chain consensus.ChainReader, header *types.Header, seal
bool) error {
// If we're running a full engine faking, accept any input as valid
if ethash.fakeFull {
return nil
}
// Short circuit if the header is known, or it's parent not
number := header.Number.Uint64()
if chain.GetHeader(header.Hash(), number) != nil {
```

```
return nil
}
parent := chain.GetHeader(header.ParentHash, number-1)
if parent == nil {
return consensus.ErrUnknownAncestor
}
// Sanity checks passed, do a proper verification
return ethash.verifyHeader(chain, header, parent, false, seal)
}

// VerifyHeaders is similar to VerifyHeader, but verifies a batch of headers
// concurrently. The method returns a quit channel to abort the operations and
// a results channel to retrieve the async verifications.
func (ethash *Ethash) VerifyHeaders(chain consensus.ChainReader, headers []*types.Header,
seals []bool) (chan<- struct{}, <-chan error) {
// If we're running a full engine faking, accept any input as valid
if ethash.fakeFull || len(headers) == 0 {
abort, results := make(chan struct{}), make(chan error, len(headers))
for i := 0; i < len(headers); i++ {
results <- nil
}
return abort, results
}

// Spawn as many workers as allowed threads
workers := runtime.GOMAXPROCS(0)
if len(headers) < workers {
workers = len(headers)
}

// Create a task channel and spawn the verifiers
var (
inputs = make(chan int)
done   = make(chan int, workers)
errors = make([]error, len(headers))
abort  = make(chan struct{})
)
for i := 0; i < workers; i++ {
go func() {
for index := range inputs {
errors[index] = ethash.verifyHeaderWorker(chain, headers, seals, index)
done <- index
```

```
}
}()
}

errorsOut := make(chan error, len(headers))
go func() {
defer close(inputs)
var (
in, out = 0, 0
checked = make([]bool, len(headers))
inputs  = inputs
)
for {
select {
case inputs <- in:
if in++; in == len(headers) {
// Reached end of headers. Stop sending to workers.
inputs = nil
}
case index := <-done:
for checked[index] = true; checked[out]; out++ {
errorsOut <- errors[out]
if out == len(headers)-1 {
return
}
}
case <-abort:
return
}
}
}()
return abort, errorsOut
}

func (ethash *Ethash) verifyHeaderWorker(chain consensus.ChainReader, headers
[]*types.Header, seals []bool, index int) error {
var parent *types.Header
if index == 0 {
parent = chain.GetHeader(headers[0].ParentHash, headers[0].Number.Uint64()-1)
} else if headers[index-1].Hash() == headers[index].ParentHash {
parent = headers[index-1]
}
```

```
if parent == nil {
return consensus.ErrUnknownAncestor
}
if chain.GetHeader(headers[index].Hash(), headers[index].Number.Uint64()) != nil {
return nil // known block
}
return ethash.verifyHeader(chain, headers[index], parent, false, seals[index])
}

// VerifyUncles verifies that the given block's uncles conform to the consensus
// rules of the stock Ethereum ethash engine.
func (ethash *Ethash) VerifyUncles(chain consensus.ChainReader, block *types.Block) error {
// If we're running a full engine faking, accept any input as valid
if ethash.fakeFull {
return nil
}
// Verify that there are at most 2 uncles included in this block
if len(block.Uncles()) > maxUncles {
return errTooManyUncles
}
// Gather the set of past uncles and ancestors
uncles, ancestors := set.New(), make(map[common.Hash]*types.Header)

number, parent := block.NumberU64()-1, block.ParentHash()
for i := 0; i < 7; i++ {
ancestor := chain.GetBlock(parent, number)
if ancestor == nil {
break
}
ancestors[ancestor.Hash()] = ancestor.Header()
for _, uncle := range ancestor.Uncles() {
uncles.Add(uncle.Hash())
}
parent, number = ancestor.ParentHash(), number-1
}
ancestors[block.Hash()] = block.Header()
uncles.Add(block.Hash())

// Verify each of the uncles that it's recent, but not an ancestor
for _, uncle := range block.Uncles() {
// Make sure every uncle is rewarded only once
hash := uncle.Hash()
```

```
if uncles.Has(hash) {
return errDuplicateUncle
}
uncles.Add(hash)

// Make sure the uncle has a valid ancestry
if ancestors[hash] != nil {
return errUncleIsAncestor
}
if ancestors[uncle.ParentHash] == nil || uncle.ParentHash == block.ParentHash() {
return errDanglingUncle
}
if err := ethash.verifyHeader(chain, uncle, ancestors[uncle.ParentHash], true, true); err != nil {
return err
}
}
return nil
}

// verifyHeader checks whether a header conforms to the consensus rules of the
// stock Ethereum ethash engine.
// See YP section 4.3.4. "Block Header Validity"
func (ethash *Ethash) verifyHeader(chain consensus.ChainReader, header, parent *types.Header,
uncle bool, seal bool) error {
// Ensure that the header's extra-data section is of a reasonable size
if uint64(len(header.Extra)) > params.MaximumExtraDataSize {
return fmt.Errorf("extra-data too long: %d > %d", len(header.Extra),
params.MaximumExtraDataSize)
}
// Verify the header's timestamp
if uncle {
if header.Time.Cmp(math.MaxBig256) > 0 {
return errLargeBlockTime
}
} else {
if header.Time.Cmp(big.NewInt(time.Now().Unix())) > 0 {
return consensus.ErrFutureBlock
}
}
if header.Time.Cmp(parent.Time) <= 0 {
return errZeroBlockTime
}
```

```
// Verify the block's difficulty based in it's timestamp and parent's difficulty
expected := CalcDifficulty(chain.Config(), header.Time.Uint64(), parent)
if expected.Cmp(header.Difficulty) != 0 {
return fmt.Errorf("invalid difficulty: have %v, want %v", header.Difficulty, expected)
}
// Verify that the gas limit is <= 2^63-1
if header.GasLimit.Cmp(math.MaxBig63) > 0 {
return fmt.Errorf("invalid gasLimit: have %v, max %v", header.GasLimit, math.MaxBig63)
}
// Verify that the gasUsed is <= gasLimit
if header.GasUsed.Cmp(header.GasLimit) > 0 {
return fmt.Errorf("invalid gasUsed: have %v, gasLimit %v", header.GasUsed, header.GasLimit)
}

// Verify that the gas limit remains within allowed bounds
diff := new(big.Int).Set(parent.GasLimit)
diff = diff.Sub(diff, header.GasLimit)
diff.Abs(diff)

limit := new(big.Int).Set(parent.GasLimit)
limit = limit.Div(limit, params.GasLimitBoundDivisor)

if diff.Cmp(limit) >= 0 || header.GasLimit.Cmp(params.MinGasLimit) < 0 {
return fmt.Errorf("invalid gas limit: have %v, want %v += %v", header.GasLimit, parent.GasLimit,
limit)
}
// Verify that the block number is parent's +1
if diff := new(big.Int).Sub(header.Number, parent.Number); diff.Cmp(big.NewInt(1)) != 0 {
return consensus.ErrInvalidNumber
}
// Verify the engine specific seal securing the block
if seal {
if err := ethash.VerifySeal(chain, header); err != nil {
return err
}
}
// If all checks passed, validate any special fields for hard forks
if err := misc.VerifyDAOHeaderExtraData(chain.Config(), header); err != nil {
return err
}
if err := misc.VerifyForkHashes(chain.Config(), header, uncle); err != nil {
return err
```

```
}
return nil
}

// CalcDifficulty is the difficulty adjustment algorithm. It returns
// the difficulty that a new block should have when created at time
// given the parent block's time and difficulty.
// TODO (karalabe): Move the chain maker into this package and make this private!
func CalcDifficulty(config *params.ChainConfig, time uint64, parent *types.Header) *big.Int {
next := new(big.Int).Add(parent.Number, common.Big1)
switch {
case config.IsHomestead(next):
return calcDifficultyHomestead(time, parent)
default:
return calcDifficultyFrontier(time, parent)
}
}

// Some weird constants to avoid constant memory allocs for them.
var (
expDiffPeriod = big.NewInt(100000)
big10         = big.NewInt(10)
bigMinus99    = big.NewInt(-99)
)

// calcDifficultyHomestead is the difficulty adjustment algorithm. It returns
// the difficulty that a new block should have when created at time given the
// parent block's time and difficulty. The calculation uses the Homestead rules.
func calcDifficultyHomestead(time uint64, parent *types.Header) *big.Int {
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2.mediawiki
// algorithm:
// diff = (parent_diff +
//         (parent_diff / 2048 * max(1 - (block_timestamp - parent_timestamp) // 10, -99))
//        ) + 2^(periodCount - 2)

bigTime := new(big.Int).SetUint64(time)
bigParentTime := new(big.Int).Set(parent.Time)

// holds intermediate values to make the algo easier to read & audit
x := new(big.Int)
y := new(big.Int)
```

```
// 1 - (block_timestamp -parent_timestamp) // 10
x.Sub(bigTime, bigParentTime)
x.Div(x, big10)
x.Sub(common.Big1, x)

// max(1 - (block_timestamp - parent_timestamp) // 10, -99)))
if x.Cmp(bigMinus99) < 0 {
x.Set(bigMinus99)
}
// (parent_diff + parent_diff // 2048 * max(1 - (block_timestamp - parent_timestamp) // 10, -99))
y.Div(parent.Difficulty, params.DifficultyBoundDivisor)
x.Mul(y, x)
x.Add(parent.Difficulty, x)

// minimum difficulty can ever be (before exponential factor)
if x.Cmp(params.MinimumDifficulty) < 0 {
x.Set(params.MinimumDifficulty)
}
// for the exponential factor
periodCount := new(big.Int).Add(parent.Number, common.Big1)
periodCount.Div(periodCount, expDiffPeriod)

// the exponential factor, commonly referred to as "the bomb"
// diff = diff + 2^(periodCount - 2)
if periodCount.Cmp(common.Big1) > 0 {
y.Sub(periodCount, common.Big2)
y.Exp(common.Big2, y, nil)
x.Add(x, y)
}
return x
}

// calcDifficultyFrontier is the difficulty adjustment algorithm. It returns the
// difficulty that a new block should have when created at time given the parent
// block's time and difficulty. The calculation uses the Frontier rules.
func calcDifficultyFrontier(time uint64, parent *types.Header) *big.Int {
diff := new(big.Int)
adjust := new(big.Int).Div(parent.Difficulty, params.DifficultyBoundDivisor)
bigTime := new(big.Int)
bigParentTime := new(big.Int)

bigTime.SetUint64(time)
```

```
bigParentTime.Set(parent.Time)

if bigTime.Sub(bigTime, bigParentTime).Cmp(params.DurationLimit) < 0 {
diff.Add(parent.Difficulty, adjust)
} else {
diff.Sub(parent.Difficulty, adjust)
}
if diff.Cmp(params.MinimumDifficulty) < 0 {
diff.Set(params.MinimumDifficulty)
}

periodCount := new(big.Int).Add(parent.Number, common.Big1)
periodCount.Div(periodCount, expDiffPeriod)
if periodCount.Cmp(common.Big1) > 0 {
// diff = diff + 2^(periodCount - 2)
expDiff := periodCount.Sub(periodCount, common.Big2)
expDiff.Exp(common.Big2, expDiff, nil)
diff.Add(diff, expDiff)
diff = math.BigMax(diff, params.MinimumDifficulty)
}
return diff
}

// VerifySeal implements consensus.Engine, checking whether the given block satisfies
// the PoW difficulty requirements.
func (ethash *Ethash) VerifySeal(chain consensus.ChainReader, header *types.Header) error {
// If we're running a fake PoW, accept any seal as valid
if ethash.fakeMode {
time.Sleep(ethash.fakeDelay)
if ethash.fakeFail == header.Number.Uint64() {
return errInvalidPoW
}
return nil
}
// If we're running a shared PoW, delegate verification to it
if ethash.shared != nil {
return ethash.shared.VerifySeal(chain, header)
}
// Sanity check that the block number is below the lookup table size (60M blocks)
number := header.Number.Uint64()
if number/epochLength >= uint64(len(cacheSizes)) {
// Go < 1.7 cannot calculate new cache/dataset sizes (no fast prime check)
```

```
return errNonceOutOfRange
}
// Ensure that we have a valid difficulty for the block
if header.Difficulty.Sign() <= 0 {
return errInvalidDifficulty
}
// Recompute the digest and PoW value and verify against the header
cache := ethash.cache(number)

size := datasetSize(number)
if ethash.tester {
size = 32 * 1024
}
digest, result := hashimotoLight(size, cache, header.HashNoNonce().Bytes(),
header.Nonce.Uint64())
if !bytes.Equal(header.MixDigest[:], digest) {
return errInvalidMixDigest
}
target := new(big.Int).Div(maxUint256, header.Difficulty)
if new(big.Int).SetBytes(result).Cmp(target) > 0 {
return errInvalidPoW
}
return nil
}

// Prepare implements consensus.Engine, initializing the difficulty field of a
// header to conform to the ethash protocol. The changes are done inline.
func (ethash *Ethash) Prepare(chain consensus.ChainReader, header *types.Header) error {
parent := chain.GetHeader(header.ParentHash, header.Number.Uint64()-1)
if parent == nil {
return consensus.ErrUnknownAncestor
}
header.Difficulty = CalcDifficulty(chain.Config(), header.Time.Uint64(), parent)

return nil
}

// Finalize implements consensus.Engine, accumulating the block and uncle rewards,
// setting the final state and assembling the block.
func (ethash *Ethash) Finalize(chain consensus.ChainReader, header *types.Header, state
*state.StateDB, txs []*types.Transaction, uncles []*types.Header, receipts []*types.Receipt)
(*types.Block, error) {
```

```
// Accumulate any block and uncle rewards and commit the final state root
AccumulateRewards(state, header, uncles)
header.Root = state.IntermediateRoot(chain.Config().IsEIP158(header.Number))

// Header seems complete, assemble into a block and return
return types.NewBlock(header, txs, uncles, receipts), nil
}

// Some weird constants to avoid constant memory allocs for them.
var (
big8  = big.NewInt(8)
big32 = big.NewInt(32)
)

// AccumulateRewards credits the coinbase of the given block with the mining
// reward. The total reward consists of the static block reward and rewards for
// included uncles. The coinbase of each uncle block is also rewarded.
// TODO (karalabe): Move the chain maker into this package and make this private!
func AccumulateRewards(state *state.StateDB, header *types.Header, uncles []*types.Header) {
reward := new(big.Int).Set(blockReward)
r := new(big.Int)
for _, uncle := range uncles {
r.Add(uncle.Number, big8)
r.Sub(r, header.Number)
r.Mul(r, blockReward)
r.Div(r, big8)
state.AddBalance(uncle.Coinbase, r)

r.Div(blockReward, big32)
reward.Add(reward, r)
}
state.AddBalance(header.Coinbase, reward)
}

34:F:\git\coin\ethereum\go-ethereum\consensus\ethash\consensus_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ethash

import (
"encoding/json"
"math/big"
```

```
"os"
"testing"

"github.com/ethereum/go-ethereum/common/math"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/params"
)

type diffTest struct {
ParentTimestamp    uint64
ParentDifficulty   *big.Int
CurrentTimestamp   uint64
CurrentBlocknumber *big.Int
CurrentDifficulty  *big.Int
}

func (d *diffTest) UnmarshalJSON(b []byte) (err error) {
var ext struct {
ParentTimestamp    string
ParentDifficulty   string
CurrentTimestamp   string
CurrentBlocknumber string
CurrentDifficulty  string
}
if err := json.Unmarshal(b, &ext); err != nil {
return err
}

d.ParentTimestamp = math.MustParseUint64(ext.ParentTimestamp)
d.ParentDifficulty = math.MustParseBig256(ext.ParentDifficulty)
d.CurrentTimestamp = math.MustParseUint64(ext.CurrentTimestamp)
d.CurrentBlocknumber = math.MustParseBig256(ext.CurrentBlocknumber)
d.CurrentDifficulty = math.MustParseBig256(ext.CurrentDifficulty)

return nil
}

func TestCalcDifficulty(t *testing.T) {
file, err := os.Open("../../tests/files/BasicTests/difficulty.json")
if err != nil {
t.Fatal(err)
}
```

```
defer file.Close()

tests := make(map[string]diffTest)
err = json.NewDecoder(file).Decode(&tests)
if err != nil {
t.Fatal(err)
}

config := &params.ChainConfig{HomesteadBlock: big.NewInt(1150000)}
for name, test := range tests {
number := new(big.Int).Sub(test.CurrentBlocknumber, big.NewInt(1))
diff := CalcDifficulty(config, test.CurrentTimestamp, &types.Header{
Number:     number,
Time:       new(big.Int).SetUint64(test.ParentTimestamp),
Difficulty: test.ParentDifficulty,
})
if diff.Cmp(test.CurrentDifficulty) != 0 {
t.Error(name, "failed. Expected", test.CurrentDifficulty, "and calculated", diff)
}
}
}

35:F:\git\coin\ethereum\go-ethereum\consensus\ethash\ethash.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package ethash implements the ethash proof-of-work consensus engine.
package ethash

import (
"errors"
"fmt"
"math"
"math/big"
"math/rand"
"os"
"path/filepath"
"reflect"
"strconv"
"sync"
"time"
"unsafe"
```

```
mmap "github.com/edsrzf/mmap-go"
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/rpc"
metrics "github.com/rcrowley/go-metrics"
)

var ErrInvalidDumpMagic = errors.New("invalid dump magic")

var (
// maxUint256 is a big integer representing 2^256-1
maxUint256 = new(big.Int).Exp(big.NewInt(2), big.NewInt(256), big.NewInt(0))

// sharedEthash is a full instance that can be shared between multiple users.
sharedEthash = New("", 3, 0, "", 1, 0)

// algorithmRevision is the data structure version used for file naming.
algorithmRevision = 23

// dumpMagic is a dataset dump header to sanity check a data dump.
dumpMagic = []uint32{0xbaddcafe, 0xfee1dead}
)

// isLittleEndian returns whether the local system is running in little or big
// endian byte order.
func isLittleEndian() bool {
n := uint32(0x01020304)
return *(*byte)(unsafe.Pointer(&n)) == 0x04
}

// memoryMap tries to memory map a file of uint32s for read only access.
func memoryMap(path string) (*os.File, mmap.MMap, []uint32, error) {
file, err := os.OpenFile(path, os.O_RDONLY, 0644)
if err != nil {
return nil, nil, nil, err
}
mem, buffer, err := memoryMapFile(file, false)
if err != nil {
file.Close()
return nil, nil, nil, err
}
for i, magic := range dumpMagic {
```

```
if buffer[i] != magic {
mem.Unmap()
file.Close()
return nil, nil, nil, ErrInvalidDumpMagic
}
}
return file, mem, buffer[len(dumpMagic):], err
}

// memoryMapFile tries to memory map an already opened file descriptor.
func memoryMapFile(file *os.File, write bool) (mmap.MMap, []uint32, error) {
// Try to memory map the file
flag := mmap.RDONLY
if write {
flag = mmap.RDWR
}
mem, err := mmap.Map(file, flag, 0)
if err != nil {
return nil, nil, err
}
// Yay, we managed to memory map the file, here be dragons
header := *(*reflect.SliceHeader)(unsafe.Pointer(&mem))
header.Len /= 4
header.Cap /= 4

return mem, *(*[]uint32)(unsafe.Pointer(&header)), nil
}

// memoryMapAndGenerate tries to memory map a temporary file of uint32s for write
// access, fill it with the data from a generator and then move it into the final
// path requested.
func memoryMapAndGenerate(path string, size uint64, generator func(buffer []uint32)) (*os.File,
mmap.MMap, []uint32, error) {
// Ensure the data folder exists
if err := os.MkdirAll(filepath.Dir(path), 0755); err != nil {
return nil, nil, nil, err
}
// Create a huge temporary empty file to fill with data
temp := path + "." + strconv.Itoa(rand.Int())

dump, err := os.Create(temp)
if err != nil {
```

```
return nil, nil, nil, err
}
if err = dump.Truncate(int64(len(dumpMagic))*4 + int64(size)); err != nil {
return nil, nil, nil, err
}
// Memory map the file for writing and fill it with the generator
mem, buffer, err := memoryMapFile(dump, true)
if err != nil {
dump.Close()
return nil, nil, nil, err
}
copy(buffer, dumpMagic)

data := buffer[len(dumpMagic):]
generator(data)

if err := mem.Unmap(); err != nil {
return nil, nil, nil, err
}
if err := dump.Close(); err != nil {
return nil, nil, nil, err
}
if err := os.Rename(temp, path); err != nil {
return nil, nil, nil, err
}
return memoryMap(path)
}

// cache wraps an ethash cache with some metadata to allow easier concurrent use.
type cache struct {
epoch uint64 // Epoch for which this cache is relevant

dump *os.File  // File descriptor of the memory mapped cache
mmap mmap.MMap // Memory map itself to unmap before releasing

cache []uint32   // The actual cache data content (may be memory mapped)
used  time.Time  // Timestamp of the last use for smarter eviction
once  sync.Once  // Ensures the cache is generated only once
lock  sync.Mutex // Ensures thread safety for updating the usage time
}

// generate ensures that the cache content is generated before use.
```

```
func (c *cache) generate(dir string, limit int, test bool) {
c.once.Do(func() {
// If we have a testing cache, generate and return
if test {
c.cache = make([]uint32, 1024/4)
generateCache(c.cache, c.epoch, seedHash(c.epoch*epochLength+1))
return
}
// If we don't store anything on disk, generate and return
size := cacheSize(c.epoch*epochLength + 1)
seed := seedHash(c.epoch*epochLength + 1)

if dir == "" {
c.cache = make([]uint32, size/4)
generateCache(c.cache, c.epoch, seed)
return
}
// Disk storage is needed, this will get fancy
var endian string
if !isLittleEndian() {
endian = ".be"
}
path := filepath.Join(dir, fmt.Sprintf("cache-R%d-%x%s", algorithmRevision, seed[:8], endian))
logger := log.New("epoch", c.epoch)

// Try to load the file from disk and memory map it
var err error
c.dump, c.mmap, c.cache, err = memoryMap(path)
if err == nil {
logger.Debug("Loaded old ethash cache from disk")
return
}
logger.Debug("Failed to load old ethash cache", "err", err)

// No previous cache available, create a new cache file to fill
c.dump, c.mmap, c.cache, err = memoryMapAndGenerate(path, size, func(buffer []uint32) {
generateCache(buffer, c.epoch, seed) })
if err != nil {
logger.Error("Failed to generate mapped ethash cache", "err", err)

c.cache = make([]uint32, size/4)
generateCache(c.cache, c.epoch, seed)
```

```
}
// Iterate over all previous instances and delete old ones
for ep := int(c.epoch) - limit; ep >= 0; ep-- {
seed := seedHash(uint64(ep)*epochLength + 1)
path := filepath.Join(dir, fmt.Sprintf("cache-R%d-%x%s", algorithmRevision, seed[:8], endian))
os.Remove(path)
}
})
}

// release closes any file handlers and memory maps open.
func (c *cache) release() {
if c.mmap != nil {
c.mmap.Unmap()
c.mmap = nil
}
if c.dump != nil {
c.dump.Close()
c.dump = nil
}
}

// dataset wraps an ethash dataset with some metadata to allow easier concurrent use.
type dataset struct {
epoch uint64 // Epoch for which this cache is relevant

dump *os.File  // File descriptor of the memory mapped cache
mmap mmap.MMap // Memory map itself to unmap before releasing

dataset []uint32   // The actual cache data content
used    time.Time  // Timestamp of the last use for smarter eviction
once    sync.Once  // Ensures the cache is generated only once
lock    sync.Mutex // Ensures thread safety for updating the usage time
}

// generate ensures that the dataset content is generated before use.
func (d *dataset) generate(dir string, limit int, test bool) {
d.once.Do(func() {
// If we have a testing dataset, generate and return
if test {
cache := make([]uint32, 1024/4)
generateCache(cache, d.epoch, seedHash(d.epoch*epochLength+1))
```

```
d.dataset = make([]uint32, 32*1024/4)
generateDataset(d.dataset, d.epoch, cache)

return
}
// If we don't store anything on disk, generate and return
csize := cacheSize(d.epoch*epochLength + 1)
dsize := datasetSize(d.epoch*epochLength + 1)
seed := seedHash(d.epoch*epochLength + 1)

if dir == "" {
cache := make([]uint32, csize/4)
generateCache(cache, d.epoch, seed)

d.dataset = make([]uint32, dsize/4)
generateDataset(d.dataset, d.epoch, cache)
}
// Disk storage is needed, this will get fancy
var endian string
if !isLittleEndian() {
endian = ".be"
}
path := filepath.Join(dir, fmt.Sprintf("full-R%d-%x%s", algorithmRevision, seed[:8], endian))
logger := log.New("epoch", d.epoch)

// Try to load the file from disk and memory map it
var err error
d.dump, d.mmap, d.dataset, err = memoryMap(path)
if err == nil {
logger.Debug("Loaded old ethash dataset from disk")
return
}
logger.Debug("Failed to load old ethash dataset", "err", err)

// No previous dataset available, create a new dataset file to fill
cache := make([]uint32, csize/4)
generateCache(cache, d.epoch, seed)

d.dump, d.mmap, d.dataset, err = memoryMapAndGenerate(path, dsize, func(buffer []uint32) {
generateDataset(buffer, d.epoch, cache) })
if err != nil {
```

```
logger.Error("Failed to generate mapped ethash dataset", "err", err)

d.dataset = make([]uint32, dsize/2)
generateDataset(d.dataset, d.epoch, cache)
}
// Iterate over all previous instances and delete old ones
for ep := int(d.epoch) - limit; ep >= 0; ep-- {
seed := seedHash(uint64(ep)*epochLength + 1)
path := filepath.Join(dir, fmt.Sprintf("full-R%d-%x%s", algorithmRevision, seed[:8], endian))
os.Remove(path)
}
})
}

// release closes any file handlers and memory maps open.
func (d *dataset) release() {
if d.mmap != nil {
d.mmap.Unmap()
d.mmap = nil
}
if d.dump != nil {
d.dump.Close()
d.dump = nil
}
}

// MakeCache generates a new ethash cache and optionally stores it to disk.
func MakeCache(block uint64, dir string) {
c := cache{epoch: block / epochLength}
c.generate(dir, math.MaxInt32, false)
c.release()
}

// MakeDataset generates a new ethash dataset and optionally stores it to disk.
func MakeDataset(block uint64, dir string) {
d := dataset{epoch: block / epochLength}
d.generate(dir, math.MaxInt32, false)
d.release()
}

// Ethash is a consensus engine based on proot-of-work implementing the ethash
// algorithm.
```

```
type Ethash struct {
cachedir     string // Data directory to store the verification caches
cachesinmem  int    // Number of caches to keep in memory
cachesondisk int    // Number of caches to keep on disk
dagdir       string // Data directory to store full mining datasets
dagsinmem    int    // Number of mining datasets to keep in memory
dagsondisk   int    // Number of mining datasets to keep on disk

caches   map[uint64]*cache   // In memory caches to avoid regenerating too often
fcache   *cache              // Pre-generated cache for the estimated future epoch
datasets map[uint64]*dataset // In memory datasets to avoid regenerating too often
fdataset *dataset            // Pre-generated dataset for the estimated future epoch

// Mining related fields
rand     *rand.Rand    // Properly seeded random source for nonces
threads  int           // Number of threads to mine on if mining
update   chan struct{} // Notification channel to update mining parameters
hashrate metrics.Meter // Meter tracking the average hashrate

// The fields below are hooks for testing
tester    bool         // Flag whether to use a smaller test dataset
shared    *Ethash      // Shared PoW verifier to avoid cache regeneration
fakeMode  bool         // Flag whether to disable PoW checking
fakeFull  bool         // Flag whether to disable all consensus rules
fakeFail  uint64       // Block number which fails PoW check even in fake mode
fakeDelay time.Duration // Time delay to sleep for before returning from verify

lock sync.Mutex // Ensures thread safety for the in-memory caches and mining fields
}

// New creates a full sized ethash PoW scheme.
func New(cachedir string, cachesinmem, cachesondisk int, dagdir string, dagsinmem, dagsondisk
int) *Ethash {
if cachesinmem <= 0 {
log.Warn("One ethash cache must always be in memory", "requested", cachesinmem)
cachesinmem = 1
}
if cachedir != "" && cachesondisk > 0 {
log.Info("Disk storage enabled for ethash caches", "dir", cachedir, "count", cachesondisk)
}
if dagdir != "" && dagsondisk > 0 {
log.Info("Disk storage enabled for ethash DAGs", "dir", dagdir, "count", dagsondisk)
```

```
}
return &Ethash{
cachedir:     cachedir,
cachesinmem:  cachesinmem,
cachesondisk: cachesondisk,
dagdir:       dagdir,
dagsinmem:    dagsinmem,
dagsondisk:   dagsondisk,
caches:       make(map[uint64]*cache),
datasets:     make(map[uint64]*dataset),
update:       make(chan struct{}),
hashrate:     metrics.NewMeter(),
}
}

// NewTester creates a small sized ethash PoW scheme useful only for testing
// purposes.
func NewTester() *Ethash {
return &Ethash{
cachesinmem: 1,
caches:      make(map[uint64]*cache),
datasets:    make(map[uint64]*dataset),
tester:      true,
update:      make(chan struct{}),
hashrate:    metrics.NewMeter(),
}
}

// NewFaker creates a ethash consensus engine with a fake PoW scheme that accepts
// all blocks' seal as valid, though they still have to conform to the Ethereum
// consensus rules.
func NewFaker() *Ethash {
return &Ethash{fakeMode: true}
}

// NewFakeFailer creates a ethash consensus engine with a fake PoW scheme that
// accepts all blocks as valid apart from the single one specified, though they
// still have to conform to the Ethereum consensus rules.
func NewFakeFailer(fail uint64) *Ethash {
return &Ethash{fakeMode: true, fakeFail: fail}
}
```

```
// NewFakeDelayer creates a ethash consensus engine with a fake PoW scheme that
// accepts all blocks as valid, but delays verifications by some time, though
// they still have to conform to the Ethereum consensus rules.
func NewFakeDelayer(delay time.Duration) *Ethash {
return &Ethash{fakeMode: true, fakeDelay: delay}
}

// NewFullFaker creates an ethash consensus engine with a full fake scheme that
// accepts all blocks as valid, without checking any consensus rules whatsoever.
func NewFullFaker() *Ethash {
return &Ethash{fakeMode: true, fakeFull: true}
}

// NewShared creates a full sized ethash PoW shared between all requesters running
// in the same process.
func NewShared() *Ethash {
return &Ethash{shared: sharedEthash}
}

// cache tries to retrieve a verification cache for the specified block number
// by first checking against a list of in-memory caches, then against caches
// stored on disk, and finally generating one if none can be found.
func (ethash *Ethash) cache(block uint64) []uint32 {
epoch := block / epochLength

// If we have a PoW for that epoch, use that
ethash.lock.Lock()

current, future := ethash.caches[epoch], (*cache)(nil)
if current == nil {
// No in-memory cache, evict the oldest if the cache limit was reached
for len(ethash.caches) > 0 && len(ethash.caches) >= ethash.cachesinmem {
var evict *cache
for _, cache := range ethash.caches {
if evict == nil || evict.used.After(cache.used) {
evict = cache
}
}
delete(ethash.caches, evict.epoch)
evict.release()

log.Trace("Evicted ethash cache", "epoch", evict.epoch, "used", evict.used)
```

```
}
// If we have the new cache pre-generated, use that, otherwise create a new one
if ethash.fcache != nil && ethash.fcache.epoch == epoch {
log.Trace("Using pre-generated cache", "epoch", epoch)
current, ethash.fcache = ethash.fcache, nil
} else {
log.Trace("Requiring new ethash cache", "epoch", epoch)
current = &cache{epoch: epoch}
}
ethash.caches[epoch] = current

// If we just used up the future cache, or need a refresh, regenerate
if ethash.fcache == nil || ethash.fcache.epoch <= epoch {
if ethash.fcache != nil {
ethash.fcache.release()
}
log.Trace("Requiring new future ethash cache", "epoch", epoch+1)
future = &cache{epoch: epoch + 1}
ethash.fcache = future
}
// New current cache, set its initial timestamp
current.used = time.Now()
}
ethash.lock.Unlock()

// Wait for generation finish, bump the timestamp and finalize the cache
current.generate(ethash.cachedir, ethash.cachesondisk, ethash.tester)

current.lock.Lock()
current.used = time.Now()
current.lock.Unlock()

// If we exhausted the future cache, now's a good time to regenerate it
if future != nil {
go future.generate(ethash.cachedir, ethash.cachesondisk, ethash.tester)
}
return current.cache
}

// dataset tries to retrieve a mining dataset for the specified block number
// by first checking against a list of in-memory datasets, then against DAGs
// stored on disk, and finally generating one if none can be found.
```

```
func (ethash *Ethash) dataset(block uint64) []uint32 {
epoch := block / epochLength

// If we have a PoW for that epoch, use that
ethash.lock.Lock()

current, future := ethash.datasets[epoch], (*dataset)(nil)
if current == nil {
// No in-memory dataset, evict the oldest if the dataset limit was reached
for len(ethash.datasets) > 0 && len(ethash.datasets) >= ethash.dagsinmem {
var evict *dataset
for _, dataset := range ethash.datasets {
if evict == nil || evict.used.After(dataset.used) {
evict = dataset
}
}
delete(ethash.datasets, evict.epoch)
evict.release()

log.Trace("Evicted ethash dataset", "epoch", evict.epoch, "used", evict.used)
}
// If we have the new cache pre-generated, use that, otherwise create a new one
if ethash.fdataset != nil && ethash.fdataset.epoch == epoch {
log.Trace("Using pre-generated dataset", "epoch", epoch)
current = &dataset{epoch: ethash.fdataset.epoch} // Reload from disk
ethash.fdataset = nil
} else {
log.Trace("Requiring new ethash dataset", "epoch", epoch)
current = &dataset{epoch: epoch}
}
ethash.datasets[epoch] = current

// If we just used up the future dataset, or need a refresh, regenerate
if ethash.fdataset == nil || ethash.fdataset.epoch <= epoch {
if ethash.fdataset != nil {
ethash.fdataset.release()
}
log.Trace("Requiring new future ethash dataset", "epoch", epoch+1)
future = &dataset{epoch: epoch + 1}
ethash.fdataset = future
}
// New current dataset, set its initial timestamp
```

```
current.used = time.Now()
}
ethash.lock.Unlock()

// Wait for generation finish, bump the timestamp and finalize the cache
current.generate(ethash.dagdir, ethash.dagsondisk, ethash.tester)

current.lock.Lock()
current.used = time.Now()
current.lock.Unlock()

// If we exhausted the future dataset, now's a good time to regenerate it
if future != nil {
go future.generate(ethash.dagdir, ethash.dagsondisk, ethash.tester)
}
return current.dataset
}

// Threads returns the number of mining threads currently enabled. This doesn't
// necessarily mean that mining is running!
func (ethash *Ethash) Threads() int {
ethash.lock.Lock()
defer ethash.lock.Unlock()

return ethash.threads
}

// SetThreads updates the number of mining threads currently enabled. Calling
// this method does not start mining, only sets the thread count. If zero is
// specified, the miner will use all cores of the machine. Setting a thread
// count below zero is allowed and will cause the miner to idle, without any
// work being done.
func (ethash *Ethash) SetThreads(threads int) {
ethash.lock.Lock()
defer ethash.lock.Unlock()

// If we're running a shared PoW, set the thread count on that instead
if ethash.shared != nil {
ethash.shared.SetThreads(threads)
return
}
// Update the threads and ping any running seal to pull in any changes
```

```
ethash.threads = threads
select {
case ethash.update <- struct{}{}:
default:
}
}

// Hashrate implements PoW, returning the measured rate of the search invocations
// per second over the last minute.
func (ethash *Ethash) Hashrate() float64 {
return ethash.hashrate.Rate1()
}

// APIs implements consensus.Engine, returning the user facing RPC APIs. Currently
// that is empty.
func (ethash *Ethash) APIs(chain consensus.ChainReader) []rpc.API {
return nil
}

// SeedHash is the seed to use for generating a verification cache and the mining
// dataset.
func SeedHash(block uint64) []byte {
return seedHash(block)
}

36:F:\git\coin\ethereum\go-ethereum\consensus\ethash\ethash_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ethash

import (
"math/big"
"testing"

"github.com/ethereum/go-ethereum/core/types"
)

// Tests that ethash works correctly in test mode.
func TestTestMode(t *testing.T) {
head := &types.Header{Number: big.NewInt(1), Difficulty: big.NewInt(100)}

ethash := NewTester()
```

```
block, err := ethash.Seal(nil, types.NewBlockWithHeader(head), nil)
if err != nil {
t.Fatalf("failed to seal block: %v", err)
}
head.Nonce = types.EncodeNonce(block.Nonce())
head.MixDigest = block.MixDigest()
if err := ethash.VerifySeal(nil, head); err != nil {
t.Fatalf("unexpected verification error: %v", err)
}
}
```

37:F:\git\coin\ethereum\go-ethereum\consensus\ethash\sealer.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ethash

import (
crand "crypto/rand"
"math"
"math/big"
"math/rand"
"runtime"
"sync"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/log"
)

// Seal implements consensus.Engine, attempting to find a nonce that satisfies
// the block's difficulty requirements.
func (ethash *Ethash) Seal(chain consensus.ChainReader, block *types.Block, stop <-chan
struct{}) (*types.Block, error) {
// If we're running a fake PoW, simply return a 0 nonce immediately
if ethash.fakeMode {
header := block.Header()
header.Nonce, header.MixDigest = types.BlockNonce{}, common.Hash{}
return block.WithSeal(header), nil
}
// If we're running a shared PoW, delegate sealing to it
if ethash.shared != nil {
```

```
return ethash.shared.Seal(chain, block, stop)
}
// Create a runner and the multiple search threads it directs
abort := make(chan struct{})
found := make(chan *types.Block)

ethash.lock.Lock()
threads := ethash.threads
if ethash.rand == nil {
seed, err := crand.Int(crand.Reader, big.NewInt(math.MaxInt64))
if err != nil {
ethash.lock.Unlock()
return nil, err
}
ethash.rand = rand.New(rand.NewSource(seed.Int64()))
}
ethash.lock.Unlock()
if threads == 0 {
threads = runtime.NumCPU()
}
if threads < 0 {
threads = 0 // Allows disabling local mining without extra logic around local/remote
}
var pend sync.WaitGroup
for i := 0; i < threads; i++ {
pend.Add(1)
go func(id int, nonce uint64) {
defer pend.Done()
ethash.mine(block, id, nonce, abort, found)
}(i, uint64(ethash.rand.Int63()))
}
// Wait until sealing is terminated or a nonce is found
var result *types.Block
select {
case <-stop:
// Outside abort, stop all miner threads
close(abort)
case result = <-found:
// One of the threads found a block, abort all others
close(abort)
case <-ethash.update:
// Thread count was changed on user request, restart
```

```
close(abort)
pend.Wait()
return ethash.Seal(chain, block, stop)
}
// Wait for all miners to terminate and return the block
pend.Wait()
return result, nil
}

// mine is the actual proof-of-work miner that searches for a nonce starting from
// seed that results in correct final block difficulty.
func (ethash *Ethash) mine(block *types.Block, id int, seed uint64, abort chan struct{}, found chan
*types.Block) {
// Extract some data from the header
var (
header = block.Header()
hash   = header.HashNoNonce().Bytes()
target = new(big.Int).Div(maxUint256, header.Difficulty)

number  = header.Number.Uint64()
dataset = ethash.dataset(number)
)
// Start generating random nonces until we abort or find a good one
var (
attempts = int64(0)
nonce    = seed
)
logger := log.New("miner", id)
logger.Trace("Started ethash search for new nonces", "seed", seed)
for {
select {
case <-abort:
// Mining terminated, update stats and abort
logger.Trace("Ethash nonce search aborted", "attempts", nonce-seed)
ethash.hashrate.Mark(attempts)
return

default:
// We don't have to update hash rate on every nonce, so update after after 2^X nonces
attempts++
if (attempts % (1 << 15)) == 0 {
ethash.hashrate.Mark(attempts)
```

```
attempts = 0
}
// Compute the PoW value of this nonce
digest, result := hashimotoFull(dataset, hash, nonce)
if new(big.Int).SetBytes(result).Cmp(target) <= 0 {
// Correct nonce found, create a new header with it
header = types.CopyHeader(header)
header.Nonce = types.EncodeNonce(nonce)
header.MixDigest = common.BytesToHash(digest)

// Seal and return a block (if still needed)
select {
case found <- block.WithSeal(header):
logger.Trace("Ethash nonce found and reported", "attempts", nonce-seed, "nonce", nonce)
case <-abort:
logger.Trace("Ethash nonce found but discarded", "attempts", nonce-seed, "nonce", nonce)
}
return
}
nonce++
}
}
}
```

38:F:\git\coin\ethereum\go-ethereum\consensus\misc\dao.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package misc

import (
"bytes"
"errors"
"math/big"

"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/params"
)

var (
// ErrBadProDAOExtra is returned if a header doens't support the DAO fork on a
// pro-fork client.
```

```
ErrBadProDAOExtra = errors.New("bad DAO pro-fork extra-data")

// ErrBadNoDAOExtra is returned if a header does support the DAO fork on a no-
// fork client.
ErrBadNoDAOExtra = errors.New("bad DAO no-fork extra-data")
)

// VerifyDAOHeaderExtraData validates the extra-data field of a block header to
// ensure it conforms to DAO hard-fork rules.
//
// DAO hard-fork extension to the header validity:
//   a) if the node is no-fork, do not accept blocks in the [fork, fork+10) range
//      with the fork specific extra-data set
//   b) if the node is pro-fork, require blocks in the specific range to have the
//      unique extra-data set.
func VerifyDAOHeaderExtraData(config *params.ChainConfig, header *types.Header) error {
// Short circuit validation if the node doesn't care about the DAO fork
if config.DAOForkBlock == nil {
return nil
}
// Make sure the block is within the fork's modified extra-data range
limit := new(big.Int).Add(config.DAOForkBlock, params.DAOForkExtraRange)
if header.Number.Cmp(config.DAOForkBlock) < 0 || header.Number.Cmp(limit) >= 0 {
return nil
}
// Depending on whether we support or oppose the fork, validate the extra-data contents
if config.DAOForkSupport {
if !bytes.Equal(header.Extra, params.DAOForkBlockExtra) {
return ErrBadProDAOExtra
}
} else {
if bytes.Equal(header.Extra, params.DAOForkBlockExtra) {
return ErrBadNoDAOExtra
}
}
// All ok, header has the same extra-data we expect
return nil
}

// ApplyDAOHardFork modifies the state database according to the DAO hard-fork
// rules, transferring all balances of a set of DAO accounts to a single refund
// contract.
```

```
func ApplyDAOHardFork(statedb *state.StateDB) {
// Retrieve the contract to refund balances into
if !statedb.Exist(params.DAORefundContract) {
statedb.CreateAccount(params.DAORefundContract)
}

// Move every DAO account and extra-balance account funds into the refund contract
for _, addr := range params.DAODrainList() {
statedb.AddBalance(params.DAORefundContract, statedb.GetBalance(addr))
statedb.SetBalance(addr, new(big.Int))
}
}
```

39:F:\git\coin\ethereum\go-ethereum\consensus\misc\forks.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package misc

import (
"fmt"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/params"
)

// VerifyForkHashes verifies that blocks conforming to network hard-forks do have
// the correct hashes, to avoid clients going off on different chains. This is an
// optional feature.
func VerifyForkHashes(config *params.ChainConfig, header *types.Header, uncle bool) error {
// We don't care about uncles
if uncle {
return nil
}
// If the homestead reprice hash is set, validate it
if config.EIP150Block != nil && config.EIP150Block.Cmp(header.Number) == 0 {
if config.EIP150Hash != (common.Hash{}) && config.EIP150Hash != header.Hash() {
return fmt.Errorf("homestead gas reprice fork: have 0x%x, want 0x%x", header.Hash(),
config.EIP150Hash)
}
}
// All ok, return
```

```
return nil
}

40:F:\git\coin\ethereum\go-ethereum\console\bridge.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package console

import (
"encoding/json"
"fmt"
"io"
"strings"
"time"

"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/rpc"
"github.com/robertkrimen/otto"
)

// bridge is a collection of JavaScript utility methods to bride the .js runtime
// environment and the Go RPC connection backing the remote method calls.
type bridge struct {
client   *rpc.Client  // RPC client to execute Ethereum requests through
prompter UserPrompter // Input prompter to allow interactive user feedback
printer  io.Writer    // Output writer to serialize any display strings to
}

// newBridge creates a new JavaScript wrapper around an RPC client.
func newBridge(client *rpc.Client, prompter UserPrompter, printer io.Writer) *bridge {
return &bridge{
client:   client,
prompter: prompter,
printer:  printer,
}
}

// NewAccount is a wrapper around the personal.newAccount RPC method that uses a
// non-echoing password prompt to acquire the passphrase and executes the original
// RPC method (saved in jeth.newAccount) with it to actually execute the RPC call.
func (b *bridge) NewAccount(call otto.FunctionCall) (response otto.Value) {
var (
```

```
password string
confirm  string
err      error
)
switch {
// No password was specified, prompt the user for it
case len(call.ArgumentList) == 0:
if password, err = b.prompter.PromptPassword("Passphrase: "); err != nil {
throwJSException(err.Error())
}
if confirm, err = b.prompter.PromptPassword("Repeat passphrase: "); err != nil {
throwJSException(err.Error())
}
if password != confirm {
throwJSException("passphrases don't match!")
}

// A single string password was specified, use that
case len(call.ArgumentList) == 1 && call.Argument(0).IsString():
password, _ = call.Argument(0).ToString()

// Otherwise fail with some error
default:
throwJSException("expected 0 or 1 string argument")
}
// Password acquired, execute the call and return
ret, err := call.Otto.Call("jeth.newAccount", nil, password)
if err != nil {
throwJSException(err.Error())
}
return ret
}

// UnlockAccount is a wrapper around the personal.unlockAccount RPC method that
// uses a non-echoing password prompt to acquire the passphrase and executes the
// original RPC method (saved in jeth.unlockAccount) with it to actually execute
// the RPC call.
func (b *bridge) UnlockAccount(call otto.FunctionCall) (response otto.Value) {
// Make sure we have an account specified to unlock
if !call.Argument(0).IsString() {
throwJSException("first argument must be the account to unlock")
}
```

```
account := call.Argument(0)

// If password is not given or is the null value, prompt the user for it
var passwd otto.Value

if call.Argument(1).IsUndefined() || call.Argument(1).IsNull() {
fmt.Fprintf(b.printer, "Unlock account %s\n", account)
if input, err := b.prompter.PromptPassword("Passphrase: "); err != nil {
throwJSException(err.Error())
} else {
passwd, _ = otto.ToValue(input)
}
} else {
if !call.Argument(1).IsString() {
throwJSException("password must be a string")
}
passwd = call.Argument(1)
}
// Third argument is the duration how long the account must be unlocked.
duration := otto.NullValue()
if call.Argument(2).IsDefined() && !call.Argument(2).IsNull() {
if !call.Argument(2).IsNumber() {
throwJSException("unlock duration must be a number")
}
duration = call.Argument(2)
}
// Send the request to the backend and return
val, err := call.Otto.Call("jeth.unlockAccount", nil, account, passwd, duration)
if err != nil {
throwJSException(err.Error())
}
return val
}

// Sign is a wrapper around the personal.sign RPC method that uses a non-echoing password
// prompt to acquire the passphrase and executes the original RPC method (saved in
// jeth.sign) with it to actually execute the RPC call.
func (b *bridge) Sign(call otto.FunctionCall) (response otto.Value) {
var (
message = call.Argument(0)
account = call.Argument(1)
passwd  = call.Argument(2)
```

```
)

if !message.IsString() {
throwJSException("first argument must be the message to sign")
}
if !account.IsString() {
throwJSException("second argument must be the account to sign with")
}

// if the password is not given or null ask the user and ensure password is a string
if passwd.IsUndefined() || passwd.IsNull() {
fmt.Fprintf(b.printer, "Give password for account %s\n", account)
if input, err := b.prompter.PromptPassword("Passphrase: "); err != nil {
throwJSException(err.Error())
} else {
passwd, _ = otto.ToValue(input)
}
}
if !passwd.IsString() {
throwJSException("third argument must be the password to unlock the account")
}

// Send the request to the backend and return
val, err := call.Otto.Call("jeth.sign", nil, message, account, passwd)
if err != nil {
throwJSException(err.Error())
}
return val
}

// Sleep will block the console for the specified number of seconds.
func (b *bridge) Sleep(call otto.FunctionCall) (response otto.Value) {
if call.Argument(0).IsNumber() {
sleep, _ := call.Argument(0).ToInteger()
time.Sleep(time.Duration(sleep) * time.Second)
return otto.TrueValue()
}
return throwJSException("usage: sleep(<number of seconds>)")
}

// SleepBlocks will block the console for a specified number of new blocks optionally
// until the given timeout is reached.
```

```
func (b *bridge) SleepBlocks(call otto.FunctionCall) (response otto.Value) {
var (
blocks = int64(0)
sleep  = int64(9999999999999999) // indefinitely
)
// Parse the input parameters for the sleep
nArgs := len(call.ArgumentList)
if nArgs == 0 {
throwJSException("usage: sleepBlocks(<n blocks>[, max sleep in seconds])")
}
if nArgs >= 1 {
if call.Argument(0).IsNumber() {
blocks, _ = call.Argument(0).ToInteger()
} else {
throwJSException("expected number as first argument")
}
}
if nArgs >= 2 {
if call.Argument(1).IsNumber() {
sleep, _ = call.Argument(1).ToInteger()
} else {
throwJSException("expected number as second argument")
}
}
// go through the console, this will allow web3 to call the appropriate
// callbacks if a delayed response or notification is received.
blockNumber := func() int64 {
result, err := call.Otto.Run("eth.blockNumber")
if err != nil {
throwJSException(err.Error())
}
block, err := result.ToInteger()
if err != nil {
throwJSException(err.Error())
}
return block
}
// Poll the current block number until either it ot a timeout is reached
targetBlockNr := blockNumber() + blocks
deadline := time.Now().Add(time.Duration(sleep) * time.Second)

for time.Now().Before(deadline) {
```

```
if blockNumber() >= targetBlockNr {
return otto.TrueValue()
}
time.Sleep(time.Second)
}
return otto.FalseValue()
}

type jsonrpcCall struct {
Id     int64
Method string
Params []interface{}
}

// Send implements the web3 provider "send" method.
func (b *bridge) Send(call otto.FunctionCall) (response otto.Value) {
// Remarshal the request into a Go value.
JSON, _ := call.Otto.Object("JSON")
reqVal, err := JSON.Call("stringify", call.Argument(0))
if err != nil {
throwJSException(err.Error())
}
var (
rawReq = reqVal.String()
dec    = json.NewDecoder(strings.NewReader(rawReq))
reqs   []jsonrpcCall
batch  bool
)
dec.UseNumber() // avoid float64s
if rawReq[0] == '[' {
batch = true
dec.Decode(&reqs)
} else {
batch = false
reqs = make([]jsonrpcCall, 1)
dec.Decode(&reqs[0])
}

// Execute the requests.
resps, _ := call.Otto.Object("new Array()")
for _, req := range reqs {
resp, _ := call.Otto.Object(`({"jsonrpc":"2.0"})`)
```

```
resp.Set("id", req.Id)
var result json.RawMessage
err = b.client.Call(&result, req.Method, req.Params...)
switch err := err.(type) {
case nil:
if result == nil {
// Special case null because it is decoded as an empty
// raw message for some reason.
resp.Set("result", otto.NullValue())
} else {
resultVal, err := JSON.Call("parse", string(result))
if err != nil {
setError(resp, -32603, err.Error())
} else {
resp.Set("result", resultVal)
}
}
case rpc.Error:
setError(resp, err.ErrorCode(), err.Error())
default:
setError(resp, -32603, err.Error())
}
resps.Call("push", resp)
}

// Return the responses either to the callback (if supplied)
// or directly as the return value.
if batch {
response = resps.Value()
} else {
response, _ = resps.Get("0")
}
if fn := call.Argument(1); fn.Class() == "Function" {
fn.Call(otto.NullValue(), otto.NullValue(), response)
return otto.UndefinedValue()
}
return response
}

func setError(resp *otto.Object, code int, msg string) {
resp.Set("error", map[string]interface{}{"code": code, "message": msg})
}
```

```
// throwJSException panics on an otto.Value. The Otto VM will recover from the
// Go panic and throw msg as a JavaScript error.
func throwJSException(msg interface{}) otto.Value {
val, err := otto.ToValue(msg)
if err != nil {
log.Error("Failed to serialize JavaScript exception", "exception", msg, "err", err)
}
panic(val)
}

41:F:\git\coin\ethereum\go-ethereum\console\console.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package console

import (
"fmt"
"io"
"io/ioutil"
"os"
"os/signal"
"path/filepath"
"regexp"
"sort"
"strings"

"github.com/ethereum/go-ethereum/internal/jsre"
"github.com/ethereum/go-ethereum/internal/web3ext"
"github.com/ethereum/go-ethereum/rpc"
"github.com/mattn/go-colorable"
"github.com/peterh/liner"
"github.com/robertkrimen/otto"
)

var (
passwordRegexp = regexp.MustCompile(`personal.[nus]`)
onlyWhitespace = regexp.MustCompile(`^\s*$`)
exit           = regexp.MustCompile(`^\s*exit\s*;*\s*$`)
)

// HistoryFile is the file within the data directory to store input scrollback.
```

```
const HistoryFile = "history"

// DefaultPrompt is the default prompt line prefix to use for user input querying.
const DefaultPrompt = "> "

// Config is te collection of configurations to fine tune the behavior of the
// JavaScript console.
type Config struct {
DataDir  string       // Data directory to store the console history at
DocRoot  string       // Filesystem path from where to load JavaScript files from
Client   *rpc.Client  // RPC client to execute Ethereum requests through
Prompt   string       // Input prompt prefix string (defaults to DefaultPrompt)
Prompter UserPrompter // Input prompter to allow interactive user feedback (defaults to
TerminalPrompter)
Printer  io.Writer    // Output writer to serialize any display strings to (defaults to os.Stdout)
Preload  []string     // Absolute paths to JavaScript files to preload
}

// Console is a JavaScript interpreted runtime environment. It is a fully fleged
// JavaScript console attached to a running node via an external or in-process RPC
// client.
type Console struct {
client   *rpc.Client  // RPC client to execute Ethereum requests through
jsre     *jsre.JSRE   // JavaScript runtime environment running the interpreter
prompt   string       // Input prompt prefix string
prompter UserPrompter // Input prompter to allow interactive user feedback
histPath string       // Absolute path to the console scrollback history
history  []string     // Scroll history maintained by the console
printer  io.Writer    // Output writer to serialize any display strings to
}

func New(config Config) (*Console, error) {
// Handle unset config values gracefully
if config.Prompter == nil {
config.Prompter = Stdin
}
if config.Prompt == "" {
config.Prompt = DefaultPrompt
}
if config.Printer == nil {
config.Printer = colorable.NewColorableStdout()
}
```

```
// Initialize the console and return
console := &Console{
client:   config.Client,
jsre:     jsre.New(config.DocRoot, config.Printer),
prompt:   config.Prompt,
prompter: config.Prompter,
printer:  config.Printer,
histPath: filepath.Join(config.DataDir, HistoryFile),
}
if err := console.init(config.Preload); err != nil {
return nil, err
}
return console, nil
}

// init retrieves the available APIs from the remote RPC provider and initializes
// the console's JavaScript namespaces based on the exposed modules.
func (c *Console) init(preload []string) error {
// Initialize the JavaScript <-> Go RPC bridge
bridge := newBridge(c.client, c.prompter, c.printer)
c.jsre.Set("jeth", struct{}{})

jethObj, _ := c.jsre.Get("jeth")
jethObj.Object().Set("send", bridge.Send)
jethObj.Object().Set("sendAsync", bridge.Send)

consoleObj, _ := c.jsre.Get("console")
consoleObj.Object().Set("log", c.consoleOutput)
consoleObj.Object().Set("error", c.consoleOutput)

// Load all the internal utility JavaScript libraries
if err := c.jsre.Compile("bignumber.js", jsre.BigNumber_JS); err != nil {
return fmt.Errorf("bignumber.js: %v", err)
}
if err := c.jsre.Compile("web3.js", jsre.Web3_JS); err != nil {
return fmt.Errorf("web3.js: %v", err)
}
if _, err := c.jsre.Run("var Web3 = require('web3');"); err != nil {
return fmt.Errorf("web3 require: %v", err)
}
if _, err := c.jsre.Run("var web3 = new Web3(jeth);"); err != nil {
return fmt.Errorf("web3 provider: %v", err)
```

```
}
// Load the supported APIs into the JavaScript runtime environment
apis, err := c.client.SupportedModules()
if err != nil {
return fmt.Errorf("api modules: %v", err)
}
flatten := "var eth = web3.eth; var personal = web3.personal; "
for api := range apis {
if api == "web3" {
continue // manually mapped or ignore
}
if file, ok := web3ext.Modules[api]; ok {
// Load our extension for the module.
if err = c.jsre.Compile(fmt.Sprintf("%s.js", api), file); err != nil {
return fmt.Errorf("%s.js: %v", api, err)
}
flatten += fmt.Sprintf("var %s = web3.%s; ", api, api)
} else if obj, err := c.jsre.Run("web3." + api); err == nil && obj.IsObject() {
// Enable web3.js built-in extension if available.
flatten += fmt.Sprintf("var %s = web3.%s; ", api, api)
}
}
if _, err = c.jsre.Run(flatten); err != nil {
return fmt.Errorf("namespace flattening: %v", err)
}
// Initialize the global name register (disabled for now)
//c.jsre.Run(`var GlobalRegistrar = eth.contract(` + registrar.GlobalRegistrarAbi + `);   registrar =
GlobalRegistrar.at("` + registrar.GlobalRegistrarAddr + `");`)

// If the console is in interactive mode, instrument password related methods to query the user
if c.prompter != nil {
// Retrieve the account management object to instrument
personal, err := c.jsre.Get("personal")
if err != nil {
return err
}
// Override the unlockAccount, newAccount and sign methods since these require user interaction.
// Assign these method in the Console the original web3 callbacks. These will be called by the
jeth.*
// methods after they got the password from the user and send the original web3 request to the
backend.
if obj := personal.Object(); obj != nil { // make sure the personal api is enabled over the interface
```

```
if _, err = c.jsre.Run(`jeth.unlockAccount = personal.unlockAccount;`); err != nil {
return fmt.Errorf("personal.unlockAccount: %v", err)
}
if _, err = c.jsre.Run(`jeth.newAccount = personal.newAccount;`); err != nil {
return fmt.Errorf("personal.newAccount: %v", err)
}
if _, err = c.jsre.Run(`jeth.sign = personal.sign;`); err != nil {
return fmt.Errorf("personal.sign: %v", err)
}
obj.Set("unlockAccount", bridge.UnlockAccount)
obj.Set("newAccount", bridge.NewAccount)
obj.Set("sign", bridge.Sign)
}
}
// The admin.sleep and admin.sleepBlocks are offered by the console and not by the RPC layer.
admin, err := c.jsre.Get("admin")
if err != nil {
return err
}
if obj := admin.Object(); obj != nil { // make sure the admin api is enabled over the interface
obj.Set("sleepBlocks", bridge.SleepBlocks)
obj.Set("sleep", bridge.Sleep)
}
// Preload any JavaScript files before starting the console
for _, path := range preload {
if err := c.jsre.Exec(path); err != nil {
failure := err.Error()
if ottoErr, ok := err.(*otto.Error); ok {
failure = ottoErr.String()
}
return fmt.Errorf("%s: %v", path, failure)
}
}
// Configure the console's input prompter for scrollback and tab completion
if c.prompter != nil {
if content, err := ioutil.ReadFile(c.histPath); err != nil {
c.prompter.SetHistory(nil)
} else {
c.history = strings.Split(string(content), "\n")
c.prompter.SetHistory(c.history)
}
c.prompter.SetWordCompleter(c.AutoCompleteInput)
```

```
}
return nil
}

// consoleOutput is an override for the console.log and console.error methods to
// stream the output into the configured output stream instead of stdout.
func (c *Console) consoleOutput(call otto.FunctionCall) otto.Value {
output := []string{}
for _, argument := range call.ArgumentList {
output = append(output, fmt.Sprintf("%v", argument))
}
fmt.Fprintln(c.printer, strings.Join(output, " "))
return otto.Value{}
}

// AutoCompleteInput is a pre-assembled word completer to be used by the user
// input prompter to provide hints to the user about the methods available.
func (c *Console) AutoCompleteInput(line string, pos int) (string, []string, string) {
// No completions can be provided for empty inputs
if len(line) == 0 || pos == 0 {
return "", nil, ""
}
// Chunck data to relevant part for autocompletion
// E.g. in case of nested lines eth.getBalance(eth.coinb<tab><tab>
start := pos - 1
for ; start > 0; start-- {
// Skip all methods and namespaces (i.e. including te dot)
if line[start] == '.' || (line[start] >= 'a' && line[start] <= 'z') || (line[start] >= 'A' && line[start] <= 'Z') {
continue
}
// Handle web3 in a special way (i.e. other numbers aren't auto completed)
if start >= 3 && line[start-3:start] == "web3" {
start -= 3
continue
}
// We've hit an unexpected character, autocomplete form here
start++
break
}
return line[:start], c.jsre.CompleteKeywords(line[start:pos]), line[pos:]
}
```

```
// Welcome show summary of current Geth instance and some metadata about the
// console's available modules.
func (c *Console) Welcome() {
// Print some generic Geth metadata
fmt.Fprintf(c.printer, "Welcome to the Geth JavaScript console!\n\n")
c.jsre.Run(`
console.log("instance: " + web3.version.node);
console.log("coinbase: " + eth.coinbase);
console.log("at block: " + eth.blockNumber + " (" + new Date(1000 *
eth.getBlock(eth.blockNumber).timestamp) + ")");
console.log(" datadir: " + admin.datadir);
`)
// List all the supported modules for the user to call
if apis, err := c.client.SupportedModules(); err == nil {
modules := make([]string, 0, len(apis))
for api, version := range apis {
modules = append(modules, fmt.Sprintf("%s:%s", api, version))
}
sort.Strings(modules)
fmt.Fprintln(c.printer, " modules:", strings.Join(modules, " "))
}
fmt.Fprintln(c.printer)
}

// Evaluate executes code and pretty prints the result to the specified output
// stream.
func (c *Console) Evaluate(statement string) error {
defer func() {
if r := recover(); r != nil {
fmt.Fprintf(c.printer, "[native] error: %v\n", r)
}
}()
return c.jsre.Evaluate(statement, c.printer)
}

// Interactive starts an interactive user session, where input is propted from
// the configured user prompter.
func (c *Console) Interactive() {
var (
prompt    = c.prompt          // Current prompt line (used for multi-line inputs)
indents   = 0                 // Current number of input indents (used for multi-line inputs)
input     = ""                // Current user input
```

```
scheduler = make(chan string) // Channel to send the next prompt on and receive the input
)
// Start a goroutine to listen for promt requests and send back inputs
go func() {
for {
// Read the next user input
line, err := c.prompter.PromptInput(<-scheduler)
if err != nil {
// In case of an error, either clear the prompt or fail
if err == liner.ErrPromptAborted { // ctrl-C
prompt, indents, input = c.prompt, 0, ""
scheduler <- ""
continue
}
close(scheduler)
return
}
// User input retrieved, send for interpretation and loop
scheduler <- line
}
}()
// Monitor Ctrl-C too in case the input is empty and we need to bail
abort := make(chan os.Signal, 1)
signal.Notify(abort, os.Interrupt)

// Start sending prompts to the user and reading back inputs
for {
// Send the next prompt, triggering an input read and process the result
scheduler <- prompt
select {
case <-abort:
// User forcefully quite the console
fmt.Fprintln(c.printer, "caught interrupt, exiting")
return

case line, ok := <-scheduler:
// User input was returned by the prompter, handle special cases
if !ok || (indents <= 0 && exit.MatchString(line)) {
return
}
if onlyWhitespace.MatchString(line) {
continue
```

```
}
// Append the line to the input and check for multi-line interpretation
input += line + "\n"

indents = countIndents(input)
if indents <= 0 {
prompt = c.prompt
} else {
prompt = strings.Repeat(".", indents*3) + " "
}
// If all the needed lines are present, save the command and run
if indents <= 0 {
if len(input) > 0 && input[0] != ' ' && !passwordRegexp.MatchString(input) {
if command := strings.TrimSpace(input); len(c.history) == 0 || command != c.history[len(c.history)-
1] {
c.history = append(c.history, command)
if c.prompter != nil {
c.prompter.AppendHistory(command)
}
}
}
c.Evaluate(input)
input = ""
}
}
}
}

// countIndents returns the number of identations for the given input.
// In case of invalid input such as var a = } the result can be negative.
func countIndents(input string) int {
var (
indents     = 0
inString    = false
strOpenChar = ' '   // keep track of the string open char to allow var str = "I'm ....";
charEscaped = false // keep track if the previous char was the '\' char, allow var str = "abc\"def";
)

for _, c := range input {
switch c {
case '\\':
// indicate next char as escaped when in string and previous char isn't escaping this backslash
```

```
if !charEscaped && inString {
charEscaped = true
}
case '\'', '"':
if inString && !charEscaped && strOpenChar == c { // end string
inString = false
} else if !inString && !charEscaped { // begin string
inString = true
strOpenChar = c
}
charEscaped = false
case '{', '(':
if !inString { // ignore brackets when in string, allow var str = "a{"; without indenting
indents++
}
charEscaped = false
case '}', ')':
if !inString {
indents--
}
charEscaped = false
default:
charEscaped = false
}
}

return indents
}

// Execute runs the JavaScript file specified as the argument.
func (c *Console) Execute(path string) error {
return c.jsre.Exec(path)
}

// Stop cleans up the console and terminates the runtime envorinment.
func (c *Console) Stop(graceful bool) error {
if err := ioutil.WriteFile(c.histPath, []byte(strings.Join(c.history, "\n")), 0600); err != nil {
return err
}
if err := os.Chmod(c.histPath, 0600); err != nil { // Force 0600, even if it was different previously
return err
}
```

```
c.jsre.Stop(graceful)
return nil
}

42:F:\git\coin\ethereum\go-ethereum\console\console_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package console

import (
"bytes"
"errors"
"fmt"
"io/ioutil"
"os"
"strings"
"testing"
"time"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/eth"
"github.com/ethereum/go-ethereum/internal/jsre"
"github.com/ethereum/go-ethereum/node"
)

const (
testInstance = "console-tester"
testAddress  = "0x8605cdbbdb6d264aa742e77020dcbc58fcdce182"
)

// hookedPrompter implements UserPrompter to simulate use input via channels.
type hookedPrompter struct {
scheduler chan string
}

func (p *hookedPrompter) PromptInput(prompt string) (string, error) {
// Send the prompt to the tester
select {
case p.scheduler <- prompt:
case <-time.After(time.Second):
return "", errors.New("prompt timeout")
```

```
}
// Retrieve the response and feed to the console
select {
case input := <-p.scheduler:
return input, nil
case <-time.After(time.Second):
return "", errors.New("input timeout")
}
}

func (p *hookedPrompter) PromptPassword(prompt string) (string, error) {
return "", errors.New("not implemented")
}
func (p *hookedPrompter) PromptConfirm(prompt string) (bool, error) {
return false, errors.New("not implemented")
}
func (p *hookedPrompter) SetHistory(history []string)              {}
func (p *hookedPrompter) AppendHistory(command string)             {}
func (p *hookedPrompter) SetWordCompleter(completer WordCompleter) {}

// tester is a console test environment for the console tests to operate on.
type tester struct {
workspace string
stack     *node.Node
ethereum  *eth.Ethereum
console   *Console
input     *hookedPrompter
output    *bytes.Buffer

lastConfirm string
}

// newTester creates a test environment based on which the console can operate.
// Please ensure you call Close() on the returned tester to avoid leaks.
func newTester(t *testing.T, confOverride func(*eth.Config)) *tester {
// Create a temporary storage for the node keys and initialize it
workspace, err := ioutil.TempDir("", "console-tester-")
if err != nil {
t.Fatalf("failed to create temporary keystore: %v", err)
}

// Create a networkless protocol stack and start an Ethereum service within
```

```
stack, err := node.New(&node.Config{DataDir: workspace, UseLightweightKDF: true, Name:
testInstance})
if err != nil {
t.Fatalf("failed to create node: %v", err)
}
ethConf := &eth.Config{
Genesis:   core.DevGenesisBlock(),
Etherbase: common.HexToAddress(testAddress),
PowTest:   true,
}
if confOverride != nil {
confOverride(ethConf)
}
if err = stack.Register(func(ctx *node.ServiceContext) (node.Service, error) { return eth.New(ctx,
ethConf) }); err != nil {
t.Fatalf("failed to register Ethereum protocol: %v", err)
}
// Start the node and assemble the JavaScript console around it
if err = stack.Start(); err != nil {
t.Fatalf("failed to start test stack: %v", err)
}
client, err := stack.Attach()
if err != nil {
t.Fatalf("failed to attach to node: %v", err)
}
prompter := &hookedPrompter{scheduler: make(chan string)}
printer := new(bytes.Buffer)

console, err := New(Config{
DataDir:  stack.DataDir(),
DocRoot:  "testdata",
Client:   client,
Prompter: prompter,
Printer:  printer,
Preload:  []string{"preload.js"},
})
if err != nil {
t.Fatalf("failed to create JavaScript console: %v", err)
}
// Create the final tester and return
var ethereum *eth.Ethereum
stack.Service(&ethereum)
```

```
return &tester{
workspace: workspace,
stack:     stack,
ethereum:  ethereum,
console:   console,
input:     prompter,
output:    printer,
}
}

// Close cleans up any temporary data folders and held resources.
func (env *tester) Close(t *testing.T) {
if err := env.console.Stop(false); err != nil {
t.Errorf("failed to stop embedded console: %v", err)
}
if err := env.stack.Stop(); err != nil {
t.Errorf("failed to stop embedded node: %v", err)
}
os.RemoveAll(env.workspace)
}

// Tests that the node lists the correct welcome message, notably that it contains
// the instance name, coinbase account, block number, data directory and supported
// console modules.
func TestWelcome(t *testing.T) {
tester := newTester(t, nil)
defer tester.Close(t)

tester.console.Welcome()

output := string(tester.output.Bytes())
if want := "Welcome"; !strings.Contains(output, want) {
t.Fatalf("console output missing welcome message: have\n%s\nwant also %s", output, want)
}
if want := fmt.Sprintf("instance: %s", testInstance); !strings.Contains(output, want) {
t.Fatalf("console output missing instance: have\n%s\nwant also %s", output, want)
}
if want := fmt.Sprintf("coinbase: %s", testAddress); !strings.Contains(output, want) {
t.Fatalf("console output missing coinbase: have\n%s\nwant also %s", output, want)
}
if want := "at block: 0"; !strings.Contains(output, want) {
```

```
t.Fatalf("console output missing sync status: have\n%s\nwant also %s", output, want)
}
if want := fmt.Sprintf("datadir: %s", tester.workspace); !strings.Contains(output, want) {
t.Fatalf("console output missing coinbase: have\n%s\nwant also %s", output, want)
}
}

// Tests that JavaScript statement evaluation works as intended.
func TestEvaluate(t *testing.T) {
tester := newTester(t, nil)
defer tester.Close(t)

tester.console.Evaluate("2 + 2")
if output := string(tester.output.Bytes()); !strings.Contains(output, "4") {
t.Fatalf("statement evaluation failed: have %s, want %s", output, "4")
}
}

// Tests that the console can be used in interactive mode.
func TestInteractive(t *testing.T) {
// Create a tester and run an interactive console in the background
tester := newTester(t, nil)
defer tester.Close(t)

go tester.console.Interactive()

// Wait for a promt and send a statement back
select {
case <-tester.input.scheduler:
case <-time.After(time.Second):
t.Fatalf("initial prompt timeout")
}
select {
case tester.input.scheduler <- "2+2":
case <-time.After(time.Second):
t.Fatalf("input feedback timeout")
}
// Wait for the second promt and ensure first statement was evaluated
select {
case <-tester.input.scheduler:
case <-time.After(time.Second):
t.Fatalf("secondary prompt timeout")
```

```
}
if output := string(tester.output.Bytes()); !strings.Contains(output, "4") {
t.Fatalf("statement evaluation failed: have %s, want %s", output, "4")
}
}

// Tests that preloaded JavaScript files have been executed before user is given
// input.
func TestPreload(t *testing.T) {
tester := newTester(t, nil)
defer tester.Close(t)

tester.console.Evaluate("preloaded")
if output := string(tester.output.Bytes()); !strings.Contains(output, "some-preloaded-string") {
t.Fatalf("preloaded variable missing: have %s, want %s", output, "some-preloaded-string")
}
}

// Tests that JavaScript scripts can be executes from the configured asset path.
func TestExecute(t *testing.T) {
tester := newTester(t, nil)
defer tester.Close(t)

tester.console.Execute("exec.js")

tester.console.Evaluate("execed")
if output := string(tester.output.Bytes()); !strings.Contains(output, "some-executed-string") {
t.Fatalf("execed variable missing: have %s, want %s", output, "some-executed-string")
}
}

// Tests that the JavaScript objects returned by statement executions are properly
// pretty printed instead of just displaing "[object]".
func TestPrettyPrint(t *testing.T) {
tester := newTester(t, nil)
defer tester.Close(t)

tester.console.Evaluate("obj = {int: 1, string: 'two', list: [3, 3, 3], obj: {null: null, func: function(){}}}")

// Define some specially formatted fields
var (
one   = jsre.NumberColor("1")
```

```
two   = jsre.StringColor("\"two\"")
three = jsre.NumberColor("3")
null  = jsre.SpecialColor("null")
fun   = jsre.FunctionColor("function()")
)
// Assemble the actual output we're after and verify
want := `{
  int: ` + one + `,
  list: [` + three + `, ` + three + `, ` + three + `],
  obj: {
    null: ` + null + `,
    func: ` + fun + `
  },
  string: ` + two + `
}
`
if output := string(tester.output.Bytes()); output != want {
t.Fatalf("pretty print mismatch: have %s, want %s", output, want)
}
}

// Tests that the JavaScript exceptions are properly formatted and colored.
func TestPrettyError(t *testing.T) {
tester := newTester(t, nil)
defer tester.Close(t)
tester.console.Evaluate("throw 'hello'")

want := jsre.ErrorColor("hello") + "\n"
if output := string(tester.output.Bytes()); output != want {
t.Fatalf("pretty error mismatch: have %s, want %s", output, want)
}
}

// Tests that tests if the number of indents for JS input is calculated correct.
func TestIndenting(t *testing.T) {
testCases := []struct {
input              string
expectedIndentCount int
}{
{`var a = 1;`, 0},
{`"some string"`, 0},
{`"some string with (parentesis`, 0},
```

```
{`"some string with newline
("`, 0},
{`function v(a,b) {}`, 0},
{`function f(a,b) { var str = "asd("; };`, 0},
{`function f(a) {`, 1},
{`function f(a, function(b) {`, 2},
{`function f(a, function(b) {
    var str = "a)}";
  });`, 0},
{`function f(a,b) {
   var str = "a{b(" + a, ", " + b;
   }`, 0},
{`var str = "\"{"`, 0},
{`var str = "'("`, 0},
{`var str = "\\{"`, 0},
{`var str = "\\\\{"`, 0},
{`var str = 'a"{`, 0},
{`var obj = {`, 1},
{`var obj = { {a:1`, 2},
{`var obj = { {a:1}`, 1},
{`var obj = { {a:1}, b:2}`, 0},
{`var obj = {}`, 0},
{`var obj = {
a: 1, b: 2
}`, 0},
{`var test = }`, -1},
{`var str = "a\""; var obj = {`, 1},
}

for i, tt := range testCases {
counted := countIndents(tt.input)
if counted != tt.expectedIndentCount {
t.Errorf("test %d: invalid indenting: have %d, want %d", i, counted, tt.expectedIndentCount)
}
}
}

43:F:\git\coin\ethereum\go-ethereum\console\prompter.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package console
```

```
import (
"fmt"
"strings"

"github.com/peterh/liner"
)

// Stdin holds the stdin line reader (also using stdout for printing prompts).
// Only this reader may be used for input because it keeps an internal buffer.
var Stdin = newTerminalPrompter()

// UserPrompter defines the methods needed by the console to promt the user for
// various types of inputs.
type UserPrompter interface {
// PromptInput displays the given prompt to the user and requests some textual
// data to be entered, returning the input of the user.
PromptInput(prompt string) (string, error)

// PromptPassword displays the given prompt to the user and requests some textual
// data to be entered, but one which must not be echoed out into the terminal.
// The method returns the input provided by the user.
PromptPassword(prompt string) (string, error)

// PromptConfirm displays the given prompt to the user and requests a boolean
// choice to be made, returning that choice.
PromptConfirm(prompt string) (bool, error)

// SetHistory sets the the input scrollback history that the prompter will allow
// the user to scroll back to.
SetHistory(history []string)

// AppendHistory appends an entry to the scrollback history. It should be called
// if and only if the prompt to append was a valid command.
AppendHistory(command string)

// SetWordCompleter sets the completion function that the prompter will call to
// fetch completion candidates when the user presses tab.
SetWordCompleter(completer WordCompleter)
}

// WordCompleter takes the currently edited line with the cursor position and
// returns the completion candidates for the partial word to be completed. If
```

```
// the line is "Hello, wo!!!" and the cursor is before the first '!', ("Hello,
// wo!!!", 9) is passed to the completer which may returns ("Hello, ", {"world",
// "Word"}, "!!!") to have "Hello, world!!!".
type WordCompleter func(line string, pos int) (string, []string, string)

// terminalPrompter is a UserPrompter backed by the liner package. It supports
// prompting the user for various input, among others for non-echoing password
// input.
type terminalPrompter struct {
*liner.State
warned     bool
supported  bool
normalMode liner.ModeApplier
rawMode    liner.ModeApplier
}

// newTerminalPrompter creates a liner based user input prompter working off the
// standard input and output streams.
func newTerminalPrompter() *terminalPrompter {
p := new(terminalPrompter)
// Get the original mode before calling NewLiner.
// This is usually regular "cooked" mode where characters echo.
normalMode, _ := liner.TerminalMode()
// Turn on liner. It switches to raw mode.
p.State = liner.NewLiner()
rawMode, err := liner.TerminalMode()
if err != nil || !liner.TerminalSupported() {
p.supported = false
} else {
p.supported = true
p.normalMode = normalMode
p.rawMode = rawMode
// Switch back to normal mode while we're not prompting.
normalMode.ApplyMode()
}
p.SetCtrlCAborts(true)
p.SetTabCompletionStyle(liner.TabPrints)
p.SetMultiLineMode(true)
return p
}

// PromptInput displays the given prompt to the user and requests some textual
```

```
// data to be entered, returning the input of the user.
func (p *terminalPrompter) PromptInput(prompt string) (string, error) {
if p.supported {
p.rawMode.ApplyMode()
defer p.normalMode.ApplyMode()
} else {
// liner tries to be smart about printing the prompt
// and doesn't print anything if input is redirected.
// Un-smart it by printing the prompt always.
fmt.Print(prompt)
prompt = ""
defer fmt.Println()
}
return p.State.Prompt(prompt)
}

// PromptPassword displays the given prompt to the user and requests some textual
// data to be entered, but one which must not be echoed out into the terminal.
// The method returns the input provided by the user.
func (p *terminalPrompter) PromptPassword(prompt string) (passwd string, err error) {
if p.supported {
p.rawMode.ApplyMode()
defer p.normalMode.ApplyMode()
return p.State.PasswordPrompt(prompt)
}
if !p.warned {
fmt.Println("!! Unsupported terminal, password will be echoed.")
p.warned = true
}
// Just as in Prompt, handle printing the prompt here instead of relying on liner.
fmt.Print(prompt)
passwd, err = p.State.Prompt("")
fmt.Println()
return passwd, err
}

// PromptConfirm displays the given prompt to the user and requests a boolean
// choice to be made, returning that choice.
func (p *terminalPrompter) PromptConfirm(prompt string) (bool, error) {
input, err := p.Prompt(prompt + " [y/N] ")
if len(input) > 0 && strings.ToUpper(input[:1]) == "Y" {
return true, nil
```

```
}
return false, err
}

// SetHistory sets the the input scrollback history that the prompter will allow
// the user to scroll back to.
func (p *terminalPrompter) SetHistory(history []string) {
p.State.ReadHistory(strings.NewReader(strings.Join(history, "\n")))
}

// AppendHistory appends an entry to the scrollback history. It should be called
// if and only if the prompt to append was a valid command.
func (p *terminalPrompter) AppendHistory(command string) {
p.State.AppendHistory(command)
}

// SetWordCompleter sets the completion function that the prompter will call to
// fetch completion candidates when the user presses tab.
func (p *terminalPrompter) SetWordCompleter(completer WordCompleter) {
p.State.SetWordCompleter(liner.WordCompleter(completer))
}

44:F:\git\coin\ethereum\go-ethereum\contracts\chequebook\api.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package chequebook

import (
"errors"
"math/big"

"github.com/ethereum/go-ethereum/common"
)

const Version = "1.0"

var errNoChequebook = errors.New("no chequebook")

type Api struct {
chequebookf func() *Chequebook
}
```

```
func NewApi(ch func() *Chequebook) *Api {
return &Api{ch}
}

func (self *Api) Balance() (string, error) {
ch := self.chequebookf()
if ch == nil {
return "", errNoChequebook
}
return ch.Balance().String(), nil
}

func (self *Api) Issue(beneficiary common.Address, amount *big.Int) (cheque *Cheque, err error) {
ch := self.chequebookf()
if ch == nil {
return nil, errNoChequebook
}
return ch.Issue(beneficiary, amount)
}

func (self *Api) Cash(cheque *Cheque) (txhash string, err error) {
ch := self.chequebookf()
if ch == nil {
return "", errNoChequebook
}
return ch.Cash(cheque)
}

func (self *Api) Deposit(amount *big.Int) (txhash string, err error) {
ch := self.chequebookf()
if ch == nil {
return "", errNoChequebook
}
return ch.Deposit(amount)
}

45:F:\git\coin\ethereum\go-ethereum\contracts\chequebook\cheque.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package chequebook package wraps the 'chequebook' Ethereum smart contract.
//
// The functions in this package allow using chequebook for
```

```
// issuing, receiving, verifying cheques in ether; (auto)cashing cheques in ether
// as well as (auto)depositing ether to the chequebook contract.
package chequebook

//go:generate abigen --sol contract/chequebook.sol --pkg contract --out contract/chequebook.go
//go:generate go run ./gencode.go

import (
"bytes"
"context"
"crypto/ecdsa"
"encoding/json"
"fmt"
"io/ioutil"
"math/big"
"os"
"sync"
"time"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/hexutil"
"github.com/ethereum/go-ethereum/contracts/chequebook/contract"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/swarm/services/swap/swap"
)

// TODO(zelig): watch peer solvency and notify of bouncing cheques
// TODO(zelig): enable paying with cheque by signing off

// Some functionality requires interacting with the blockchain:
// * setting current balance on peer's chequebook
// * sending the transaction to cash the cheque
// * depositing ether to the chequebook
// * watching incoming ether

var (
gasToCash = big.NewInt(2000000) // gas cost of a cash transaction using chequebook
// gasToDeploy = big.NewInt(3000000)
)
```

```
// Backend wraps all methods required for chequebook operation.
type Backend interface {
bind.ContractBackend
TransactionReceipt(ctx context.Context, txHash common.Hash) (*types.Receipt, error)
BalanceAt(ctx context.Context, address common.Address, blockNum *big.Int) (*big.Int, error)
}

// Cheque represents a payment promise to a single beneficiary.
type Cheque struct {
Contract    common.Address // address of chequebook, needed to avoid cross-contract
submission
Beneficiary common.Address
Amount      *big.Int // cumulative amount of all funds sent
Sig         []byte   // signature Sign(Keccak256(contract, beneficiary, amount), prvKey)
}

func (self *Cheque) String() string {
return fmt.Sprintf("contract: %s, beneficiary: %s, amount: %v, signature: %x", self.Contract.Hex(),
self.Beneficiary.Hex(), self.Amount, self.Sig)
}

type Params struct {
ContractCode, ContractAbi string
}

var ContractParams = &Params{contract.ChequebookBin, contract.ChequebookABI}

// Chequebook can create and sign cheques from a single contract to multiple beneficiaries.
// It is the outgoing payment handler for peer to peer micropayments.
type Chequebook struct {
path     string                 // path to chequebook file
prvKey   *ecdsa.PrivateKey      // private key to sign cheque with
lock     sync.Mutex             //
backend  Backend                // blockchain API
quit     chan bool              // when closed causes autodeposit to stop
owner    common.Address         // owner address (derived from pubkey)
contract *contract.Chequebook   // abigen binding
session  *contract.ChequebookSession // abigen binding with Tx Opts

// persisted fields
balance      *big.Int            // not synced with blockchain
```

```
contractAddr common.Address              // contract address
sent         map[common.Address]*big.Int //tallies for beneficiaries

txhash    string   // tx hash of last deposit tx
threshold *big.Int // threshold that triggers autodeposit if not nil
buffer    *big.Int // buffer to keep on top of balance for fork protection

log log.Logger // contextual logger with the contract address embedded
}

func (self *Chequebook) String() string {
return fmt.Sprintf("contract: %s, owner: %s, balance: %v, signer: %x", self.contractAddr.Hex(),
self.owner.Hex(), self.balance, self.prvKey.PublicKey)
}

// NewChequebook creates a new Chequebook.
func NewChequebook(path string, contractAddr common.Address, prvKey *ecdsa.PrivateKey,
backend Backend) (self *Chequebook, err error) {
balance := new(big.Int)
sent := make(map[common.Address]*big.Int)

chbook, err := contract.NewChequebook(contractAddr, backend)
if err != nil {
return nil, err
}
transactOpts := bind.NewKeyedTransactor(prvKey)
session := &contract.ChequebookSession{
Contract:     chbook,
TransactOpts: *transactOpts,
}

self = &Chequebook{
prvKey:       prvKey,
balance:      balance,
contractAddr: contractAddr,
sent:         sent,
path:         path,
backend:      backend,
owner:        transactOpts.From,
contract:     chbook,
session:      session,
log:          log.New("contract", contractAddr),
```

```
}

if (contractAddr != common.Address{}) {
self.setBalanceFromBlockChain()
self.log.Trace("New chequebook initialised", "owner", self.owner, "balance", self.balance)
}
return
}

func (self *Chequebook) setBalanceFromBlockChain() {
balance, err := self.backend.BalanceAt(context.TODO(), self.contractAddr, nil)
if err != nil {
log.Error("Failed to retrieve chequebook balance", "err", err)
} else {
self.balance.Set(balance)
}
}

// LoadChequebook loads a chequebook from disk (file path).
func LoadChequebook(path string, prvKey *ecdsa.PrivateKey, backend Backend, checkBalance
bool) (self *Chequebook, err error) {
var data []byte
data, err = ioutil.ReadFile(path)
if err != nil {
return
}
self, _ = NewChequebook(path, common.Address{}, prvKey, backend)

err = json.Unmarshal(data, self)
if err != nil {
return nil, err
}
if checkBalance {
self.setBalanceFromBlockChain()
}
log.Trace("Loaded chequebook from disk", "path", path)

return
}

// chequebookFile is the JSON representation of a chequebook.
type chequebookFile struct {
```

```
Balance  string
Contract string
Owner    string
Sent     map[string]string
}

// UnmarshalJSON deserialises a chequebook.
func (self *Chequebook) UnmarshalJSON(data []byte) error {
var file chequebookFile
err := json.Unmarshal(data, &file)
if err != nil {
return err
}
_, ok := self.balance.SetString(file.Balance, 10)
if !ok {
return fmt.Errorf("cumulative amount sent: unable to convert string to big integer: %v",
file.Balance)
}
self.contractAddr = common.HexToAddress(file.Contract)
for addr, sent := range file.Sent {
self.sent[common.HexToAddress(addr)], ok = new(big.Int).SetString(sent, 10)
if !ok {
return fmt.Errorf("beneficiary %v cumulative amount sent: unable to convert string to big integer:
%v", addr, sent)
}
}
return nil
}

// MarshalJSON serialises a chequebook.
func (self *Chequebook) MarshalJSON() ([]byte, error) {
var file = &chequebookFile{
Balance:  self.balance.String(),
Contract: self.contractAddr.Hex(),
Owner:    self.owner.Hex(),
Sent:     make(map[string]string),
}
for addr, sent := range self.sent {
file.Sent[addr.Hex()] = sent.String()
}
return json.Marshal(file)
}
```

```
// Save persists the chequebook on disk, remembering balance, contract address and
// cumulative amount of funds sent for each beneficiary.
func (self *Chequebook) Save() (err error) {
data, err := json.MarshalIndent(self, "", " ")
if err != nil {
return err
}
self.log.Trace("Saving chequebook to disk", self.path)

return ioutil.WriteFile(self.path, data, os.ModePerm)
}

// Stop quits the autodeposit go routine to terminate
func (self *Chequebook) Stop() {
defer self.lock.Unlock()
self.lock.Lock()
if self.quit != nil {
close(self.quit)
self.quit = nil
}
}

// Issue creates a cheque signed by the chequebook owner's private key. The
// signer commits to a contract (one that they own), a beneficiary and amount.
func (self *Chequebook) Issue(beneficiary common.Address, amount *big.Int) (ch *Cheque, err
error) {
defer self.lock.Unlock()
self.lock.Lock()

if amount.Sign() <= 0 {
return nil, fmt.Errorf("amount must be greater than zero (%v)", amount)
}
if self.balance.Cmp(amount) < 0 {
err = fmt.Errorf("insufficient funds to issue cheque for amount: %v. balance: %v", amount,
self.balance)
} else {
var sig []byte
sent, found := self.sent[beneficiary]
if !found {
sent = new(big.Int)
self.sent[beneficiary] = sent
```

```
}
sum := new(big.Int).Set(sent)
sum.Add(sum, amount)

sig, err = crypto.Sign(sigHash(self.contractAddr, beneficiary, sum), self.prvKey)
if err == nil {
ch = &Cheque{
Contract:    self.contractAddr,
Beneficiary: beneficiary,
Amount:      sum,
Sig:         sig,
}
sent.Set(sum)
self.balance.Sub(self.balance, amount) // subtract amount from balance
}
}

// auto deposit if threshold is set and balance is less then threshold
// note this is called even if issuing cheque fails
// so we reattempt depositing
if self.threshold != nil {
if self.balance.Cmp(self.threshold) < 0 {
send := new(big.Int).Sub(self.buffer, self.balance)
self.deposit(send)
}
}

return
}

// Cash is a convenience method to cash any cheque.
func (self *Chequebook) Cash(ch *Cheque) (txhash string, err error) {
return ch.Cash(self.session)
}

// data to sign: contract address, beneficiary, cumulative amount of funds ever sent
func sigHash(contract, beneficiary common.Address, sum *big.Int) []byte {
bigamount := sum.Bytes()
if len(bigamount) > 32 {
return nil
}
var amount32 [32]byte
```

```
copy(amount32[32-len(bigamount):32], bigamount)
input := append(contract.Bytes(), beneficiary.Bytes()...)
input = append(input, amount32[:]...)
return crypto.Keccak256(input)
}

// Balance returns the current balance of the chequebook.
func (self *Chequebook) Balance() *big.Int {
defer self.lock.Unlock()
self.lock.Lock()
return new(big.Int).Set(self.balance)
}

// Owner returns the owner account of the chequebook.
func (self *Chequebook) Owner() common.Address {
return self.owner
}

// Address returns the on-chain contract address of the chequebook.
func (self *Chequebook) Address() common.Address {
return self.contractAddr
}

// Deposit deposits money to the chequebook account.
func (self *Chequebook) Deposit(amount *big.Int) (string, error) {
defer self.lock.Unlock()
self.lock.Lock()
return self.deposit(amount)
}

// deposit deposits amount to the chequebook account.
// The caller must hold self.lock.
func (self *Chequebook) deposit(amount *big.Int) (string, error) {
// since the amount is variable here, we do not use sessions
depositTransactor := bind.NewKeyedTransactor(self.prvKey)
depositTransactor.Value = amount
chbookRaw := &contract.ChequebookRaw{Contract: self.contract}
tx, err := chbookRaw.Transfer(depositTransactor)
if err != nil {
self.log.Warn("Failed to fund chequebook", "amount", amount, "balance", self.balance, "target",
self.buffer, "err", err)
return "", err
```

```
}
// assume that transaction is actually successful, we add the amount to balance right away
self.balance.Add(self.balance, amount)
self.log.Trace("Deposited funds to chequebook", "amount", amount, "balance", self.balance,
"target", self.buffer)
return tx.Hash().Hex(), nil
}

// AutoDeposit (re)sets interval time and amount which triggers sending funds to the
// chequebook. Contract backend needs to be set if threshold is not less than buffer, then
// deposit will be triggered on every new cheque issued.
func (self *Chequebook) AutoDeposit(interval time.Duration, threshold, buffer *big.Int) {
defer self.lock.Unlock()
self.lock.Lock()
self.threshold = threshold
self.buffer = buffer
self.autoDeposit(interval)
}

// autoDeposit starts a goroutine that periodically sends funds to the chequebook
// contract caller holds the lock the go routine terminates if Chequebook.quit is closed.
func (self *Chequebook) autoDeposit(interval time.Duration) {
if self.quit != nil {
close(self.quit)
self.quit = nil
}
// if threshold >= balance autodeposit after every cheque issued
if interval == time.Duration(0) || self.threshold != nil && self.buffer != nil &&
self.threshold.Cmp(self.buffer) >= 0 {
return
}

ticker := time.NewTicker(interval)
self.quit = make(chan bool)
quit := self.quit
go func() {
FOR:
for {
select {
case <-quit:
break FOR
case <-ticker.C:
```

```
self.lock.Lock()
if self.balance.Cmp(self.buffer) < 0 {
amount := new(big.Int).Sub(self.buffer, self.balance)
txhash, err := self.deposit(amount)
if err == nil {
self.txhash = txhash
}
}
self.lock.Unlock()
}
}
}()
return
}

// Outbox can issue cheques from a single contract to a single beneficiary.
type Outbox struct {
chequeBook  *Chequebook
beneficiary common.Address
}

// NewOutbox creates an outbox.
func NewOutbox(chbook *Chequebook, beneficiary common.Address) *Outbox {
return &Outbox{chbook, beneficiary}
}

// Issue creates cheque.
func (self *Outbox) Issue(amount *big.Int) (swap.Promise, error) {
return self.chequeBook.Issue(self.beneficiary, amount)
}

// AutoDeposit enables auto-deposits on the underlying chequebook.
func (self *Outbox) AutoDeposit(interval time.Duration, threshold, buffer *big.Int) {
self.chequeBook.AutoDeposit(interval, threshold, buffer)
}

// Stop helps satisfy the swap.OutPayment interface.
func (self *Outbox) Stop() {}

// String implements fmt.Stringer.
func (self *Outbox) String() string {
return fmt.Sprintf("chequebook: %v, beneficiary: %s, balance: %v",
```

```
self.chequeBook.Address().Hex(), self.beneficiary.Hex(), self.chequeBook.Balance())
}

// Inbox can deposit, verify and cash cheques from a single contract to a single
// beneficiary. It is the incoming payment handler for peer to peer micropayments.
type Inbox struct {
lock        sync.Mutex
contract    common.Address             // peer's chequebook contract
beneficiary common.Address             // local peer's receiving address
sender      common.Address             // local peer's address to send cashing tx from
signer      *ecdsa.PublicKey           // peer's public key
txhash      string                    // tx hash of last cashing tx
abigen      bind.ContractBackend       // blockchain API
session     *contract.ChequebookSession // abi contract backend with tx opts
quit        chan bool                  // when closed causes autocash to stop
maxUncashed *big.Int                   // threshold that triggers autocashing
cashed      *big.Int                   // cumulative amount cashed
cheque      *Cheque                    // last cheque, nil if none yet received
log         log.Logger                 // contextual logger with the contract address embedded
}

// NewInbox creates an Inbox. An Inboxes is not persisted, the cumulative sum is updated
// from blockchain when first cheque is received.
func NewInbox(prvKey *ecdsa.PrivateKey, contractAddr, beneficiary common.Address, signer
*ecdsa.PublicKey, abigen bind.ContractBackend) (self *Inbox, err error) {
if signer == nil {
return nil, fmt.Errorf("signer is null")
}
chbook, err := contract.NewChequebook(contractAddr, abigen)
if err != nil {
return nil, err
}
transactOpts := bind.NewKeyedTransactor(prvKey)
transactOpts.GasLimit = gasToCash
session := &contract.ChequebookSession{
Contract:     chbook,
TransactOpts: *transactOpts,
}
sender := transactOpts.From

self = &Inbox{
contract:    contractAddr,
```

```
beneficiary: beneficiary,
sender:      sender,
signer:      signer,
session:     session,
cashed:      new(big.Int).Set(common.Big0),
log:         log.New("contract", contractAddr),
}
self.log.Trace("New chequebook inbox initialized", "beneficiary", self.beneficiary, "signer",
hexutil.Bytes(crypto.FromECDSAPub(signer)))
return
}

func (self *Inbox) String() string {
return fmt.Sprintf("chequebook: %v, beneficiary: %s, balance: %v", self.contract.Hex(),
self.beneficiary.Hex(), self.cheque.Amount)
}

// Stop quits the autocash goroutine.
func (self *Inbox) Stop() {
defer self.lock.Unlock()
self.lock.Lock()
if self.quit != nil {
close(self.quit)
self.quit = nil
}
}

// Cash attempts to cash the current cheque.
func (self *Inbox) Cash() (txhash string, err error) {
if self.cheque != nil {
txhash, err = self.cheque.Cash(self.session)
self.log.Trace("Cashing in chequebook cheque", "amount", self.cheque.Amount, "beneficiary",
self.beneficiary)
self.cashed = self.cheque.Amount
}
return
}

// AutoCash (re)sets maximum time and amount which triggers cashing of the last uncashed
// cheque if maxUncashed is set to 0, then autocash on receipt.
func (self *Inbox) AutoCash(cashInterval time.Duration, maxUncashed *big.Int) {
defer self.lock.Unlock()
```

```
self.lock.Lock()
self.maxUncashed = maxUncashed
self.autoCash(cashInterval)
}

// autoCash starts a loop that periodically clears the last cheque
// if the peer is trusted. Clearing period could be 24h or a week.
// The caller must hold self.lock.
func (self *Inbox) autoCash(cashInterval time.Duration) {
if self.quit != nil {
close(self.quit)
self.quit = nil
}
// if maxUncashed is set to 0, then autocash on receipt
if cashInterval == time.Duration(0) || self.maxUncashed != nil && self.maxUncashed.Sign() == 0 {
return
}

ticker := time.NewTicker(cashInterval)
self.quit = make(chan bool)
quit := self.quit
go func() {
FOR:
for {
select {
case <-quit:
break FOR
case <-ticker.C:
self.lock.Lock()
if self.cheque != nil && self.cheque.Amount.Cmp(self.cashed) != 0 {
txhash, err := self.Cash()
if err == nil {
self.txhash = txhash
}
}
self.lock.Unlock()
}
}
}()
return
}
```

```
// Receive is called to deposit the latest cheque to the incoming Inbox.
// The given promise must be a *Cheque.
func (self *Inbox) Receive(promise swap.Promise) (*big.Int, error) {
ch := promise.(*Cheque)

defer self.lock.Unlock()
self.lock.Lock()

var sum *big.Int
if self.cheque == nil {
// the sum is checked against the blockchain once a cheque is received
tally, err := self.session.Sent(self.beneficiary)
if err != nil {
return nil, fmt.Errorf("inbox: error calling backend to set amount: %v", err)
}
sum = tally
} else {
sum = self.cheque.Amount
}

amount, err := ch.Verify(self.signer, self.contract, self.beneficiary, sum)
var uncashed *big.Int
if err == nil {
self.cheque = ch

if self.maxUncashed != nil {
uncashed = new(big.Int).Sub(ch.Amount, self.cashed)
if self.maxUncashed.Cmp(uncashed) < 0 {
self.Cash()
}
}
self.log.Trace("Received cheque in chequebook inbox", "amount", amount, "uncashed", uncashed)
}

return amount, err
}

// Verify verifies cheque for signer, contract, beneficiary, amount, valid signature.
func (self *Cheque) Verify(signerKey *ecdsa.PublicKey, contract, beneficiary common.Address,
sum *big.Int) (*big.Int, error) {
log.Trace("Verifying chequebook cheque", "cheque", self, "sum", sum)
if sum == nil {
```

```
return nil, fmt.Errorf("invalid amount")
}

if self.Beneficiary != beneficiary {
return nil, fmt.Errorf("beneficiary mismatch: %v != %v", self.Beneficiary.Hex(), beneficiary.Hex())
}
if self.Contract != contract {
return nil, fmt.Errorf("contract mismatch: %v != %v", self.Contract.Hex(), contract.Hex())
}

amount := new(big.Int).Set(self.Amount)
if sum != nil {
amount.Sub(amount, sum)
if amount.Sign() <= 0 {
return nil, fmt.Errorf("incorrect amount: %v <= 0", amount)
}
}

pubKey, err := crypto.SigToPub(sigHash(self.Contract, beneficiary, self.Amount), self.Sig)
if err != nil {
return nil, fmt.Errorf("invalid signature: %v", err)
}
if !bytes.Equal(crypto.FromECDSAPub(pubKey), crypto.FromECDSAPub(signerKey)) {
return nil, fmt.Errorf("signer mismatch: %x != %x", crypto.FromECDSAPub(pubKey),
crypto.FromECDSAPub(signerKey))
}
return amount, nil
}

// v/r/s representation of signature
func sig2vrs(sig []byte) (v byte, r, s [32]byte) {
v = sig[64] + 27
copy(r[:], sig[:32])
copy(s[:], sig[32:64])
return
}

// Cash cashes the cheque by sending an Ethereum transaction.
func (self *Cheque) Cash(session *contract.ChequebookSession) (string, error) {
v, r, s := sig2vrs(self.Sig)
tx, err := session.Cash(self.Beneficiary, self.Amount, v, r, s)
if err != nil {
```

```
return "", err
}
return tx.Hash().Hex(), nil
}

// ValidateCode checks that the on-chain code at address matches the expected chequebook
// contract code. This is used to detect suicided chequebooks.
func ValidateCode(ctx context.Context, b Backend, address common.Address) (ok bool, err error)
{
code, err := b.CodeAt(ctx, address, nil)
if err != nil {
return false, err
}
return bytes.Equal(code, common.FromHex(contract.ContractDeployedCode)), nil
}

46:F:\git\coin\ethereum\go-ethereum\contracts\chequebook\cheque_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package chequebook

import (
"crypto/ecdsa"
"math/big"
"os"
"path/filepath"
"testing"
"time"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/accounts/abi/bind/backends"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/contracts/chequebook/contract"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/crypto"
)

var (
key0, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
key1, _ =
```

```
crypto.HexToECDSA("8a1f9a8f95be41cd7ccb6168179afb4504aefe388d1e14474d32c45c72ce7b
7a")
key2, _ =
crypto.HexToECDSA("49a7b37aa6f6645917e7b807e9d1c00d4fa71f18343b0d4122a4d2df64dd6f
ee")
addr0   = crypto.PubkeyToAddress(key0.PublicKey)
addr1   = crypto.PubkeyToAddress(key1.PublicKey)
addr2   = crypto.PubkeyToAddress(key2.PublicKey)
)

func newTestBackend() *backends.SimulatedBackend {
return backends.NewSimulatedBackend(core.GenesisAlloc{
addr0: {Balance: big.NewInt(1000000000)},
addr1: {Balance: big.NewInt(1000000000)},
addr2: {Balance: big.NewInt(1000000000)},
})
}

func deploy(prvKey *ecdsa.PrivateKey, amount *big.Int, backend *backends.SimulatedBackend)
(common.Address, error) {
deployTransactor := bind.NewKeyedTransactor(prvKey)
deployTransactor.Value = amount
addr, _, _, err := contract.DeployChequebook(deployTransactor, backend)
if err != nil {
return common.Address{}, err
}
backend.Commit()
return addr, nil
}

func TestIssueAndReceive(t *testing.T) {
path := filepath.Join(os.TempDir(), "chequebook-test.json")
backend := newTestBackend()
addr0, err := deploy(key0, big.NewInt(0), backend)
if err != nil {
t.Fatalf("deploy contract: expected no error, got %v", err)
}
chbook, err := NewChequebook(path, addr0, key0, backend)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
chbook.sent[addr1] = new(big.Int).SetUint64(42)
```

```
amount := common.Big1

if _, err = chbook.Issue(addr1, amount); err == nil {
t.Fatalf("expected insufficient funds error, got none")
}

chbook.balance = new(big.Int).Set(common.Big1)
if chbook.Balance().Cmp(common.Big1) != 0 {
t.Fatalf("expected: %v, got %v", "0", chbook.Balance())
}

ch, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

if chbook.Balance().Sign() != 0 {
t.Errorf("expected: %v, got %v", "0", chbook.Balance())
}

chbox, err := NewInbox(key1, addr0, addr1, &key0.PublicKey, backend)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

received, err := chbox.Receive(ch)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

if received.Cmp(big.NewInt(43)) != 0 {
t.Errorf("expected: %v, got %v", "43", received)
}

}

func TestCheckbookFile(t *testing.T) {
path := filepath.Join(os.TempDir(), "chequebook-test.json")
backend := newTestBackend()
chbook, err := NewChequebook(path, addr0, key0, backend)
if err != nil {
t.Fatalf("expected no error, got %v", err)
```

```
}
chbook.sent[addr1] = new(big.Int).SetUint64(42)
chbook.balance = new(big.Int).Set(common.Big1)

chbook.Save()

chbook, err = LoadChequebook(path, key0, backend, false)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
if chbook.Balance().Cmp(common.Big1) != 0 {
t.Errorf("expected: %v, got %v", "0", chbook.Balance())
}

var ch *Cheque
if ch, err = chbook.Issue(addr1, common.Big1); err != nil {
t.Fatalf("expected no error, got %v", err)
}
if ch.Amount.Cmp(new(big.Int).SetUint64(43)) != 0 {
t.Errorf("expected: %v, got %v", "0", ch.Amount)
}

err = chbook.Save()
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
}

func TestVerifyErrors(t *testing.T) {
path0 := filepath.Join(os.TempDir(), "chequebook-test-0.json")
backend := newTestBackend()
contr0, err := deploy(key0, common.Big2, backend)
if err != nil {
t.Errorf("expected no error, got %v", err)
}
chbook0, err := NewChequebook(path0, contr0, key0, backend)
if err != nil {
t.Errorf("expected no error, got %v", err)
}

path1 := filepath.Join(os.TempDir(), "chequebook-test-1.json")
contr1, _ := deploy(key1, common.Big2, backend)
```

```
chbook1, err := NewChequebook(path1, contr1, key1, backend)
if err != nil {
t.Errorf("expected no error, got %v", err)
}

chbook0.sent[addr1] = new(big.Int).SetUint64(42)
chbook0.balance = new(big.Int).Set(common.Big2)
chbook1.balance = new(big.Int).Set(common.Big1)
amount := common.Big1
ch0, err := chbook0.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

time.Sleep(5)
chbox, err := NewInbox(key1, contr0, addr1, &key0.PublicKey, backend)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

received, err := chbox.Receive(ch0)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

if received.Cmp(big.NewInt(43)) != 0 {
t.Errorf("expected: %v, got %v", "43", received)
}

ch1, err := chbook0.Issue(addr2, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

received, err = chbox.Receive(ch1)
t.Logf("correct error: %v", err)
if err == nil {
t.Fatalf("expected receiver error, got none")
}

ch2, err := chbook1.Issue(addr1, amount)
if err != nil {
```

```
t.Fatalf("expected no error, got %v", err)
}
received, err = chbox.Receive(ch2)
t.Logf("correct error: %v", err)
if err == nil {
t.Fatalf("expected sender error, got none")
}

_, err = chbook1.Issue(addr1, new(big.Int).SetInt64(-1))
t.Logf("correct error: %v", err)
if err == nil {
t.Fatalf("expected incorrect amount error, got none")
}

received, err = chbox.Receive(ch0)
t.Logf("correct error: %v", err)
if err == nil {
t.Fatalf("expected incorrect amount error, got none")
}

}

func TestDeposit(t *testing.T) {
path0 := filepath.Join(os.TempDir(), "chequebook-test-0.json")
backend := newTestBackend()
contr0, _ := deploy(key0, new(big.Int), backend)

chbook, err := NewChequebook(path0, contr0, key0, backend)
if err != nil {
t.Errorf("expected no error, got %v", err)
}

balance := new(big.Int).SetUint64(42)
chbook.Deposit(balance)
backend.Commit()
if chbook.balance.Cmp(balance) != 0 {
t.Fatalf("expected balance %v, got %v", balance, chbook.balance)
}

amount := common.Big1
_, err = chbook.Issue(addr1, amount)
if err != nil {
```

```
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
exp := new(big.Int).SetUint64(41)
if chbook.balance.Cmp(exp) != 0 {
t.Fatalf("expected balance %v, got %v", exp, chbook.balance)
}

// autodeposit on each issue
chbook.AutoDeposit(0, balance, balance)
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
if chbook.balance.Cmp(balance) != 0 {
t.Fatalf("expected balance %v, got %v", balance, chbook.balance)
}

// autodeposit off
chbook.AutoDeposit(0, common.Big0, balance)
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

exp = new(big.Int).SetUint64(40)
if chbook.balance.Cmp(exp) != 0 {
t.Fatalf("expected balance %v, got %v", exp, chbook.balance)
}
```

```
// autodeposit every 10ms if new cheque issued
interval := 30 * time.Millisecond
chbook.AutoDeposit(interval, common.Big1, balance)
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

exp = new(big.Int).SetUint64(38)
if chbook.balance.Cmp(exp) != 0 {
t.Fatalf("expected balance %v, got %v", exp, chbook.balance)
}

time.Sleep(3 * interval)
backend.Commit()
if chbook.balance.Cmp(balance) != 0 {
t.Fatalf("expected balance %v, got %v", balance, chbook.balance)
}

exp = new(big.Int).SetUint64(40)
chbook.AutoDeposit(4*interval, exp, balance)
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
time.Sleep(3 * interval)
backend.Commit()
if chbook.balance.Cmp(exp) != 0 {
t.Fatalf("expected balance %v, got %v", exp, chbook.balance)
}
```

```
_, err = chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
time.Sleep(1 * interval)
backend.Commit()

if chbook.balance.Cmp(balance) != 0 {
t.Fatalf("expected balance %v, got %v", balance, chbook.balance)
}

chbook.AutoDeposit(1*interval, common.Big0, balance)
chbook.Stop()

_, err = chbook.Issue(addr1, common.Big1)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

_, err = chbook.Issue(addr1, common.Big2)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

time.Sleep(1 * interval)
backend.Commit()

exp = new(big.Int).SetUint64(39)
if chbook.balance.Cmp(exp) != 0 {
t.Fatalf("expected balance %v, got %v", exp, chbook.balance)
}

}

func TestCash(t *testing.T) {
path := filepath.Join(os.TempDir(), "chequebook-test.json")
backend := newTestBackend()
contr0, _ := deploy(key0, common.Big2, backend)

chbook, err := NewChequebook(path, contr0, key0, backend)
if err != nil {
```

```
t.Errorf("expected no error, got %v", err)
}
chbook.sent[addr1] = new(big.Int).SetUint64(42)
amount := common.Big1
chbook.balance = new(big.Int).Set(common.Big1)
ch, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
chbox, err := NewInbox(key1, contr0, addr1, &key0.PublicKey, backend)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

// cashing latest cheque
if _, err = chbox.Receive(ch); err != nil {
t.Fatalf("expected no error, got %v", err)
}
if _, err = ch.Cash(chbook.session); err != nil {
t.Fatal("Cash failed:", err)
}
backend.Commit()

chbook.balance = new(big.Int).Set(common.Big3)
ch0, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
ch1, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

interval := 10 * time.Millisecond
// setting autocash with interval of 10ms
chbox.AutoCash(interval, nil)
_, err = chbox.Receive(ch0)
if err != nil {
t.Fatalf("expected no error, got %v", err)
```

```
}
_, err = chbox.Receive(ch1)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()
// after 3x interval time and 2 cheques received, exactly one cashing tx is sent
time.Sleep(4 * interval)
backend.Commit()

// after stopping autocash no more tx are sent
ch2, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
chbox.Stop()
_, err = chbox.Receive(ch2)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
time.Sleep(2 * interval)
backend.Commit()

// autocash below 1
chbook.balance = big.NewInt(2)
chbox.AutoCash(0, common.Big1)

ch3, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

ch4, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

_, err = chbox.Receive(ch3)
if err != nil {
t.Fatalf("expected no error, got %v", err)
```

```
}
backend.Commit()
_, err = chbox.Receive(ch4)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

// autochash on receipt when maxUncashed is 0
chbook.balance = new(big.Int).Set(common.Big2)
chbox.AutoCash(0, common.Big0)

ch5, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

ch6, err := chbook.Issue(addr1, amount)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}

_, err = chbox.Receive(ch5)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

_, err = chbox.Receive(ch6)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
backend.Commit()

}

47:F:\git\coin\ethereum\go-ethereum\contracts\chequebook\contract\chequebook.go

// ChequebookABI is the input ABI used to generate the binding from.
const ChequebookABI =
`[{"constant":false,"inputs":[],"name":"kill","outputs":[],"type":"function"},{"constant":true,"inputs":[{"n
```

ame":"","type":"address"}],"name":"sent","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"
constant":false,"inputs":[{"name":"beneficiary","type":"address"},{"name":"amount","type":"uint256"}
,{"name":"sig_v","type":"uint8"},{"name":"sig_r","type":"bytes32"},{"name":"sig_s","type":"bytes32"}]
,"name":"cash","outputs":[],"type":"function"},{"anonymous":false,"inputs":[{"indexed":false,"name":"
deadbeat","type":"address"}],"name":"Overdraft","type":"event"}]`

```
// ChequebookBin is the compiled bytecode used for deploying new contracts.
const ChequebookBin =
`0x6060604052600080546001600160a060020a0319163317905561ff806100246000396000f360606
0405260e060020a600035046341c0e1b581146100315780637bf786f814610059578063fbf788d61
4610071575b005b61002f60005433600160a060020a0390811691161415610bd5760005460016
0a060020a0316ff5b6100ab60043560016020526000908152604090205481565b61002f60043560
2435604435606435608435600160a060020a038516600090815260016020526040812054851161
00bf575b505050505050565b60408051918252519081900360200190f35b565b50604080516c010
0000000000000000000000030600160a060020a039081168202835288160260148201526028810
1869052815190819003604801812080825260ff8616602083810191909152828401869052606083
0185905292519092600192608081810193918290030181600086616da5a03f115610002575050
6040515160005460016a060020a0390811691161461015a576100a3565b600160a060020a0386
8116600090815260016020526040902054309091163190860310610b3576040600081812087905
59051600160a060020a0388169190819081818181818881f193505050501515610100a357610002
565b60005460408051600160a060020a0392909216825251 7f2250e2993c15843b32621c89447cc
589ee7a9f049c026986e545d3c2c0c6f97891819003602001 90a185600160a060020a0316ff`

// DeployChequebook deploys a new Ethereum contract, binding an instance of Chequebook to it.
func DeployChequebook(auth *bind.TransactOpts, backend bind.ContractBackend)
(common.Address, *types.Transaction, *Chequebook, error) {
parsed, err := abi.JSON(strings.NewReader(ChequebookABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed,
common.FromHex(ChequebookBin), backend)
if err != nil {
return common.Address{}, nil, nil, err
}
return address, tx, &Chequebook{ChequebookCaller: ChequebookCaller{contract: contract},
ChequebookTransactor: ChequebookTransactor{contract: contract}}, nil
}

// Chequebook is an auto generated Go binding around an Ethereum contract.
type Chequebook struct {
ChequebookCaller     // Read-only binding to the contract
```

```
ChequebookTransactor // Write-only binding to the contract
}

// ChequebookCaller is an auto generated read-only Go binding around an Ethereum contract.
type ChequebookCaller struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ChequebookTransactor is an auto generated write-only Go binding around an Ethereum
contract.
type ChequebookTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ChequebookSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type ChequebookSession struct {
Contract     *Chequebook       // Generic contract binding to set the session for
CallOpts     bind.CallOpts     // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// ChequebookCallerSession is an auto generated read-only Go binding around an Ethereum
contract,
// with pre-set call options.
type ChequebookCallerSession struct {
Contract *ChequebookCaller // Generic contract caller binding to set the session for
CallOpts bind.CallOpts     // Call options to use throughout this session
}

// ChequebookTransactorSession is an auto generated write-only Go binding around an Ethereum
contract,
// with pre-set transact options.
type ChequebookTransactorSession struct {
Contract     *ChequebookTransactor // Generic contract transactor binding to set the session for
TransactOpts bind.TransactOpts     // Transaction auth options to use throughout this session
}

// ChequebookRaw is an auto generated low-level Go binding around an Ethereum contract.
type ChequebookRaw struct {
Contract *Chequebook // Generic contract binding to access the raw methods on
}
```

```
// ChequebookCallerRaw is an auto generated low-level read-only Go binding around an
Ethereum contract.
type ChequebookCallerRaw struct {
Contract *ChequebookCaller // Generic read-only contract binding to access the raw methods on
}

// ChequebookTransactorRaw is an auto generated low-level write-only Go binding around an
Ethereum contract.
type ChequebookTransactorRaw struct {
Contract *ChequebookTransactor // Generic write-only contract binding to access the raw methods
on
}

// NewChequebook creates a new instance of Chequebook, bound to a specific deployed contract.
func NewChequebook(address common.Address, backend bind.ContractBackend) (*Chequebook,
error) {
contract, err := bindChequebook(address, backend, backend)
if err != nil {
return nil, err
}
return &Chequebook{ChequebookCaller: ChequebookCaller{contract: contract},
ChequebookTransactor: ChequebookTransactor{contract: contract}}, nil
}

// NewChequebookCaller creates a new read-only instance of Chequebook, bound to a specific
deployed contract.
func NewChequebookCaller(address common.Address, caller bind.ContractCaller)
(*ChequebookCaller, error) {
contract, err := bindChequebook(address, caller, nil)
if err != nil {
return nil, err
}
return &ChequebookCaller{contract: contract}, nil
}

// NewChequebookTransactor creates a new write-only instance of Chequebook, bound to a
specific deployed contract.
func NewChequebookTransactor(address common.Address, transactor bind.ContractTransactor)
(*ChequebookTransactor, error) {
contract, err := bindChequebook(address, nil, transactor)
if err != nil {
```

```
return nil, err
}
return &ChequebookTransactor{contract: contract}, nil
}

// bindChequebook binds a generic wrapper to an already deployed contract.
func bindChequebook(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(ChequebookABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Chequebook *ChequebookRaw) Call(opts *bind.CallOpts, result interface{}, method string,
params ...interface{}) error {
return _Chequebook.Contract.ChequebookCaller.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Chequebook *ChequebookRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction,
error) {
return _Chequebook.Contract.ChequebookTransactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_Chequebook *ChequebookRaw) Transact(opts *bind.TransactOpts, method string, params
...interface{}) (*types.Transaction, error) {
return _Chequebook.Contract.ChequebookTransactor.contract.Transact(opts, method, params...)
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Chequebook *ChequebookCallerRaw) Call(opts *bind.CallOpts, result interface{}, method
```

```
string, params ...interface{}) error {
return _Chequebook.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Chequebook *ChequebookTransactorRaw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _Chequebook.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_Chequebook *ChequebookTransactorRaw) Transact(opts *bind.TransactOpts, method
string, params ...interface{}) (*types.Transaction, error) {
return _Chequebook.Contract.contract.Transact(opts, method, params...)
}

// Sent is a free data retrieval call binding the contract method 0x7bf786f8.
//
// Solidity: function sent( address) constant returns(uint256)
func (_Chequebook *ChequebookCaller) Sent(opts *bind.CallOpts, arg0 common.Address)
(*big.Int, error) {
var (
ret0 = new(*big.Int)
)
out := ret0
err := _Chequebook.contract.Call(opts, out, "sent", arg0)
return *ret0, err
}

// Sent is a free data retrieval call binding the contract method 0x7bf786f8.
//
// Solidity: function sent( address) constant returns(uint256)
func (_Chequebook *ChequebookSession) Sent(arg0 common.Address) (*big.Int, error) {
return _Chequebook.Contract.Sent(&_Chequebook.CallOpts, arg0)
}

// Sent is a free data retrieval call binding the contract method 0x7bf786f8.
//
// Solidity: function sent( address) constant returns(uint256)
func (_Chequebook *ChequebookCallerSession) Sent(arg0 common.Address) (*big.Int, error) {
return _Chequebook.Contract.Sent(&_Chequebook.CallOpts, arg0)
```

```
}

// Cash is a paid mutator transaction binding the contract method 0xfbf788d6.
//
// Solidity: function cash(beneficiary address, amount uint256, sig_v uint8, sig_r bytes32, sig_s bytes32) returns()
func (_Chequebook *ChequebookTransactor) Cash(opts *bind.TransactOpts, beneficiary common.Address, amount *big.Int, sig_v uint8, sig_r [32]byte, sig_s [32]byte) (*types.Transaction, error) {
return _Chequebook.contract.Transact(opts, "cash", beneficiary, amount, sig_v, sig_r, sig_s)
}

// Cash is a paid mutator transaction binding the contract method 0xfbf788d6.
//
// Solidity: function cash(beneficiary address, amount uint256, sig_v uint8, sig_r bytes32, sig_s bytes32) returns()
func (_Chequebook *ChequebookSession) Cash(beneficiary common.Address, amount *big.Int, sig_v uint8, sig_r [32]byte, sig_s [32]byte) (*types.Transaction, error) {
return _Chequebook.Contract.Cash(&_Chequebook.TransactOpts, beneficiary, amount, sig_v, sig_r, sig_s)
}

// Cash is a paid mutator transaction binding the contract method 0xfbf788d6.
//
// Solidity: function cash(beneficiary address, amount uint256, sig_v uint8, sig_r bytes32, sig_s bytes32) returns()
func (_Chequebook *ChequebookTransactorSession) Cash(beneficiary common.Address, amount *big.Int, sig_v uint8, sig_r [32]byte, sig_s [32]byte) (*types.Transaction, error) {
return _Chequebook.Contract.Cash(&_Chequebook.TransactOpts, beneficiary, amount, sig_v, sig_r, sig_s)
}

// Kill is a paid mutator transaction binding the contract method 0x41c0e1b5.
//
// Solidity: function kill() returns()
func (_Chequebook *ChequebookTransactor) Kill(opts *bind.TransactOpts) (*types.Transaction, error) {
return _Chequebook.contract.Transact(opts, "kill")
}

// Kill is a paid mutator transaction binding the contract method 0x41c0e1b5.
//
```

```
// Solidity: function kill() returns()
func (_Chequebook *ChequebookSession) Kill() (*types.Transaction, error) {
return _Chequebook.Contract.Kill(&_Chequebook.TransactOpts)
}

// Kill is a paid mutator transaction binding the contract method 0x41c0e1b5.
//
// Solidity: function kill() returns()
func (_Chequebook *ChequebookTransactorSession) Kill() (*types.Transaction, error) {
return _Chequebook.Contract.Kill(&_Chequebook.TransactOpts)
}

// MortalABI is the input ABI used to generate the binding from.
const MortalABI = `[{"constant":false,"inputs":[],"name":"kill","outputs":[],"type":"function"}]`

// MortalBin is the compiled bytecode used for deploying new contracts.
const MortalBin =
`0x606060405260008054600160a060020a0319163317905560 5c806022600039 6000f3606060 40
5260e060020a600035046341c0e1b58114601a575b005b60186000543373ffffffffffffffffffffffffffffffffffff
ffff9081169116141560 5a5760005473ffffffffffffffffffffffffffffffffffffffff16ff5b56`

// DeployMortal deploys a new Ethereum contract, binding an instance of Mortal to it.
func DeployMortal(auth *bind.TransactOpts, backend bind.ContractBackend) (common.Address,
*types.Transaction, *Mortal, error) {
parsed, err := abi.JSON(strings.NewReader(MortalABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed, common.FromHex(MortalBin),
backend)
if err != nil {
return common.Address{}, nil, nil, err
}
return address, tx, &Mortal{MortalCaller: MortalCaller{contract: contract}, MortalTransactor:
MortalTransactor{contract: contract}}, nil
}

// Mortal is an auto generated Go binding around an Ethereum contract.
type Mortal struct {
MortalCaller     // Read-only binding to the contract
MortalTransactor // Write-only binding to the contract
}
```

```
// MortalCaller is an auto generated read-only Go binding around an Ethereum contract.
type MortalCaller struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// MortalTransactor is an auto generated write-only Go binding around an Ethereum contract.
type MortalTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// MortalSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type MortalSession struct {
Contract     *Mortal           // Generic contract binding to set the session for
CallOpts     bind.CallOpts     // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// MortalCallerSession is an auto generated read-only Go binding around an Ethereum contract,
// with pre-set call options.
type MortalCallerSession struct {
Contract *MortalCaller // Generic contract caller binding to set the session for
CallOpts bind.CallOpts // Call options to use throughout this session
}

// MortalTransactorSession is an auto generated write-only Go binding around an Ethereum
contract,
// with pre-set transact options.
type MortalTransactorSession struct {
Contract     *MortalTransactor // Generic contract transactor binding to set the session for
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// MortalRaw is an auto generated low-level Go binding around an Ethereum contract.
type MortalRaw struct {
Contract *Mortal // Generic contract binding to access the raw methods on
}

// MortalCallerRaw is an auto generated low-level read-only Go binding around an Ethereum
contract.
type MortalCallerRaw struct {
```

```
Contract *MortalCaller // Generic read-only contract binding to access the raw methods on
}

// MortalTransactorRaw is an auto generated low-level write-only Go binding around an Ethereum
contract.
type MortalTransactorRaw struct {
Contract *MortalTransactor // Generic write-only contract binding to access the raw methods on
}

// NewMortal creates a new instance of Mortal, bound to a specific deployed contract.
func NewMortal(address common.Address, backend bind.ContractBackend) (*Mortal, error) {
contract, err := bindMortal(address, backend, backend)
if err != nil {
return nil, err
}
return &Mortal{MortalCaller: MortalCaller{contract: contract}, MortalTransactor:
MortalTransactor{contract: contract}}, nil
}

// NewMortalCaller creates a new read-only instance of Mortal, bound to a specific deployed
contract.
func NewMortalCaller(address common.Address, caller bind.ContractCaller) (*MortalCaller, error)
{
contract, err := bindMortal(address, caller, nil)
if err != nil {
return nil, err
}
return &MortalCaller{contract: contract}, nil
}

// NewMortalTransactor creates a new write-only instance of Mortal, bound to a specific deployed
contract.
func NewMortalTransactor(address common.Address, transactor bind.ContractTransactor)
(*MortalTransactor, error) {
contract, err := bindMortal(address, nil, transactor)
if err != nil {
return nil, err
}
return &MortalTransactor{contract: contract}, nil
}

// bindMortal binds a generic wrapper to an already deployed contract.
```

```
func bindMortal(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(MortalABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Mortal *MortalRaw) Call(opts *bind.CallOpts, result interface{}, method string, params
...interface{}) error {
return _Mortal.Contract.MortalCaller.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Mortal *MortalRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction, error) {
return _Mortal.Contract.MortalTransactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_Mortal *MortalRaw) Transact(opts *bind.TransactOpts, method string, params ...interface{})
(*types.Transaction, error) {
return _Mortal.Contract.MortalTransactor.contract.Transact(opts, method, params...)
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Mortal *MortalCallerRaw) Call(opts *bind.CallOpts, result interface{}, method string, params
...interface{}) error {
return _Mortal.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Mortal *MortalTransactorRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction,
```

```
error) {
return _Mortal.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_Mortal *MortalTransactorRaw) Transact(opts *bind.TransactOpts, method string, params
...interface{}) (*types.Transaction, error) {
return _Mortal.Contract.contract.Transact(opts, method, params...)
}

// Kill is a paid mutator transaction binding the contract method 0x41c0e1b5.
//
// Solidity: function kill() returns()
func (_Mortal *MortalTransactor) Kill(opts *bind.TransactOpts) (*types.Transaction, error) {
return _Mortal.contract.Transact(opts, "kill")
}

// Kill is a paid mutator transaction binding the contract method 0x41c0e1b5.
//
// Solidity: function kill() returns()
func (_Mortal *MortalSession) Kill() (*types.Transaction, error) {
return _Mortal.Contract.Kill(&_Mortal.TransactOpts)
}

// Kill is a paid mutator transaction binding the contract method 0x41c0e1b5.
//
// Solidity: function kill() returns()
func (_Mortal *MortalTransactorSession) Kill() (*types.Transaction, error) {
return _Mortal.Contract.Kill(&_Mortal.TransactOpts)
}

// OwnedABI is the input ABI used to generate the binding from.
const OwnedABI = `[{"inputs":[],"type":"constructor"}]`

// OwnedBin is the compiled bytecode used for deploying new contracts.
const OwnedBin =
`0x6060604052600080546001600160a060020a0319163317905560068060226000396000f360606040
5200`

// DeployOwned deploys a new Ethereum contract, binding an instance of Owned to it.
func DeployOwned(auth *bind.TransactOpts, backend bind.ContractBackend) (common.Address,
*types.Transaction, *Owned, error) {
```

```
parsed, err := abi.JSON(strings.NewReader(OwnedABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed, common.FromHex(OwnedBin),
backend)
if err != nil {
return common.Address{}, nil, nil, err
}
return address, tx, &Owned{OwnedCaller: OwnedCaller{contract: contract}, OwnedTransactor:
OwnedTransactor{contract: contract}}, nil
}

// Owned is an auto generated Go binding around an Ethereum contract.
type Owned struct {
OwnedCaller     // Read-only binding to the contract
OwnedTransactor // Write-only binding to the contract
}

// OwnedCaller is an auto generated read-only Go binding around an Ethereum contract.
type OwnedCaller struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// OwnedTransactor is an auto generated write-only Go binding around an Ethereum contract.
type OwnedTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// OwnedSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type OwnedSession struct {
Contract     *Owned            // Generic contract binding to set the session for
CallOpts     bind.CallOpts     // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// OwnedCallerSession is an auto generated read-only Go binding around an Ethereum contract,
// with pre-set call options.
type OwnedCallerSession struct {
Contract *OwnedCaller  // Generic contract caller binding to set the session for
CallOpts bind.CallOpts // Call options to use throughout this session
```

```
}

// OwnedTransactorSession is an auto generated write-only Go binding around an Ethereum
contract,
// with pre-set transact options.
type OwnedTransactorSession struct {
Contract     *OwnedTransactor  // Generic contract transactor binding to set the session for
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// OwnedRaw is an auto generated low-level Go binding around an Ethereum contract.
type OwnedRaw struct {
Contract *Owned // Generic contract binding to access the raw methods on
}

// OwnedCallerRaw is an auto generated low-level read-only Go binding around an Ethereum
contract.
type OwnedCallerRaw struct {
Contract *OwnedCaller // Generic read-only contract binding to access the raw methods on
}

// OwnedTransactorRaw is an auto generated low-level write-only Go binding around an Ethereum
contract.
type OwnedTransactorRaw struct {
Contract *OwnedTransactor // Generic write-only contract binding to access the raw methods on
}

// NewOwned creates a new instance of Owned, bound to a specific deployed contract.
func NewOwned(address common.Address, backend bind.ContractBackend) (*Owned, error) {
contract, err := bindOwned(address, backend, backend)
if err != nil {
return nil, err
}
return &Owned{OwnedCaller: OwnedCaller{contract: contract}, OwnedTransactor:
OwnedTransactor{contract: contract}}, nil
}

// NewOwnedCaller creates a new read-only instance of Owned, bound to a specific deployed
contract.
func NewOwnedCaller(address common.Address, caller bind.ContractCaller) (*OwnedCaller,
error) {
contract, err := bindOwned(address, caller, nil)
```

```
if err != nil {
return nil, err
}
return &OwnedCaller{contract: contract}, nil
}

// NewOwnedTransactor creates a new write-only instance of Owned, bound to a specific
deployed contract.
func NewOwnedTransactor(address common.Address, transactor bind.ContractTransactor)
(*OwnedTransactor, error) {
contract, err := bindOwned(address, nil, transactor)
if err != nil {
return nil, err
}
return &OwnedTransactor{contract: contract}, nil
}

// bindOwned binds a generic wrapper to an already deployed contract.
func bindOwned(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(OwnedABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Owned *OwnedRaw) Call(opts *bind.CallOpts, result interface{}, method string, params
...interface{}) error {
return _Owned.Contract.OwnedCaller.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Owned *OwnedRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction, error) {
return _Owned.Contract.OwnedTransactor.contract.Transfer(opts)
}
```

```
// Transact invokes the (paid) contract method with params as input values.
func (_Owned *OwnedRaw) Transact(opts *bind.TransactOpts, method string, params
...interface{}) (*types.Transaction, error) {
return _Owned.Contract.OwnedTransactor.contract.Transact(opts, method, params...)
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Owned *OwnedCallerRaw) Call(opts *bind.CallOpts, result interface{}, method string,
params ...interface{}) error {
return _Owned.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Owned *OwnedTransactorRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction,
error) {
return _Owned.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_Owned *OwnedTransactorRaw) Transact(opts *bind.TransactOpts, method string, params
...interface{}) (*types.Transaction, error) {
return _Owned.Contract.contract.Transact(opts, method, params...)
}

48:F:\git\coin\ethereum\go-ethereum\contracts\chequebook\contract\code.go

49:F:\git\coin\ethereum\go-ethereum\contracts\chequebook\gencode.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// +build none

// This program generates contract/code.go, which contains the chequebook code
// after deployment.
package main

import (
"fmt"
"io/ioutil"
```

```
"math/big"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/accounts/abi/bind/backends"
"github.com/ethereum/go-ethereum/contracts/chequebook/contract"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/crypto"
)

var (
testKey, _  =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
testAccount = core.GenesisAccount{
Address: crypto.PubkeyToAddress(testKey.PublicKey),
Balance: big.NewInt(500000000000),
}
)

func main() {
backend := backends.NewSimulatedBackend(testAccount)
auth := bind.NewKeyedTransactor(testKey)

// Deploy the contract, get the code.
addr, _, _, err := contract.DeployChequebook(auth, backend)
if err != nil {
panic(err)
}
backend.Commit()
code, err := backend.CodeAt(nil, addr, nil)
if err != nil {
panic(err)
}
if len(code) == 0 {
panic("empty code")
}

// Write the output file.
content := fmt.Sprintf(`package contract

// ContractDeployedCode is used to detect suicides. This constant needs to be
// updated when the contract code is changed.
```

```
const ContractDeployedCode = "%#x"
`, code)
if err := ioutil.WriteFile("contract/code.go", []byte(content), 0644); err != nil {
panic(err)
}
}
```

50:F:\git\coin\ethereum\go-ethereum\contracts\ens\contract\ens.go

```
// ENSABI is the input ABI used to generate the binding from.
const ENSABI =
`[{"constant":true,"inputs":[{"name":"node","type":"bytes32"}],"name":"resolver","outputs":[{"name":""
,"type":"address"}],"type":"function"},{"constant":true,"inputs":[{"name":"node","type":"bytes32"}],"n
ame":"owner","outputs":[{"name":"","type":"address"}],"type":"function"},{"constant":false,"inputs":[{"
name":"node","type":"bytes32"},{"name":"label","type":"bytes32"},{"name":"owner","type":"address"
}],"name":"setSubnodeOwner","outputs":[],"type":"function"},{"constant":false,"inputs":[{"name":"no
de","type":"bytes32"},{"name":"resolver","type":"address"}],"name":"setResolver","outputs":[],"type"
:"function"},{"constant":false,"inputs":[{"name":"node","type":"bytes32"},{"name":"owner","type":"ad
dress"}],"name":"setOwner","outputs":[],"type":"function"},{"inputs":[{"name":"owner","type":"addres
s"}],"type":"constructor"},{"anonymous":false,"inputs":[{"indexed":true,"name":"node","type":"bytes3
2"},{"indexed":true,"name":"label","type":"bytes32"},{"indexed":false,"name":"owner","type":"addres
s"}],"name":"NewOwner","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"name":"node
","type":"bytes32"},{"indexed":false,"name":"owner","type":"address"}],"name":"Transfer","type":"ev
ent"},{"anonymous":false,"inputs":[{"indexed":true,"name":"node","type":"bytes32"},{"indexed":false
,"name":"resolver","type":"address"}],"name":"NewResolver","type":"event"}]`

// ENSBin is the compiled bytecode used for deploying new contracts.
const ENSBin =
`0x606060405260405160208061032683395060806040525160008080526020527fad3228b676f7
d3cd4284a5443f17f1962b36e491b30a40b2405849e597ba5fb58054600160a060020a031916821
79055506102c88061005e6000396000f3606060405260e060020a60003504630178b8bf81146100
4757806302571be31461006e57806306ab592314610091578063189f70a146100c85780635b0fc
9c3146100fc575b005b6101306004356000818152602081905260409020600101546001160a06002
0a03165b919050565b6101306004356000818152602081905260409020546001160a060020a0316
610069565b6100456004356024356044356000838152602081905260408120548490600160a060
020a0390811633919091161461014d57610002565b6100456004356024356000828152602081 90
526040902054829060016 0a060020a03908116339190911614610 1e757610002565b6100456004
3560243560008281526020819052604090205482906001 60a060020a039081163391909116 1461
025957610002565b60408051600160a060020a039290921682525190819003602001 90f35b6040
805186815260208181018790528251918290038301822060016 0a060020a0387168352925192 94
50869288927fce0457fe73731f824cc272376169235128c118b49d344817417c6d108d155e829290
8290030190a3826000600050600084600019168152602001908152602001600020600050600001
60006101000a815481600160a060020a030219169083021790555050505050565b6040805160
```

0160a060020a03841681529051849 17f335721b01866dc23fbee8b6b2c7b1e14d6f05c28cd35a2c9
34239f94095602a091908190036020019 0a2506000828152602081905260409020600101805473f
ffffffffffffffffffffffffffffffffffffff191682179055505056 5b60408051600160a060020a0384168152905184
917fd4735d920b0f87494915f556dd9b54c8f309026070caea5c737245152564d266919081900360
200190a2506000828152602081905260409020805473ffffffffffffffffffffffffffffffffffffff19168217905550
5056`

// DeployENS deploys a new Ethereum contract, binding an instance of ENS to it.
func DeployENS(auth *bind.TransactOpts, backend bind.ContractBackend, owner
common.Address) (common.Address, *types.Transaction, *ENS, error) {
parsed, err := abi.JSON(strings.NewReader(ENSABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed, common.FromHex(ENSBin),
backend, owner)
if err != nil {
return common.Address{}, nil, nil, err
}
return address, tx, &ENS{ENSCaller: ENSCaller{contract: contract}, ENSTransactor:
ENSTransactor{contract: contract}}, nil
}

// ENS is an auto generated Go binding around an Ethereum contract.
type ENS struct {
ENSCaller   // Read-only binding to the contract
ENSTransactor // Write-only binding to the contract
}

// ENSCaller is an auto generated read-only Go binding around an Ethereum contract.
type ENSCaller struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ENSTransactor is an auto generated write-only Go binding around an Ethereum contract.
type ENSTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ENSSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type ENSSession struct {

```
Contract     *ENS              // Generic contract binding to set the session for
CallOpts     bind.CallOpts     // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// ENSCallerSession is an auto generated read-only Go binding around an Ethereum contract,
// with pre-set call options.
type ENSCallerSession struct {
Contract *ENSCaller     // Generic contract caller binding to set the session for
CallOpts bind.CallOpts // Call options to use throughout this session
}

// ENSTransactorSession is an auto generated write-only Go binding around an Ethereum
contract,
// with pre-set transact options.
type ENSTransactorSession struct {
Contract     *ENSTransactor     // Generic contract transactor binding to set the session for
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// ENSRaw is an auto generated low-level Go binding around an Ethereum contract.
type ENSRaw struct {
Contract *ENS // Generic contract binding to access the raw methods on
}

// ENSCallerRaw is an auto generated low-level read-only Go binding around an Ethereum
contract.
type ENSCallerRaw struct {
Contract *ENSCaller // Generic read-only contract binding to access the raw methods on
}

// ENSTransactorRaw is an auto generated low-level write-only Go binding around an Ethereum
contract.
type ENSTransactorRaw struct {
Contract *ENSTransactor // Generic write-only contract binding to access the raw methods on
}

// NewENS creates a new instance of ENS, bound to a specific deployed contract.
func NewENS(address common.Address, backend bind.ContractBackend) (*ENS, error) {
contract, err := bindENS(address, backend, backend)
if err != nil {
return nil, err
```

```
}
return &ENS{ENSCaller: ENSCaller{contract: contract}, ENSTransactor: ENSTransactor{contract:
contract}}, nil
}

// NewENSCaller creates a new read-only instance of ENS, bound to a specific deployed contract.
func NewENSCaller(address common.Address, caller bind.ContractCaller) (*ENSCaller, error) {
contract, err := bindENS(address, caller, nil)
if err != nil {
return nil, err
}
return &ENSCaller{contract: contract}, nil
}

// NewENSTransactor creates a new write-only instance of ENS, bound to a specific deployed
contract.
func NewENSTransactor(address common.Address, transactor bind.ContractTransactor)
(*ENSTransactor, error) {
contract, err := bindENS(address, nil, transactor)
if err != nil {
return nil, err
}
return &ENSTransactor{contract: contract}, nil
}

// bindENS binds a generic wrapper to an already deployed contract.
func bindENS(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(ENSABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_ENS *ENSRaw) Call(opts *bind.CallOpts, result interface{}, method string, params
...interface{}) error {
return _ENS.Contract.ENSCaller.contract.Call(opts, result, method, params...)
```

```
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_ENS *ENSRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction, error) {
return _ENS.Contract.ENSTransactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_ENS *ENSRaw) Transact(opts *bind.TransactOpts, method string, params ...interface{})
(*types.Transaction, error) {
return _ENS.Contract.ENSTransactor.contract.Transact(opts, method, params...)
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_ENS *ENSCallerRaw) Call(opts *bind.CallOpts, result interface{}, method string, params
...interface{}) error {
return _ENS.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_ENS *ENSTransactorRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction, error) {
return _ENS.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_ENS *ENSTransactorRaw) Transact(opts *bind.TransactOpts, method string, params
...interface{}) (*types.Transaction, error) {
return _ENS.Contract.contract.Transact(opts, method, params...)
}

// Owner is a free data retrieval call binding the contract method 0x02571be3.
//
// Solidity: function owner(node bytes32) constant returns(address)
func (_ENS *ENSCaller) Owner(opts *bind.CallOpts, node [32]byte) (common.Address, error) {
var (
ret0 = new(common.Address)
)
```

```
out := ret0
err := _ENS.contract.Call(opts, out, "owner", node)
return *ret0, err
}

// Owner is a free data retrieval call binding the contract method 0x02571be3.
//
// Solidity: function owner(node bytes32) constant returns(address)
func (_ENS *ENSSession) Owner(node [32]byte) (common.Address, error) {
return _ENS.Contract.Owner(&_ENS.CallOpts, node)
}

// Owner is a free data retrieval call binding the contract method 0x02571be3.
//
// Solidity: function owner(node bytes32) constant returns(address)
func (_ENS *ENSCallerSession) Owner(node [32]byte) (common.Address, error) {
return _ENS.Contract.Owner(&_ENS.CallOpts, node)
}

// Resolver is a free data retrieval call binding the contract method 0x0178b8bf.
//
// Solidity: function resolver(node bytes32) constant returns(address)
func (_ENS *ENSCaller) Resolver(opts *bind.CallOpts, node [32]byte) (common.Address, error) {
var (
ret0 = new(common.Address)
)
out := ret0
err := _ENS.contract.Call(opts, out, "resolver", node)
return *ret0, err
}

// Resolver is a free data retrieval call binding the contract method 0x0178b8bf.
//
// Solidity: function resolver(node bytes32) constant returns(address)
func (_ENS *ENSSession) Resolver(node [32]byte) (common.Address, error) {
return _ENS.Contract.Resolver(&_ENS.CallOpts, node)
}

// Resolver is a free data retrieval call binding the contract method 0x0178b8bf.
//
// Solidity: function resolver(node bytes32) constant returns(address)
func (_ENS *ENSCallerSession) Resolver(node [32]byte) (common.Address, error) {
```

```
return _ENS.Contract.Resolver(&_ENS.CallOpts, node)
}

// SetOwner is a paid mutator transaction binding the contract method 0x5b0fc9c3.
//
// Solidity: function setOwner(node bytes32, owner address) returns()
func (_ENS *ENSTransactor) SetOwner(opts *bind.TransactOpts, node [32]byte, owner
common.Address) (*types.Transaction, error) {
return _ENS.contract.Transact(opts, "setOwner", node, owner)
}

// SetOwner is a paid mutator transaction binding the contract method 0x5b0fc9c3.
//
// Solidity: function setOwner(node bytes32, owner address) returns()
func (_ENS *ENSSession) SetOwner(node [32]byte, owner common.Address)
(*types.Transaction, error) {
return _ENS.Contract.SetOwner(&_ENS.TransactOpts, node, owner)
}

// SetOwner is a paid mutator transaction binding the contract method 0x5b0fc9c3.
//
// Solidity: function setOwner(node bytes32, owner address) returns()
func (_ENS *ENSTransactorSession) SetOwner(node [32]byte, owner common.Address)
(*types.Transaction, error) {
return _ENS.Contract.SetOwner(&_ENS.TransactOpts, node, owner)
}

// SetResolver is a paid mutator transaction binding the contract method 0x1896f70a.
//
// Solidity: function setResolver(node bytes32, resolver address) returns()
func (_ENS *ENSTransactor) SetResolver(opts *bind.TransactOpts, node [32]byte, resolver
common.Address) (*types.Transaction, error) {
return _ENS.contract.Transact(opts, "setResolver", node, resolver)
}

// SetResolver is a paid mutator transaction binding the contract method 0x1896f70a.
//
// Solidity: function setResolver(node bytes32, resolver address) returns()
func (_ENS *ENSSession) SetResolver(node [32]byte, resolver common.Address)
(*types.Transaction, error) {
return _ENS.Contract.SetResolver(&_ENS.TransactOpts, node, resolver)
}
```

```
// SetResolver is a paid mutator transaction binding the contract method 0x1896f70a.
//
// Solidity: function setResolver(node bytes32, resolver address) returns()
func (_ENS *ENSTransactorSession) SetResolver(node [32]byte, resolver common.Address)
(*types.Transaction, error) {
return _ENS.Contract.SetResolver(&_ENS.TransactOpts, node, resolver)
}

// SetSubnodeOwner is a paid mutator transaction binding the contract method 0x06ab5923.
//
// Solidity: function setSubnodeOwner(node bytes32, label bytes32, owner address) returns()
func (_ENS *ENSTransactor) SetSubnodeOwner(opts *bind.TransactOpts, node [32]byte, label
[32]byte, owner common.Address) (*types.Transaction, error) {
return _ENS.contract.Transact(opts, "setSubnodeOwner", node, label, owner)
}

// SetSubnodeOwner is a paid mutator transaction binding the contract method 0x06ab5923.
//
// Solidity: function setSubnodeOwner(node bytes32, label bytes32, owner address) returns()
func (_ENS *ENSSession) SetSubnodeOwner(node [32]byte, label [32]byte, owner
common.Address) (*types.Transaction, error) {
return _ENS.Contract.SetSubnodeOwner(&_ENS.TransactOpts, node, label, owner)
}

// SetSubnodeOwner is a paid mutator transaction binding the contract method 0x06ab5923.
//
// Solidity: function setSubnodeOwner(node bytes32, label bytes32, owner address) returns()
func (_ENS *ENSTransactorSession) SetSubnodeOwner(node [32]byte, label [32]byte, owner
common.Address) (*types.Transaction, error) {
return _ENS.Contract.SetSubnodeOwner(&_ENS.TransactOpts, node, label, owner)
}

// FIFSRegistrarABI is the input ABI used to generate the binding from.
const FIFSRegistrarABI =
`[{"constant":false,"inputs":[{"name":"subnode","type":"bytes32"},{"name":"owner","type":"address"}
],"name":"register","outputs":[],"type":"function"},{"inputs":[{"name":"ensAddr","type":"address"},{"na
me":"node","type":"bytes32"}],"type":"constructor"}]`

// FIFSRegistrarBin is the compiled bytecode used for deploying new contracts.
const FIFSRegistrarBin =
`0x60606040818152806106208339608a090525160805160008054600160a060020a03191683179
```

0558160a06103678061008783390180826001 60a060020a03168152602001915050604051809103906000f060016000610100 0a815481600160a060020a030219169083021790555080600260005081905550505061023280610 3ee6000396000f360606040526040516020806103678339506080604052516000805460016 0a060020a03191682179055506103308061003760003960 00f36060604052361561004b5760e060020a60003504632dff694181146100535780633b3b57de1461007557806341b9dc2b146100a0578063c3d014d614610139578063d5fa2b00146101b2575b61022b610002565b61022d60043560008181526002602052604081205490811415610270576100025 65b61023f60043560008181526001602052604081205460016 0a060020a03169081141561027057610002565b61025c60043560243560007f6164647200000000000000000000000000000000000000000000000000000000000000000000000000821480156100f057506000838152600160205260408120546001 60a060020a031614155b8061013257507f636f6e74656e7400000000000000000000000000000000000000000000000000000000000000000000821480156101325750600083815260026020526040812054141 55b9392505050565b61022b60043560243560008054604080516 0e060020a6302571be3028152600481018690529051859360 0160a060020a033381169416926302571be392602482810193602093839003909101908290876161da5a03f1156100025750506040515160016 0a060020a03169190911490506102755761000256 5b61022b60043560243560008054604080516 0e060020a6302571be3028152600481018690529051859360 0160a060020a033381169416926302571be392602482810193602093839003909101908290876161da5a03f1156100025750506040515160016 0a060020a03169190911490506102c157610002565b005b60408051918252519081900360200190f35b60408051600160a060020a039290921682525190819003602001 90f35b60408051911515825251908190036020019 0f35b919050565b60008381526002602090815260409182902084905581518481529151859 27f0424b6fe0d9c3bdbece0e7879dc241bb0c22e900be8b6c168b4ee08bd9bf83bc92908290030190a2505050565b6000838152600160209081526040918290208054 73ffffffffffffffffffffffffffffffffffffffff191685179055815160016 0a060020a03851681529151859 27f52d7d861f09ab3d26239d492e8968629f95e9e318cf0b73bfddc441522a15fd292908290030190a2505050566060604052 60e060020a6000350463d22057a9811461001b575b005b6100196004356024356002546040805191825260208281018590526000805483519485900384018520 7f02571be3000000000000000000000000000000000000000000000000000000000000000000000865260048601819052935193949193600160a060020a0390911692 6302571be3924024818101939182900301818787 6161da5a03f1156100025750506040515191505060 0160a060020a03811660001480159061 00d4575033600160a060020a031681600160a060020a031614155b156100de57610002565b60408051600080546002547f06ab59230000000000000000000000000000000000000000000000000000000000000000000000000000845260048401526024830188905230600160a060020a039081166044850152935193169263 06ab59239260648181019392918290030181838 76161da5a03f1156100025750506000805460015 4604080517f1896f70a0000000000000000000000000000000000000000000000000000000000000000000000000000815260048101889052600160a060020a039283166024820152905192909116935 0631896f70a92604482810193919282900301818 3876161da5a03f1156100025750506000805460 4080517f5b0fc9c3000000000000000000000000000000000000000000000000000000000000000000000000000081526004810187905260 0160a060020a038881166024830152915192909 1169350635b0fc9c392604482810193919282900301818 3876161da5a03f1156100025750505050505056`

// DeployFIFSRegistrar deploys a new Ethereum contract, binding an instance of FIFSRegistrar to it.

```
func DeployFIFSRegistrar(auth *bind.TransactOpts, backend bind.ContractBackend, ensAddr
common.Address, node [32]byte) (common.Address, *types.Transaction, *FIFSRegistrar, error) {
parsed, err := abi.JSON(strings.NewReader(FIFSRegistrarABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed,
common.FromHex(FIFSRegistrarBin), backend, ensAddr, node)
if err != nil {
return common.Address{}, nil, nil, err
}
return address, tx, &FIFSRegistrar{FIFSRegistrarCaller: FIFSRegistrarCaller{contract: contract},
FIFSRegistrarTransactor: FIFSRegistrarTransactor{contract: contract}}, nil
}

// FIFSRegistrar is an auto generated Go binding around an Ethereum contract.
type FIFSRegistrar struct {
FIFSRegistrarCaller     // Read-only binding to the contract
FIFSRegistrarTransactor // Write-only binding to the contract
}

// FIFSRegistrarCaller is an auto generated read-only Go binding around an Ethereum contract.
type FIFSRegistrarCaller struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// FIFSRegistrarTransactor is an auto generated write-only Go binding around an Ethereum
contract.
type FIFSRegistrarTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// FIFSRegistrarSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type FIFSRegistrarSession struct {
Contract     *FIFSRegistrar    // Generic contract binding to set the session for
CallOpts     bind.CallOpts     // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// FIFSRegistrarCallerSession is an auto generated read-only Go binding around an Ethereum
contract,
```

```
// with pre-set call options.
type FIFSRegistrarCallerSession struct {
Contract *FIFSRegistrarCaller // Generic contract caller binding to set the session for
CallOpts bind.CallOpts        // Call options to use throughout this session
}

// FIFSRegistrarTransactorSession is an auto generated write-only Go binding around an
Ethereum contract,
// with pre-set transact options.
type FIFSRegistrarTransactorSession struct {
Contract     *FIFSRegistrarTransactor // Generic contract transactor binding to set the session for
TransactOpts bind.TransactOpts        // Transaction auth options to use throughout this session
}

// FIFSRegistrarRaw is an auto generated low-level Go binding around an Ethereum contract.
type FIFSRegistrarRaw struct {
Contract *FIFSRegistrar // Generic contract binding to access the raw methods on
}

// FIFSRegistrarCallerRaw is an auto generated low-level read-only Go binding around an
Ethereum contract.
type FIFSRegistrarCallerRaw struct {
Contract *FIFSRegistrarCaller // Generic read-only contract binding to access the raw methods on
}

// FIFSRegistrarTransactorRaw is an auto generated low-level write-only Go binding around an
Ethereum contract.
type FIFSRegistrarTransactorRaw struct {
Contract *FIFSRegistrarTransactor // Generic write-only contract binding to access the raw
methods on
}

// NewFIFSRegistrar creates a new instance of FIFSRegistrar, bound to a specific deployed
contract.
func NewFIFSRegistrar(address common.Address, backend bind.ContractBackend)
(*FIFSRegistrar, error) {
contract, err := bindFIFSRegistrar(address, backend, backend)
if err != nil {
return nil, err
}
return &FIFSRegistrar{FIFSRegistrarCaller: FIFSRegistrarCaller{contract: contract},
FIFSRegistrarTransactor: FIFSRegistrarTransactor{contract: contract}}, nil
```

```
}

// NewFIFSRegistrarCaller creates a new read-only instance of FIFSRegistrar, bound to a specific deployed contract.
func NewFIFSRegistrarCaller(address common.Address, caller bind.ContractCaller) (*FIFSRegistrarCaller, error) {
contract, err := bindFIFSRegistrar(address, caller, nil)
if err != nil {
return nil, err
}
return &FIFSRegistrarCaller{contract: contract}, nil
}

// NewFIFSRegistrarTransactor creates a new write-only instance of FIFSRegistrar, bound to a specific deployed contract.
func NewFIFSRegistrarTransactor(address common.Address, transactor bind.ContractTransactor) (*FIFSRegistrarTransactor, error) {
contract, err := bindFIFSRegistrar(address, nil, transactor)
if err != nil {
return nil, err
}
return &FIFSRegistrarTransactor{contract: contract}, nil
}

// bindFIFSRegistrar binds a generic wrapper to an already deployed contract.
func bindFIFSRegistrar(address common.Address, caller bind.ContractCaller, transactor bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(FIFSRegistrarABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_FIFSRegistrar *FIFSRegistrarRaw) Call(opts *bind.CallOpts, result interface{}, method string, params ...interface{}) error {
return _FIFSRegistrar.Contract.FIFSRegistrarCaller.contract.Call(opts, result, method, params...)
}
```

```
// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_FIFSRegistrar *FIFSRegistrarRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction,
error) {
return _FIFSRegistrar.Contract.FIFSRegistrarTransactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_FIFSRegistrar *FIFSRegistrarRaw) Transact(opts *bind.TransactOpts, method string,
params ...interface{}) (*types.Transaction, error) {
return _FIFSRegistrar.Contract.FIFSRegistrarTransactor.contract.Transact(opts, method,
params...)
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_FIFSRegistrar *FIFSRegistrarCallerRaw) Call(opts *bind.CallOpts, result interface{},
method string, params ...interface{}) error {
return _FIFSRegistrar.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_FIFSRegistrar *FIFSRegistrarTransactorRaw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _FIFSRegistrar.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_FIFSRegistrar *FIFSRegistrarTransactorRaw) Transact(opts *bind.TransactOpts, method
string, params ...interface{}) (*types.Transaction, error) {
return _FIFSRegistrar.Contract.contract.Transact(opts, method, params...)
}

// Register is a paid mutator transaction binding the contract method 0xd22057a9.
//
// Solidity: function register(subnode bytes32, owner address) returns()
func (_FIFSRegistrar *FIFSRegistrarTransactor) Register(opts *bind.TransactOpts, subnode
[32]byte, owner common.Address) (*types.Transaction, error) {
```

```
return _FIFSRegistrar.contract.Transact(opts, "register", subnode, owner)
}

// Register is a paid mutator transaction binding the contract method 0xd22057a9.
//
// Solidity: function register(subnode bytes32, owner address) returns()
func (_FIFSRegistrar *FIFSRegistrarSession) Register(subnode [32]byte, owner
common.Address) (*types.Transaction, error) {
return _FIFSRegistrar.Contract.Register(&_FIFSRegistrar.TransactOpts, subnode, owner)
}

// Register is a paid mutator transaction binding the contract method 0xd22057a9.
//
// Solidity: function register(subnode bytes32, owner address) returns()
func (_FIFSRegistrar *FIFSRegistrarTransactorSession) Register(subnode [32]byte, owner
common.Address) (*types.Transaction, error) {
return _FIFSRegistrar.Contract.Register(&_FIFSRegistrar.TransactOpts, subnode, owner)
}

// PublicResolverABI is the input ABI used to generate the binding from.
const PublicResolverABI =
`[{"constant":true,"inputs":[{"name":"node","type":"bytes32"}],"name":"content","outputs":[{"name":"r
et","type":"bytes32"}],"type":"function"},{"constant":true,"inputs":[{"name":"node","type":"bytes32"}],"
name":"addr","outputs":[{"name":"ret","type":"address"}],"type":"function"},{"constant":false,"inputs":
[{"name":"node","type":"bytes32"},{"name":"kind","type":"bytes32"}],"name":"has","outputs":[{"name
":"","type":"bool"}],"type":"function"},{"constant":false,"inputs":[{"name":"node","type":"bytes32"},{"n
ame":"hash","type":"bytes32"}],"name":"setContent","outputs":[],"type":"function"},{"constant":false,
"inputs":[{"name":"node","type":"bytes32"},{"name":"addr","type":"address"}],"name":"setAddr","out
puts":[],"type":"function"},{"inputs":[{"name":"ensAddr","type":"address"}],"type":"constructor"},{"ano
nymous":false,"inputs":[{"indexed":true,"name":"node","type":"bytes32"},{"indexed":false,"name":"a
","type":"address"}],"name":"AddrChanged","type":"event"},{"anonymous":false,"inputs":[{"indexed"
:true,"name":"node","type":"bytes32"},{"indexed":false,"name":"hash","type":"bytes32"}],"name":"C
ontentChanged","type":"event"}]`

// PublicResolverBin is the compiled bytecode used for deploying new contracts.
const PublicResolverBin =
`0x60606040526040516020806103678339506080604052516000805460016a060020a0319168
21790555061033080610037600039600f3606060405236156100 4b5760e060020a60003504632
dff6941811461005357806 3b3b57de1461007557806341b9dc2b146100a0578063c3d014d61461
0139578063d5fa2b00146101b2575b61022b610002565b61022d6004356000818152600260205 2
60408120549081141561027057610002565b61023f6004356000818152600160205260408120 54
600160a060020a0316908114156102705761 0002565b61025c6004356024356007f6164647200
```

00000000000000000000000000000000000000000000000000821480156100f05750600083
81526001602052604081205460016 0a060020a031614155b8061013257507f636f6e74656e74000
00000000000000000000000000000000000000000000000821480156101325750600083815260 0
26020526040812054 14155b9392505050565b61022b60043560243560008054604080516 0e0600
20a6302571be3028152600481018690529051859360016 0a060020a033381169416926302571be
39260248281019360209383900390910190829087616 1da5a03f11561000257505060405151600
160a060020a031691909114905061027557610002565b61022b60043560243560008054604080 5
160e060020a6302571be3028152600481018690529051859360016 0a060020a033381169416926
302571be39260248281019360209383900390910190829087616 1da5a03f115610002575050604
05151600160a060020a031691909114905061 02c157610002565b005b60408051918252519081 9
00360200190f35b60408051600160a060020a039290921682525190819003602001 90f35b604080
519115158252519081900360200190f35b919050565b60008381526002602090815260409182 90
208490558151848152915185927f0424b6fe0d9c3bdbece0e7879dc241bb0c22e900be8b6c168b4
ee08bd9bf83bc92908290030190a2505050565b60008381526001602090815260409182902080 5
473ffffffffffffffffffffffffffffffffffffffff191685179055815160016 0a060020a038516815291518 5927f52d7d
861f09ab3d26239d492e8968629f95e9e318cf0b73bfddc441522a15fd292908290030190a2505050
56`

// DeployPublicResolver deploys a new Ethereum contract, binding an instance of PublicResolver to it.
func DeployPublicResolver(auth *bind.TransactOpts, backend bind.ContractBackend, ensAddr common.Address) (common.Address, *types.Transaction, *PublicResolver, error) {
parsed, err := abi.JSON(strings.NewReader(PublicResolverABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed,
common.FromHex(PublicResolverBin), backend, ensAddr)
if err != nil {
return common.Address{}, nil, nil, err
}
return address, tx, &PublicResolver{PublicResolverCaller: PublicResolverCaller{contract: contract}, PublicResolverTransactor: PublicResolverTransactor{contract: contract}}, nil
}

// PublicResolver is an auto generated Go binding around an Ethereum contract.
type PublicResolver struct {
PublicResolverCaller   // Read-only binding to the contract
PublicResolverTransactor // Write-only binding to the contract
}

// PublicResolverCaller is an auto generated read-only Go binding around an Ethereum contract.

```
type PublicResolverCaller struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// PublicResolverTransactor is an auto generated write-only Go binding around an Ethereum
contract.
type PublicResolverTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// PublicResolverSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type PublicResolverSession struct {
Contract     *PublicResolver   // Generic contract binding to set the session for
CallOpts     bind.CallOpts     // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// PublicResolverCallerSession is an auto generated read-only Go binding around an Ethereum
contract,
// with pre-set call options.
type PublicResolverCallerSession struct {
Contract *PublicResolverCaller // Generic contract caller binding to set the session for
CallOpts bind.CallOpts         // Call options to use throughout this session
}

// PublicResolverTransactorSession is an auto generated write-only Go binding around an
Ethereum contract,
// with pre-set transact options.
type PublicResolverTransactorSession struct {
Contract     *PublicResolverTransactor // Generic contract transactor binding to set the session for
TransactOpts bind.TransactOpts         // Transaction auth options to use throughout this session
}

// PublicResolverRaw is an auto generated low-level Go binding around an Ethereum contract.
type PublicResolverRaw struct {
Contract *PublicResolver // Generic contract binding to access the raw methods on
}

// PublicResolverCallerRaw is an auto generated low-level read-only Go binding around an
Ethereum contract.
type PublicResolverCallerRaw struct {
```

```
Contract *PublicResolverCaller // Generic read-only contract binding to access the raw methods on
}

// PublicResolverTransactorRaw is an auto generated low-level write-only Go binding around an Ethereum contract.
type PublicResolverTransactorRaw struct {
Contract *PublicResolverTransactor // Generic write-only contract binding to access the raw methods on
}

// NewPublicResolver creates a new instance of PublicResolver, bound to a specific deployed contract.
func NewPublicResolver(address common.Address, backend bind.ContractBackend) (*PublicResolver, error) {
contract, err := bindPublicResolver(address, backend, backend)
if err != nil {
return nil, err
}
return &PublicResolver{PublicResolverCaller: PublicResolverCaller{contract: contract}, PublicResolverTransactor: PublicResolverTransactor{contract: contract}}, nil
}

// NewPublicResolverCaller creates a new read-only instance of PublicResolver, bound to a specific deployed contract.
func NewPublicResolverCaller(address common.Address, caller bind.ContractCaller) (*PublicResolverCaller, error) {
contract, err := bindPublicResolver(address, caller, nil)
if err != nil {
return nil, err
}
return &PublicResolverCaller{contract: contract}, nil
}

// NewPublicResolverTransactor creates a new write-only instance of PublicResolver, bound to a specific deployed contract.
func NewPublicResolverTransactor(address common.Address, transactor bind.ContractTransactor) (*PublicResolverTransactor, error) {
contract, err := bindPublicResolver(address, nil, transactor)
if err != nil {
return nil, err
}
```

```
return &PublicResolverTransactor{contract: contract}, nil
}

// bindPublicResolver binds a generic wrapper to an already deployed contract.
func bindPublicResolver(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(PublicResolverABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_PublicResolver *PublicResolverRaw) Call(opts *bind.CallOpts, result interface{}, method
string, params ...interface{}) error {
return _PublicResolver.Contract.PublicResolverCaller.contract.Call(opts, result, method,
params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_PublicResolver *PublicResolverRaw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _PublicResolver.Contract.PublicResolverTransactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_PublicResolver *PublicResolverRaw) Transact(opts *bind.TransactOpts, method string,
params ...interface{}) (*types.Transaction, error) {
return _PublicResolver.Contract.PublicResolverTransactor.contract.Transact(opts, method,
params...)
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_PublicResolver *PublicResolverCallerRaw) Call(opts *bind.CallOpts, result interface{},
```

```
method string, params ...interface{}) error {
return _PublicResolver.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_PublicResolver *PublicResolverTransactorRaw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _PublicResolver.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_PublicResolver *PublicResolverTransactorRaw) Transact(opts *bind.TransactOpts, method
string, params ...interface{}) (*types.Transaction, error) {
return _PublicResolver.Contract.contract.Transact(opts, method, params...)
}

// Addr is a free data retrieval call binding the contract method 0x3b3b57de.
//
// Solidity: function addr(node bytes32) constant returns(ret address)
func (_PublicResolver *PublicResolverCaller) Addr(opts *bind.CallOpts, node [32]byte)
(common.Address, error) {
var (
ret0 = new(common.Address)
)
out := ret0
err := _PublicResolver.contract.Call(opts, out, "addr", node)
return *ret0, err
}

// Addr is a free data retrieval call binding the contract method 0x3b3b57de.
//
// Solidity: function addr(node bytes32) constant returns(ret address)
func (_PublicResolver *PublicResolverSession) Addr(node [32]byte) (common.Address, error) {
return _PublicResolver.Contract.Addr(&_PublicResolver.CallOpts, node)
}

// Addr is a free data retrieval call binding the contract method 0x3b3b57de.
//
// Solidity: function addr(node bytes32) constant returns(ret address)
func (_PublicResolver *PublicResolverCallerSession) Addr(node [32]byte) (common.Address,
error) {
```

```
return _PublicResolver.Contract.Addr(&_PublicResolver.CallOpts, node)
}

// Content is a free data retrieval call binding the contract method 0x2dff6941.
//
// Solidity: function content(node bytes32) constant returns(ret bytes32)
func (_PublicResolver *PublicResolverCaller) Content(opts *bind.CallOpts, node [32]byte)
([32]byte, error) {
var (
ret0 = new([32]byte)
)
out := ret0
err := _PublicResolver.contract.Call(opts, out, "content", node)
return *ret0, err
}

// Content is a free data retrieval call binding the contract method 0x2dff6941.
//
// Solidity: function content(node bytes32) constant returns(ret bytes32)
func (_PublicResolver *PublicResolverSession) Content(node [32]byte) ([32]byte, error) {
return _PublicResolver.Contract.Content(&_PublicResolver.CallOpts, node)
}

// Content is a free data retrieval call binding the contract method 0x2dff6941.
//
// Solidity: function content(node bytes32) constant returns(ret bytes32)
func (_PublicResolver *PublicResolverCallerSession) Content(node [32]byte) ([32]byte, error) {
return _PublicResolver.Contract.Content(&_PublicResolver.CallOpts, node)
}

// Has is a paid mutator transaction binding the contract method 0x41b9dc2b.
//
// Solidity: function has(node bytes32, kind bytes32) returns(bool)
func (_PublicResolver *PublicResolverTransactor) Has(opts *bind.TransactOpts, node [32]byte,
kind [32]byte) (*types.Transaction, error) {
return _PublicResolver.contract.Transact(opts, "has", node, kind)
}

// Has is a paid mutator transaction binding the contract method 0x41b9dc2b.
//
// Solidity: function has(node bytes32, kind bytes32) returns(bool)
func (_PublicResolver *PublicResolverSession) Has(node [32]byte, kind [32]byte)
```

```
(*types.Transaction, error) {
return _PublicResolver.Contract.Has(&_PublicResolver.TransactOpts, node, kind)
}

// Has is a paid mutator transaction binding the contract method 0x41b9dc2b.
//
// Solidity: function has(node bytes32, kind bytes32) returns(bool)
func (_PublicResolver *PublicResolverTransactorSession) Has(node [32]byte, kind [32]byte)
(*types.Transaction, error) {
return _PublicResolver.Contract.Has(&_PublicResolver.TransactOpts, node, kind)
}

// SetAddr is a paid mutator transaction binding the contract method 0xd5fa2b00.
//
// Solidity: function setAddr(node bytes32, addr address) returns()
func (_PublicResolver *PublicResolverTransactor) SetAddr(opts *bind.TransactOpts, node
[32]byte, addr common.Address) (*types.Transaction, error) {
return _PublicResolver.contract.Transact(opts, "setAddr", node, addr)
}

// SetAddr is a paid mutator transaction binding the contract method 0xd5fa2b00.
//
// Solidity: function setAddr(node bytes32, addr address) returns()
func (_PublicResolver *PublicResolverSession) SetAddr(node [32]byte, addr common.Address)
(*types.Transaction, error) {
return _PublicResolver.Contract.SetAddr(&_PublicResolver.TransactOpts, node, addr)
}

// SetAddr is a paid mutator transaction binding the contract method 0xd5fa2b00.
//
// Solidity: function setAddr(node bytes32, addr address) returns()
func (_PublicResolver *PublicResolverTransactorSession) SetAddr(node [32]byte, addr
common.Address) (*types.Transaction, error) {
return _PublicResolver.Contract.SetAddr(&_PublicResolver.TransactOpts, node, addr)
}

// SetContent is a paid mutator transaction binding the contract method 0xc3d014d6.
//
// Solidity: function setContent(node bytes32, hash bytes32) returns()
func (_PublicResolver *PublicResolverTransactor) SetContent(opts *bind.TransactOpts, node
[32]byte, hash [32]byte) (*types.Transaction, error) {
return _PublicResolver.contract.Transact(opts, "setContent", node, hash)
```

```
}

// SetContent is a paid mutator transaction binding the contract method 0xc3d014d6.
//
// Solidity: function setContent(node bytes32, hash bytes32) returns()
func (_PublicResolver *PublicResolverSession) SetContent(node [32]byte, hash [32]byte)
(*types.Transaction, error) {
return _PublicResolver.Contract.SetContent(&_PublicResolver.TransactOpts, node, hash)
}

// SetContent is a paid mutator transaction binding the contract method 0xc3d014d6.
//
// Solidity: function setContent(node bytes32, hash bytes32) returns()
func (_PublicResolver *PublicResolverTransactorSession) SetContent(node [32]byte, hash
[32]byte) (*types.Transaction, error) {
return _PublicResolver.Contract.SetContent(&_PublicResolver.TransactOpts, node, hash)
}

// ResolverABI is the input ABI used to generate the binding from.
const ResolverABI =
`[{"constant":true,"inputs":[{"name":"node","type":"bytes32"}],"name":"content","outputs":[{"name":"r
et","type":"bytes32"}],"type":"function"},{"constant":true,"inputs":[{"name":"node","type":"bytes32"}],"
name":"addr","outputs":[{"name":"ret","type":"address"}],"type":"function"},{"constant":false,"inputs":
[{"name":"node","type":"bytes32"},{"name":"kind","type":"bytes32"}],"name":"has","outputs":[{"name
":"","type":"bool"}],"type":"function"},{"anonymous":false,"inputs":[{"indexed":true,"name":"node","ty
pe":"bytes32"},{"indexed":false,"name":"a","type":"address"}],"name":"AddrChanged","type":"event"
},{"anonymous":false,"inputs":[{"indexed":true,"name":"node","type":"bytes32"},{"indexed":false,"na
me":"hash","type":"bytes32"}],"name":"ContentChanged","type":"event"}]`

// ResolverBin is the compiled bytecode used for deploying new contracts.
const ResolverBin = `0x`

// DeployResolver deploys a new Ethereum contract, binding an instance of Resolver to it.
func DeployResolver(auth *bind.TransactOpts, backend bind.ContractBackend)
(common.Address, *types.Transaction, *Resolver, error) {
parsed, err := abi.JSON(strings.NewReader(ResolverABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed, common.FromHex(ResolverBin),
backend)
if err != nil {
```

```
return common.Address{}, nil, nil, err
}
return address, tx, &Resolver{ResolverCaller: ResolverCaller{contract: contract},
ResolverTransactor: ResolverTransactor{contract: contract}}, nil
}

// Resolver is an auto generated Go binding around an Ethereum contract.
type Resolver struct {
ResolverCaller     // Read-only binding to the contract
ResolverTransactor // Write-only binding to the contract
}

// ResolverCaller is an auto generated read-only Go binding around an Ethereum contract.
type ResolverCaller struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ResolverTransactor is an auto generated write-only Go binding around an Ethereum contract.
type ResolverTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ResolverSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type ResolverSession struct {
Contract     *Resolver         // Generic contract binding to set the session for
CallOpts     bind.CallOpts     // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// ResolverCallerSession is an auto generated read-only Go binding around an Ethereum contract,
// with pre-set call options.
type ResolverCallerSession struct {
Contract *ResolverCaller // Generic contract caller binding to set the session for
CallOpts bind.CallOpts   // Call options to use throughout this session
}

// ResolverTransactorSession is an auto generated write-only Go binding around an Ethereum
contract,
// with pre-set transact options.
type ResolverTransactorSession struct {
Contract     *ResolverTransactor // Generic contract transactor binding to set the session for
```

```
TransactOpts bind.TransactOpts   // Transaction auth options to use throughout this session
}

// ResolverRaw is an auto generated low-level Go binding around an Ethereum contract.
type ResolverRaw struct {
Contract *Resolver // Generic contract binding to access the raw methods on
}

// ResolverCallerRaw is an auto generated low-level read-only Go binding around an Ethereum
contract.
type ResolverCallerRaw struct {
Contract *ResolverCaller // Generic read-only contract binding to access the raw methods on
}

// ResolverTransactorRaw is an auto generated low-level write-only Go binding around an
Ethereum contract.
type ResolverTransactorRaw struct {
Contract *ResolverTransactor // Generic write-only contract binding to access the raw methods on
}

// NewResolver creates a new instance of Resolver, bound to a specific deployed contract.
func NewResolver(address common.Address, backend bind.ContractBackend) (*Resolver, error) {
contract, err := bindResolver(address, backend, backend)
if err != nil {
return nil, err
}
return &Resolver{ResolverCaller: ResolverCaller{contract: contract}, ResolverTransactor:
ResolverTransactor{contract: contract}}, nil
}

// NewResolverCaller creates a new read-only instance of Resolver, bound to a specific deployed
contract.
func NewResolverCaller(address common.Address, caller bind.ContractCaller) (*ResolverCaller,
error) {
contract, err := bindResolver(address, caller, nil)
if err != nil {
return nil, err
}
return &ResolverCaller{contract: contract}, nil
}

// NewResolverTransactor creates a new write-only instance of Resolver, bound to a specific
```

```
deployed contract.
func NewResolverTransactor(address common.Address, transactor bind.ContractTransactor)
(*ResolverTransactor, error) {
contract, err := bindResolver(address, nil, transactor)
if err != nil {
return nil, err
}
return &ResolverTransactor{contract: contract}, nil
}

// bindResolver binds a generic wrapper to an already deployed contract.
func bindResolver(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(ResolverABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Resolver *ResolverRaw) Call(opts *bind.CallOpts, result interface{}, method string, params
...interface{}) error {
return _Resolver.Contract.ResolverCaller.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Resolver *ResolverRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction, error) {
return _Resolver.Contract.ResolverTransactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_Resolver *ResolverRaw) Transact(opts *bind.TransactOpts, method string, params
...interface{}) (*types.Transaction, error) {
return _Resolver.Contract.ResolverTransactor.contract.Transact(opts, method, params...)
}

// Call invokes the (constant) contract method with params as input values and
```

```
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_Resolver *ResolverCallerRaw) Call(opts *bind.CallOpts, result interface{}, method string,
params ...interface{}) error {
return _Resolver.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_Resolver *ResolverTransactorRaw) Transfer(opts *bind.TransactOpts) (*types.Transaction,
error) {
return _Resolver.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_Resolver *ResolverTransactorRaw) Transact(opts *bind.TransactOpts, method string,
params ...interface{}) (*types.Transaction, error) {
return _Resolver.Contract.contract.Transact(opts, method, params...)
}

// Addr is a free data retrieval call binding the contract method 0x3b3b57de.
//
// Solidity: function addr(node bytes32) constant returns(ret address)
func (_Resolver *ResolverCaller) Addr(opts *bind.CallOpts, node [32]byte) (common.Address,
error) {
var (
ret0 = new(common.Address)
)
out := ret0
err := _Resolver.contract.Call(opts, out, "addr", node)
return *ret0, err
}

// Addr is a free data retrieval call binding the contract method 0x3b3b57de.
//
// Solidity: function addr(node bytes32) constant returns(ret address)
func (_Resolver *ResolverSession) Addr(node [32]byte) (common.Address, error) {
return _Resolver.Contract.Addr(&_Resolver.CallOpts, node)
}

// Addr is a free data retrieval call binding the contract method 0x3b3b57de.
```

```
//
// Solidity: function addr(node bytes32) constant returns(ret address)
func (_Resolver *ResolverCallerSession) Addr(node [32]byte) (common.Address, error) {
return _Resolver.Contract.Addr(&_Resolver.CallOpts, node)
}

// Content is a free data retrieval call binding the contract method 0x2dff6941.
//
// Solidity: function content(node bytes32) constant returns(ret bytes32)
func (_Resolver *ResolverCaller) Content(opts *bind.CallOpts, node [32]byte) ([32]byte, error) {
var (
ret0 = new([32]byte)
)
out := ret0
err := _Resolver.contract.Call(opts, out, "content", node)
return *ret0, err
}

// Content is a free data retrieval call binding the contract method 0x2dff6941.
//
// Solidity: function content(node bytes32) constant returns(ret bytes32)
func (_Resolver *ResolverSession) Content(node [32]byte) ([32]byte, error) {
return _Resolver.Contract.Content(&_Resolver.CallOpts, node)
}

// Content is a free data retrieval call binding the contract method 0x2dff6941.
//
// Solidity: function content(node bytes32) constant returns(ret bytes32)
func (_Resolver *ResolverCallerSession) Content(node [32]byte) ([32]byte, error) {
return _Resolver.Contract.Content(&_Resolver.CallOpts, node)
}

// Has is a paid mutator transaction binding the contract method 0x41b9dc2b.
//
// Solidity: function has(node bytes32, kind bytes32) returns(bool)
func (_Resolver *ResolverTransactor) Has(opts *bind.TransactOpts, node [32]byte, kind [32]byte)
(*types.Transaction, error) {
return _Resolver.contract.Transact(opts, "has", node, kind)
}

// Has is a paid mutator transaction binding the contract method 0x41b9dc2b.
//
```

```
// Solidity: function has(node bytes32, kind bytes32) returns(bool)
func (_Resolver *ResolverSession) Has(node [32]byte, kind [32]byte) (*types.Transaction, error) {
return _Resolver.Contract.Has(&_Resolver.TransactOpts, node, kind)
}

// Has is a paid mutator transaction binding the contract method 0x41b9dc2b.
//
// Solidity: function has(node bytes32, kind bytes32) returns(bool)
func (_Resolver *ResolverTransactorSession) Has(node [32]byte, kind [32]byte)
(*types.Transaction, error) {
return _Resolver.Contract.Has(&_Resolver.TransactOpts, node, kind)
}

51:F:\git\coin\ethereum\go-ethereum\contracts\ens\ens.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ens

//go:generate abigen --sol contract/ens.sol --pkg contract --out contract/ens.go

import (
"math/big"
"strings"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/contracts/ens/contract"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
)

var (
MainNetAddress =
common.HexToAddress("0x314159265dD8dbb310642f98f50C066173C1259b")
TestNetAddress = common.HexToAddress("0x112234455c3a32fd11230c42e7bccd4a84e02010")
)

// swarm domain name registry and resolver
type ENS struct {
*contract.ENSSession
contractBackend bind.ContractBackend
}
```

```
// NewENS creates a struct exposing convenient high-level operations for interacting with
// the Ethereum Name Service.
func NewENS(transactOpts *bind.TransactOpts, contractAddr common.Address, contractBackend
bind.ContractBackend) (*ENS, error) {
ens, err := contract.NewENS(contractAddr, contractBackend)
if err != nil {
return nil, err
}

return &ENS{
&contract.ENSSession{
Contract:     ens,
TransactOpts: *transactOpts,
},
contractBackend,
}, nil
}

// DeployENS deploys an instance of the ENS nameservice, with a 'first-in, first-served' root
registrar.
func DeployENS(transactOpts *bind.TransactOpts, contractBackend bind.ContractBackend)
(*ENS, error) {
// Deploy the ENS registry
ensAddr, _, _, err := contract.DeployENS(transactOpts, contractBackend, transactOpts.From)
if err != nil {
return nil, err
}

ens, err := NewENS(transactOpts, ensAddr, contractBackend)
if err != nil {
return nil, err
}

// Deploy the registrar
regAddr, _, _, err := contract.DeployFIFSRegistrar(transactOpts, contractBackend, ensAddr,
[32]byte{})
if err != nil {
return nil, err
}

// Set the registrar as owner of the ENS root
```

```
_, err = ens.SetOwner([32]byte{}, regAddr)
if err != nil {
return nil, err
}

return ens, nil
}

func ensParentNode(name string) (common.Hash, common.Hash) {
parts := strings.SplitN(name, ".", 2)
label := crypto.Keccak256Hash([]byte(parts[0]))
if len(parts) == 1 {
return [32]byte{}, label
} else {
parentNode, parentLabel := ensParentNode(parts[1])
return crypto.Keccak256Hash(parentNode[:], parentLabel[:]), label
}
}

func ensNode(name string) common.Hash {
parentNode, parentLabel := ensParentNode(name)
return crypto.Keccak256Hash(parentNode[:], parentLabel[:])
}

func (self *ENS) getResolver(node [32]byte) (*contract.PublicResolverSession, error) {
resolverAddr, err := self.Resolver(node)
if err != nil {
return nil, err
}

resolver, err := contract.NewPublicResolver(resolverAddr, self.contractBackend)
if err != nil {
return nil, err
}

return &contract.PublicResolverSession{
Contract:     resolver,
TransactOpts: self.TransactOpts,
}, nil
}

func (self *ENS) getRegistrar(node [32]byte) (*contract.FIFSRegistrarSession, error) {
```

```
registrarAddr, err := self.Owner(node)
if err != nil {
return nil, err
}

registrar, err := contract.NewFIFSRegistrar(registrarAddr, self.contractBackend)
if err != nil {
return nil, err
}

return &contract.FIFSRegistrarSession{
Contract:     registrar,
TransactOpts: self.TransactOpts,
}, nil
}

// Resolve is a non-transactional call that returns the content hash associated with a name.
func (self *ENS) Resolve(name string) (common.Hash, error) {
node := ensNode(name)

resolver, err := self.getResolver(node)
if err != nil {
return common.Hash{}, err
}

ret, err := resolver.Content(node)
if err != nil {
return common.Hash{}, err
}

return common.BytesToHash(ret[:]), nil
}

// Register registers a new domain name for the caller, making them the owner of the new name.
// Only works if the registrar for the parent domain implements the FIFS registrar protocol.
func (self *ENS) Register(name string) (*types.Transaction, error) {
parentNode, label := ensParentNode(name)

registrar, err := self.getRegistrar(parentNode)
if err != nil {
return nil, err
}
```

```
opts := self.TransactOpts
opts.GasLimit = big.NewInt(200000)
return registrar.Contract.Register(&opts, label, self.TransactOpts.From)
}

// SetContentHash sets the content hash associated with a name. Only works if the caller
// owns the name, and the associated resolver implements a `setContent` function.
func (self *ENS) SetContentHash(name string, hash common.Hash) (*types.Transaction, error) {
node := ensNode(name)

resolver, err := self.getResolver(node)
if err != nil {
return nil, err
}

opts := self.TransactOpts
opts.GasLimit = big.NewInt(200000)
return resolver.Contract.SetContent(&opts, node, hash)
}

52:F:\git\coin\ethereum\go-ethereum\contracts\ens\ens_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package ens

import (
"math/big"
"testing"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/accounts/abi/bind/backends"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/crypto"
)

var (
key, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
name   = "my name on ENS"
hash   = crypto.Keccak256Hash([]byte("my content"))
```

```
addr   = crypto.PubkeyToAddress(key.PublicKey)
)

func TestENS(t *testing.T) {
contractBackend := backends.NewSimulatedBackend(core.GenesisAlloc{addr: {Balance:
big.NewInt(1000000000)}})
transactOpts := bind.NewKeyedTransactor(key)
// Workaround for bug estimating gas in the call to Register
transactOpts.GasLimit = big.NewInt(1000000)

ens, err := DeployENS(transactOpts, contractBackend)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
contractBackend.Commit()

_, err = ens.Register(name)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
contractBackend.Commit()

_, err = ens.SetContentHash(name, hash)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
contractBackend.Commit()

vhost, err := ens.Resolve(name)
if err != nil {
t.Fatalf("expected no error, got %v", err)
}
if vhost != hash {
t.Fatalf("resolve error, expected %v, got %v", hash.Hex(), vhost.Hex())
}
}

53:F:\git\coin\ethereum\go-ethereum\contracts\release\contract.go

// ReleaseOracleABI is the input ABI used to generate the binding from.
const ReleaseOracleABI =
`[{"constant":true,"inputs":[],"name":"proposedVersion","outputs":[{"name":"major","type":"uint32"},{
```

"name":"minor","type":"uint32"},{"name":"patch","type":"uint32"},{"name":"commit","type":"bytes20"}
,{"name":"pass","type":"address[]"},{"name":"fail","type":"address[]"}],"type":"function"},{"constant":t
rue,"inputs":[],"name":"signers","outputs":[{"name":"","type":"address[]"}],"type":"function"},{"consta
nt":false,"inputs":[{"name":"user","type":"address"}],"name":"demote","outputs":[],"type":"function"},{
"constant":true,"inputs":[{"name":"user","type":"address"}],"name":"authVotes","outputs":[{"name":"
promote","type":"address[]"},{"name":"demote","type":"address[]"}],"type":"function"},{"constant":tru
e,"inputs":[],"name":"currentVersion","outputs":[{"name":"major","type":"uint32"},{"name":"minor","ty
pe":"uint32"},{"name":"patch","type":"uint32"},{"name":"commit","type":"bytes20"},{"name":"time","ty
pe":"uint256"}],"type":"function"},{"constant":false,"inputs":[],"name":"nuke","outputs":[],"type":"funct
ion"},{"constant":true,"inputs":[],"name":"authProposals","outputs":[{"name":"","type":"address[]"}],"t
ype":"function"},{"constant":false,"inputs":[{"name":"user","type":"address"}],"name":"promote","out
puts":[],"type":"function"},{"constant":false,"inputs":[{"name":"major","type":"uint32"},{"name":"minor
","type":"uint32"},{"name":"patch","type":"uint32"},{"name":"commit","type":"bytes20"}],"name":"rele
ase","outputs":[],"type":"function"},{"inputs":[{"name":"signers","type":"address[]"}],"type":"construct
or"}]`

// ReleaseOracleBin is the compiled bytecode used for deploying new contracts.
const ReleaseOracleBin =
`0x60606040526040516113533803806113538339810160405280510160008151600014156100
8457600160a060020a033316815260208190526040812080546 0ff191660019081179091558054 80
8201808355828183801582901161013357600 0
838152602090206101339181019083016100705 65b505050600092835250602090912001805460
0160a060020a031916331790555b50506111df806101746000396000f35b5090565b50505091909
0600052602060002090016000848481518110156100025750505060208381028501015181546 00
160a060020a031916179055506001016100885660606040523615610077576 0e060020a6000350
46326db76488114610079578063 46f0975a1461019e5780635c3d005d1461020a57806364ed31fe
146102935780639d888e861461038d578063bc8fbbf8146103b2578063bf8ecf9c146103fc578063d
0e0813a14610468578063d67cbec914610479575b005b6104966040805160208181018352600 8
0835283518083018552818152600454600680548751818702810187019098528085293968796 8
796879691959463ffffffff8181169564010000000830482169560406002 0a840490921694606060 0
20a9384900490930293909260079291849190830182828015610126576020028201919060005 26
020600020905b8154600160a060020a0316815260019190910190602001808311610107575b505
0505050915080805480602002602001604051908101604052809291908181526020018280548 01
561018357602002820191906000526020600020905b8154600160a060020a0316815260019190 9
101906020018083116101645 75b505050505090509550955095509550955095509550955091929394955
65b604080516020818101835260008252600180548451818402810184019095528085526105589
492830182828015610 1ff576020028201919060005260206000209 05b8154600160a060020a0316
815260019190910190602001808311610 1e0575b505050505090505b90565b6100776004356106
6d8160005b600160a060020a033316600090815260208190526040812054819060ff16156107005

7600160a060020a038416815260026020526040812091505b8154811015610706578154600160a
060020a033316908390839081101561000257600091825260209091200154600160a060020a031
6141561075157610700565b6105a26004356040805160208181018352600080835283518083018
552818152600160a060020a0386168252600283529084902080548551818502810185019096528
086529394919390926001840192918491830182828015610320576020028201919060005260206
00020905b8154600160a060020a0316815260019190910190602001808311610301575b5050505
050915080805480602002602001604051908101604052809291908181526020018280548015610
37d57602002820191906000526020600020905b8154600160a060020a03168152600191909101 9
060200180831161035e575b5050505050905091509150915091565b61062760006000600060006
000600060086000508054905060001415610670576106f1565b6100776106f96000808080805b60
0160a060020a033316600090815260208190526040812054819060ff16156111b65782158015610
3f257506006546000145b15610c2e576111b6565b6040805160208181018352600082526003805
484518184028101840190955280855261055894928301828280156101ff5760200282019190600 0
526020600020908154600160a060020a0316815260019190910190602001808311610 1e0575b50
50505050905061020756 5b61007760043561066d816001610217565b6100776004356024356044
3560643561070084848484600161 03bf565b604051808763ffffffff1681526020018663ffffffff168152
6020018563ffffffff1681526020018 46bffffffffffffffffffffffff191681526020018060200180602001838103
8352858181518152602001915080519060200190602002808383829060006004602084601f0104
600302600f01f150905001838103825284818151815260200191508051906020019060200280838
382906000600460208460 1f0104600302600f01f150905001985050505050505050506040518091
0390f35b6040518080602001828103825283818151815260200191508051906020019060200280
8383829060006004602084601f0104600302600f01f1509050019250505060405180910390f35b60
4051808060200180602001838103835285818151815260200191508051906020019060200280 83
8382906000600460208460 1f0104600302600f01f15090500183810382528481815181526020019
1508051906020019060200280838382906000600460208460 1f0104600302600f01f15090500194
5050505050604051809103 90f35b6040805163ffffffff968716815294861660208601529290941683
8301526bffffffffffffffffffffffff19166060830152608082019290925290519081900360a00190f35b5056
5b600880546000198101908110156100025760009182526004027ff3f7a9fe364faab93b216da50a
3214154f22a0a2b415b23a84c8169e8b636ee30190508054600182015463ffffffff82811699506401
00000000830481169850604060020a8304169650606060020a91829004909102945067ffffffffffffffff
16925090505b50909192939456 5b565b505050505b50505050565b5060005b6001820154811015
6107595733600160a060020a031682600101600050828154811015610002576000918252602090
91200154600160a060020a031614156107a357610700565b600101610252565b81546000148015
61076e575060018201546000145b156107cb5760038054600181018083558281838015829011 61
07ab57818360005260206000209182019101610 7ab9190610851565b60010161070a565b505050
6000928352506020909120018054600160a060020a031916851790555b82156108695781546001
8101808455839190828183801582901161089e576000838152602090 2061089e91810190830161
0851565b5050506000928352506020909120018054600160a060020a031916851790555b600160
a060020a0384166000908152600260209081526040822080548382558184529183209092916 10b
2f91908101905b808211156108655760008155600101610851565b5090565b8160010160005080
5480600101828181548183558181151161095057818360005260206000209182019101610 95091
90610851565b5050506000928352506020909120018054600160a060020a031916331790556001

54825460029091049011610 8d257610700565b8280156108f85750600160a060020a0384166000
90815260208190526040902054 60ff16155b1561098757600160a060020a0384166000908152602
08190526040902080546 0ff191660019081179091558054808201808355828138015829011610 8
00578183600052602060002091820191016108009190610851565b5050506000928352506020 90
9120018054600160a060020a0319163317905560018054908301546002909104901161 08d25761
0700565b821580156109ad5750600160a060020a0384166000908152602081905260409020546 0
ff165b1561082057506001 60a060020a0383166000908152602081905260408120805460ff191690
555b60015481101561082057836001 60a060020a0316600160005082815481101561000257600 0
91825260209091200154600160a060020a03161415610aa3576001805460001981019081101561
0002576020600090812092905260018054929091015460016 0a060020a0316918390811015610 0
02579060005260206000209001600061010 00a815481600160a060020a0302191690830217905 5
506001600050805480919060019003909081548183558181151161 0aab5760008381526020902 0
610aab9181019083016108515 65b6001016109d4565b5050600060048181556005805467ffffffffffff
ffff191690556006805483825581845291945091925082906 10b05907ff652222313e28459528d920
b65115c16c04f3efc82aaedc97be59f3f377c0d3f9081019061085156 5b5060018201805460008083
559182526020909120610b259181019061085156 5b5050505050610820565b5060018201805460
008083559182526020909120610b4f9181019061085156 5b506000925050505b60035481101561
07005783600160a060020a0316600360005082815481101561000257600091825260209091200 1
54600160a060020a03161415610c2657600380546000198101908110156100025760206000908 1
209290526003805492909101546001 60a060020a0316918390811015610002579060005260206 0
002090016000610 1000a815481600160a060020a0302191690830217905550600360005080548 0
919060019003909081548183558181151161 06fb576000838152602090206106fb9181019083016
10851565b600101610b57565b60065460001415610c8c576004805463ffffffff1916881767ffffffff000
000001916640100000000 8802176bffffffff0000000000000000191660406 0020a8702176bfffffffffffff
fffffffff16606060020a808704021790555b828015610d08575060045463ffffffff88811691161415806
10cc1575060045463ffffffff878116640100000000909204161415 5b80610cde575060045463ffffffff8
68116604060020a9092041614155b80610d08575060045460606 0020a90819004026bfffffffffffffffff
ffffff199081169085161 4155b15610d12576111b6565b506006905060005b8154811015610d5b578
154600160a060020a0333169083908390811015610002576000918252602090912001 54600160a
060020a03161415610da6576111b6565b5060005b600182015481101561 0dae5733600160a0600
20a0316826001016000508281548110156100025760009182526020909120015 4600160a060020
a03161415610de3576111b6565b600101610d1a565b8215610deb57815460018101808455839 19
082818380158290116 10e2057600083815260209020610e209181019083016108515 65b6001016
10d5f565b816001016000508054806001018281815481835581811511610ea3578183600052602
060002091820191 01610ea39190610851565b5050506000928352506020909120018054600160a
060020a0319163317905560015482546002909104901161 0e54576111b6565b8215610eda57600
5805467ffffffffffffffff191642179055600880546001810180835582818380158290116 10f2f57600402
816004028360005260206000209182019101610f2f9190611048565b5050506000928352506020 9
091200180546001 60a060020a0319163317905560018054908301546002909104901161 0e54576
111b6565b60006004818155600580546 7ffffffffffffffff191690556006805483825581845291929182 9
06111bf907ff652222313e28459528d920b65115c16c04f3efc82aaedc97be59f3f377c0d3f9081019
0610851565b5050509190906000526020600020906004020160005060048054825463ffffffff19166

3fffffff918216178084558254640100000000908190048316026 7fffffff0000000199190911617808
4558254604060020a908190049092169091026bfffffff0000000000000000199190911617808355 8
154606060020a90819004810281900 4026bfffffffffffffffffffffff919091161782556005546001830180 5
467fffffffffffffff191667fffffffffffffff909216919091179055600680546002840180548282556000828 15
260209020949594919283929182019185821561 10a75760005260206000209182015b828111156
110a75782548255916001019190600101906110 25565b505050506004015b80821115610865576
000808255600182018054 67fffffffffffffff191690556002820180548282558183526020832083916 11
087919081019061085156 5b506001820180546000808355918252602090912061104091810190 6
10851565b506110cd9291505b8082111561086557805460016 0a060020a031916815560010161 1
0af565b505060018181018054918401805480835560008381526020902092938301929091821 56
1111b576000526020600020918201 5b8281111561111b57825482559160010191906001019061 1
100565b506111279291506110af565b505060006004818155600580546 7fffffffffffffff19169055600
6805483825581845291975091955090935084925061118691507ff652222313e28459528d920b65
115c16c04f3efc82aaedc97be59f3f377c0d3f90810190610851565b506001820180546000808355 9
1825260209091206111a691810190610851565b50505050506111b6565b50505050505b5050505
0505050565b506001820180546000808355918252602090912061 11b09181019061085156`

// DeployReleaseOracle deploys a new Ethereum contract, binding an instance of ReleaseOracle to it.
func DeployReleaseOracle(auth *bind.TransactOpts, backend bind.ContractBackend, signers []common.Address) (common.Address, *types.Transaction, *ReleaseOracle, error) {
parsed, err := abi.JSON(strings.NewReader(ReleaseOracleABI))
if err != nil {
return common.Address{}, nil, nil, err
}
address, tx, contract, err := bind.DeployContract(auth, parsed,
common.FromHex(ReleaseOracleBin), backend, signers)
if err != nil {
return common.Address{}, nil, nil, err
}
return address, tx, &ReleaseOracle{ReleaseOracleCaller: ReleaseOracleCaller{contract: contract},
ReleaseOracleTransactor: ReleaseOracleTransactor{contract: contract}}, nil
}

// ReleaseOracle is an auto generated Go binding around an Ethereum contract.
type ReleaseOracle struct {
ReleaseOracleCaller   // Read-only binding to the contract
ReleaseOracleTransactor // Write-only binding to the contract
}

// ReleaseOracleCaller is an auto generated read-only Go binding around an Ethereum contract.
type ReleaseOracleCaller struct {

```
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ReleaseOracleTransactor is an auto generated write-only Go binding around an Ethereum
contract.
type ReleaseOracleTransactor struct {
contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// ReleaseOracleSession is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type ReleaseOracleSession struct {
Contract     *ReleaseOracle     // Generic contract binding to set the session for
CallOpts     bind.CallOpts      // Call options to use throughout this session
TransactOpts bind.TransactOpts // Transaction auth options to use throughout this session
}

// ReleaseOracleCallerSession is an auto generated read-only Go binding around an Ethereum
contract,
// with pre-set call options.
type ReleaseOracleCallerSession struct {
Contract *ReleaseOracleCaller // Generic contract caller binding to set the session for
CallOpts bind.CallOpts        // Call options to use throughout this session
}

// ReleaseOracleTransactorSession is an auto generated write-only Go binding around an
Ethereum contract,
// with pre-set transact options.
type ReleaseOracleTransactorSession struct {
Contract     *ReleaseOracleTransactor // Generic contract transactor binding to set the session for
TransactOpts bind.TransactOpts        // Transaction auth options to use throughout this session
}

// ReleaseOracleRaw is an auto generated low-level Go binding around an Ethereum contract.
type ReleaseOracleRaw struct {
Contract *ReleaseOracle // Generic contract binding to access the raw methods on
}

// ReleaseOracleCallerRaw is an auto generated low-level read-only Go binding around an
Ethereum contract.
type ReleaseOracleCallerRaw struct {
Contract *ReleaseOracleCaller // Generic read-only contract binding to access the raw methods
```

```
on
}

// ReleaseOracleTransactorRaw is an auto generated low-level write-only Go binding around an
Ethereum contract.
type ReleaseOracleTransactorRaw struct {
Contract *ReleaseOracleTransactor // Generic write-only contract binding to access the raw
methods on
}

// NewReleaseOracle creates a new instance of ReleaseOracle, bound to a specific deployed
contract.
func NewReleaseOracle(address common.Address, backend bind.ContractBackend)
(*ReleaseOracle, error) {
contract, err := bindReleaseOracle(address, backend, backend)
if err != nil {
return nil, err
}
return &ReleaseOracle{ReleaseOracleCaller: ReleaseOracleCaller{contract: contract},
ReleaseOracleTransactor: ReleaseOracleTransactor{contract: contract}}, nil
}

// NewReleaseOracleCaller creates a new read-only instance of ReleaseOracle, bound to a
specific deployed contract.
func NewReleaseOracleCaller(address common.Address, caller bind.ContractCaller)
(*ReleaseOracleCaller, error) {
contract, err := bindReleaseOracle(address, caller, nil)
if err != nil {
return nil, err
}
return &ReleaseOracleCaller{contract: contract}, nil
}

// NewReleaseOracleTransactor creates a new write-only instance of ReleaseOracle, bound to a
specific deployed contract.
func NewReleaseOracleTransactor(address common.Address, transactor
bind.ContractTransactor) (*ReleaseOracleTransactor, error) {
contract, err := bindReleaseOracle(address, nil, transactor)
if err != nil {
return nil, err
}
return &ReleaseOracleTransactor{contract: contract}, nil
```

```
}

// bindReleaseOracle binds a generic wrapper to an already deployed contract.
func bindReleaseOracle(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
parsed, err := abi.JSON(strings.NewReader(ReleaseOracleABI))
if err != nil {
return nil, err
}
return bind.NewBoundContract(address, parsed, caller, transactor), nil
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_ReleaseOracle *ReleaseOracleRaw) Call(opts *bind.CallOpts, result interface{}, method
string, params ...interface{}) error {
return _ReleaseOracle.Contract.ReleaseOracleCaller.contract.Call(opts, result, method,
params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_ReleaseOracle *ReleaseOracleRaw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _ReleaseOracle.Contract.ReleaseOracleTransactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_ReleaseOracle *ReleaseOracleRaw) Transact(opts *bind.TransactOpts, method string,
params ...interface{}) (*types.Transaction, error) {
return _ReleaseOracle.Contract.ReleaseOracleTransactor.contract.Transact(opts, method,
params...)
}

// Call invokes the (constant) contract method with params as input values and
// sets the output to result. The result type might be a single field for simple
// returns, a slice of interfaces for anonymous returns and a struct for named
// returns.
func (_ReleaseOracle *ReleaseOracleCallerRaw) Call(opts *bind.CallOpts, result interface{},
method string, params ...interface{}) error {
```

```
return _ReleaseOracle.Contract.contract.Call(opts, result, method, params...)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (_ReleaseOracle *ReleaseOracleTransactorRaw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _ReleaseOracle.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_ReleaseOracle *ReleaseOracleTransactorRaw) Transact(opts *bind.TransactOpts, method
string, params ...interface{}) (*types.Transaction, error) {
return _ReleaseOracle.Contract.contract.Transact(opts, method, params...)
}

// AuthProposals is a free data retrieval call binding the contract method 0xbf8ecf9c.
//
// Solidity: function authProposals() constant returns(address[])
func (_ReleaseOracle *ReleaseOracleCaller) AuthProposals(opts *bind.CallOpts)
([]common.Address, error) {
var (
ret0 = new([]common.Address)
)
out := ret0
err := _ReleaseOracle.contract.Call(opts, out, "authProposals")
return *ret0, err
}

// AuthProposals is a free data retrieval call binding the contract method 0xbf8ecf9c.
//
// Solidity: function authProposals() constant returns(address[])
func (_ReleaseOracle *ReleaseOracleSession) AuthProposals() ([]common.Address, error) {
return _ReleaseOracle.Contract.AuthProposals(&_ReleaseOracle.CallOpts)
}

// AuthProposals is a free data retrieval call binding the contract method 0xbf8ecf9c.
//
// Solidity: function authProposals() constant returns(address[])
func (_ReleaseOracle *ReleaseOracleCallerSession) AuthProposals() ([]common.Address, error) {
return _ReleaseOracle.Contract.AuthProposals(&_ReleaseOracle.CallOpts)
}
```

```
// AuthVotes is a free data retrieval call binding the contract method 0x64ed31fe.
//
// Solidity: function authVotes(user address) constant returns(promote address[], demote address[])
func (_ReleaseOracle *ReleaseOracleCaller) AuthVotes(opts *bind.CallOpts, user common.Address) (struct {
Promote []common.Address
Demote  []common.Address
}, error) {
ret := new(struct {
Promote []common.Address
Demote  []common.Address
})
out := ret
err := _ReleaseOracle.contract.Call(opts, out, "authVotes", user)
return *ret, err
}

// AuthVotes is a free data retrieval call binding the contract method 0x64ed31fe.
//
// Solidity: function authVotes(user address) constant returns(promote address[], demote address[])
func (_ReleaseOracle *ReleaseOracleSession) AuthVotes(user common.Address) (struct {
Promote []common.Address
Demote  []common.Address
}, error) {
return _ReleaseOracle.Contract.AuthVotes(&_ReleaseOracle.CallOpts, user)
}

// AuthVotes is a free data retrieval call binding the contract method 0x64ed31fe.
//
// Solidity: function authVotes(user address) constant returns(promote address[], demote address[])
func (_ReleaseOracle *ReleaseOracleCallerSession) AuthVotes(user common.Address) (struct {
Promote []common.Address
Demote  []common.Address
}, error) {
return _ReleaseOracle.Contract.AuthVotes(&_ReleaseOracle.CallOpts, user)
}

// CurrentVersion is a free data retrieval call binding the contract method 0x9d888e86.
```

```
//
// Solidity: function currentVersion() constant returns(major uint32, minor uint32, patch uint32,
commit bytes20, time uint256)
func (_ReleaseOracle *ReleaseOracleCaller) CurrentVersion(opts *bind.CallOpts) (struct {
Major  uint32
Minor  uint32
Patch  uint32
Commit [20]byte
Time   *big.Int
}, error) {
ret := new(struct {
Major  uint32
Minor  uint32
Patch  uint32
Commit [20]byte
Time   *big.Int
})
out := ret
err := _ReleaseOracle.contract.Call(opts, out, "currentVersion")
return *ret, err
}

// CurrentVersion is a free data retrieval call binding the contract method 0x9d888e86.
//
// Solidity: function currentVersion() constant returns(major uint32, minor uint32, patch uint32,
commit bytes20, time uint256)
func (_ReleaseOracle *ReleaseOracleSession) CurrentVersion() (struct {
Major  uint32
Minor  uint32
Patch  uint32
Commit [20]byte
Time   *big.Int
}, error) {
return _ReleaseOracle.Contract.CurrentVersion(&_ReleaseOracle.CallOpts)
}

// CurrentVersion is a free data retrieval call binding the contract method 0x9d888e86.
//
// Solidity: function currentVersion() constant returns(major uint32, minor uint32, patch uint32,
commit bytes20, time uint256)
func (_ReleaseOracle *ReleaseOracleCallerSession) CurrentVersion() (struct {
Major  uint32
```

```
Minor  uint32
Patch  uint32
Commit [20]byte
Time   *big.Int
}, error) {
return _ReleaseOracle.Contract.CurrentVersion(&_ReleaseOracle.CallOpts)
}

// ProposedVersion is a free data retrieval call binding the contract method 0x26db7648.
//
// Solidity: function proposedVersion() constant returns(major uint32, minor uint32, patch uint32,
commit bytes20, pass address[], fail address[])
func (_ReleaseOracle *ReleaseOracleCaller) ProposedVersion(opts *bind.CallOpts) (struct {
Major  uint32
Minor  uint32
Patch  uint32
Commit [20]byte
Pass   []common.Address
Fail   []common.Address
}, error) {
ret := new(struct {
Major  uint32
Minor  uint32
Patch  uint32
Commit [20]byte
Pass   []common.Address
Fail   []common.Address
})
out := ret
err := _ReleaseOracle.contract.Call(opts, out, "proposedVersion")
return *ret, err
}

// ProposedVersion is a free data retrieval call binding the contract method 0x26db7648.
//
// Solidity: function proposedVersion() constant returns(major uint32, minor uint32, patch uint32,
commit bytes20, pass address[], fail address[])
func (_ReleaseOracle *ReleaseOracleSession) ProposedVersion() (struct {
Major  uint32
Minor  uint32
Patch  uint32
Commit [20]byte
```

```
Pass   []common.Address
Fail   []common.Address
}, error) {
return _ReleaseOracle.Contract.ProposedVersion(&_ReleaseOracle.CallOpts)
}

// ProposedVersion is a free data retrieval call binding the contract method 0x26db7648.
//
// Solidity: function proposedVersion() constant returns(major uint32, minor uint32, patch uint32,
commit bytes20, pass address[], fail address[])
func (_ReleaseOracle *ReleaseOracleCallerSession) ProposedVersion() (struct {
Major  uint32
Minor  uint32
Patch  uint32
Commit [20]byte
Pass   []common.Address
Fail   []common.Address
}, error) {
return _ReleaseOracle.Contract.ProposedVersion(&_ReleaseOracle.CallOpts)
}

// Signers is a free data retrieval call binding the contract method 0x46f0975a.
//
// Solidity: function signers() constant returns(address[])
func (_ReleaseOracle *ReleaseOracleCaller) Signers(opts *bind.CallOpts) ([]common.Address,
error) {
var (
ret0 = new([]common.Address)
)
out := ret0
err := _ReleaseOracle.contract.Call(opts, out, "signers")
return *ret0, err
}

// Signers is a free data retrieval call binding the contract method 0x46f0975a.
//
// Solidity: function signers() constant returns(address[])
func (_ReleaseOracle *ReleaseOracleSession) Signers() ([]common.Address, error) {
return _ReleaseOracle.Contract.Signers(&_ReleaseOracle.CallOpts)
}

// Signers is a free data retrieval call binding the contract method 0x46f0975a.
```

```
//
// Solidity: function signers() constant returns(address[])
func (_ReleaseOracle *ReleaseOracleCallerSession) Signers() ([]common.Address, error) {
return _ReleaseOracle.Contract.Signers(&_ReleaseOracle.CallOpts)
}

// Demote is a paid mutator transaction binding the contract method 0x5c3d005d.
//
// Solidity: function demote(user address) returns()
func (_ReleaseOracle *ReleaseOracleTransactor) Demote(opts *bind.TransactOpts, user
common.Address) (*types.Transaction, error) {
return _ReleaseOracle.contract.Transact(opts, "demote", user)
}

// Demote is a paid mutator transaction binding the contract method 0x5c3d005d.
//
// Solidity: function demote(user address) returns()
func (_ReleaseOracle *ReleaseOracleSession) Demote(user common.Address)
(*types.Transaction, error) {
return _ReleaseOracle.Contract.Demote(&_ReleaseOracle.TransactOpts, user)
}

// Demote is a paid mutator transaction binding the contract method 0x5c3d005d.
//
// Solidity: function demote(user address) returns()
func (_ReleaseOracle *ReleaseOracleTransactorSession) Demote(user common.Address)
(*types.Transaction, error) {
return _ReleaseOracle.Contract.Demote(&_ReleaseOracle.TransactOpts, user)
}

// Nuke is a paid mutator transaction binding the contract method 0xbc8fbbf8.
//
// Solidity: function nuke() returns()
func (_ReleaseOracle *ReleaseOracleTransactor) Nuke(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _ReleaseOracle.contract.Transact(opts, "nuke")
}

// Nuke is a paid mutator transaction binding the contract method 0xbc8fbbf8.
//
// Solidity: function nuke() returns()
func (_ReleaseOracle *ReleaseOracleSession) Nuke() (*types.Transaction, error) {
```

```
return _ReleaseOracle.Contract.Nuke(&_ReleaseOracle.TransactOpts)
}

// Nuke is a paid mutator transaction binding the contract method 0xbc8fbbf8.
//
// Solidity: function nuke() returns()
func (_ReleaseOracle *ReleaseOracleTransactorSession) Nuke() (*types.Transaction, error) {
return _ReleaseOracle.Contract.Nuke(&_ReleaseOracle.TransactOpts)
}

// Promote is a paid mutator transaction binding the contract method 0xd0e0813a.
//
// Solidity: function promote(user address) returns()
func (_ReleaseOracle *ReleaseOracleTransactor) Promote(opts *bind.TransactOpts, user
common.Address) (*types.Transaction, error) {
return _ReleaseOracle.contract.Transact(opts, "promote", user)
}

// Promote is a paid mutator transaction binding the contract method 0xd0e0813a.
//
// Solidity: function promote(user address) returns()
func (_ReleaseOracle *ReleaseOracleSession) Promote(user common.Address)
(*types.Transaction, error) {
return _ReleaseOracle.Contract.Promote(&_ReleaseOracle.TransactOpts, user)
}

// Promote is a paid mutator transaction binding the contract method 0xd0e0813a.
//
// Solidity: function promote(user address) returns()
func (_ReleaseOracle *ReleaseOracleTransactorSession) Promote(user common.Address)
(*types.Transaction, error) {
return _ReleaseOracle.Contract.Promote(&_ReleaseOracle.TransactOpts, user)
}

// Release is a paid mutator transaction binding the contract method 0xd67cbec9.
//
// Solidity: function release(major uint32, minor uint32, patch uint32, commit bytes20) returns()
func (_ReleaseOracle *ReleaseOracleTransactor) Release(opts *bind.TransactOpts, major uint32,
minor uint32, patch uint32, commit [20]byte) (*types.Transaction, error) {
return _ReleaseOracle.contract.Transact(opts, "release", major, minor, patch, commit)
}
```

```
// Release is a paid mutator transaction binding the contract method 0xd67cbec9.
//
// Solidity: function release(major uint32, minor uint32, patch uint32, commit bytes20) returns()
func (_ReleaseOracle *ReleaseOracleSession) Release(major uint32, minor uint32, patch uint32,
commit [20]byte) (*types.Transaction, error) {
return _ReleaseOracle.Contract.Release(&_ReleaseOracle.TransactOpts, major, minor, patch,
commit)
}

// Release is a paid mutator transaction binding the contract method 0xd67cbec9.
//
// Solidity: function release(major uint32, minor uint32, patch uint32, commit bytes20) returns()
func (_ReleaseOracle *ReleaseOracleTransactorSession) Release(major uint32, minor uint32,
patch uint32, commit [20]byte) (*types.Transaction, error) {
return _ReleaseOracle.Contract.Release(&_ReleaseOracle.TransactOpts, major, minor, patch,
commit)
}

54:F:\git\coin\ethereum\go-ethereum\contracts\release\contract_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package release

import (
"crypto/ecdsa"
"math/big"
"testing"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/accounts/abi/bind/backends"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/crypto"
)

// setupReleaseTest creates a blockchain simulator and deploys a version oracle
// contract for testing.
func setupReleaseTest(t *testing.T, prefund ...*ecdsa.PrivateKey) (*ecdsa.PrivateKey,
*ReleaseOracle, *backends.SimulatedBackend) {
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
```

```
alloc := core.GenesisAlloc{auth.From: {Balance: big.NewInt(10000000000)}}
for _, key := range prefund {
alloc[crypto.PubkeyToAddress(key.PublicKey)] = core.GenesisAccount{Balance:
big.NewInt(10000000000)}
}
sim := backends.NewSimulatedBackend(alloc)

// Deploy a version oracle contract, commit and return
_, _, oracle, err := DeployReleaseOracle(auth, sim, []common.Address{auth.From})
if err != nil {
t.Fatalf("Failed to deploy version contract: %v", err)
}
sim.Commit()

return key, oracle, sim
}

// Tests that the version contract can be deployed and the creator is assigned
// the sole authorized signer.
func TestContractCreation(t *testing.T) {
key, oracle, _ := setupReleaseTest(t)

owner := crypto.PubkeyToAddress(key.PublicKey)
signers, err := oracle.Signers(nil)
if err != nil {
t.Fatalf("Failed to retrieve list of signers: %v", err)
}
if len(signers) != 1 || signers[0] != owner {
t.Fatalf("Initial signer mismatch: have %v, want %v", signers, owner)
}
}

// Tests that subsequent signers can be promoted, each requiring half plus one
// votes for it to pass through.
func TestSignerPromotion(t *testing.T) {
// Prefund a few accounts to authorize with and create the oracle
keys := make([]*ecdsa.PrivateKey, 5)
for i := 0; i < len(keys); i++ {
keys[i], _ = crypto.GenerateKey()
}
key, oracle, sim := setupReleaseTest(t, keys...)
```

```
// Gradually promote the keys, until all are authorized
keys = append([]*ecdsa.PrivateKey{key}, keys...)
for i := 1; i < len(keys); i++ {
// Check that no votes are accepted from the not yet authorized user
if _, err := oracle.Promote(bind.NewKeyedTransactor(keys[i]), common.Address{}); err != nil {
t.Fatalf("Iter #%d: failed invalid promotion attempt: %v", i, err)
}
sim.Commit()

pend, err := oracle.AuthProposals(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve active proposals: %v", i, err)
}
if len(pend) != 0 {
t.Fatalf("Iter #%d: proposal count mismatch: have %d, want 0", i, len(pend))
}
// Promote with half - 1 voters and check that the user's not yet authorized
for j := 0; j < i/2; j++ {
if _, err = oracle.Promote(bind.NewKeyedTransactor(keys[j]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid promotion attempt: %v", i, err)
}
}
sim.Commit()

signers, err := oracle.Signers(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve list of signers: %v", i, err)
}
if len(signers) != i {
t.Fatalf("Iter #%d: signer count mismatch: have %v, want %v", i, len(signers), i)
}
// Promote with the last one needed to pass the promotion
if _, err = oracle.Promote(bind.NewKeyedTransactor(keys[i/2]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid promotion completion attempt: %v", i, err)
}
sim.Commit()

signers, err = oracle.Signers(nil)
if err != nil {
```

```
t.Fatalf("Iter #%d: failed to retrieve list of signers: %v", i, err)
}
if len(signers) != i+1 {
t.Fatalf("Iter #%d: signer count mismatch: have %v, want %v", i, len(signers), i+1)
}
}
}

// Tests that subsequent signers can be demoted, each requiring half plus one
// votes for it to pass through.
func TestSignerDemotion(t *testing.T) {
// Prefund a few accounts to authorize with and create the oracle
keys := make([]*ecdsa.PrivateKey, 5)
for i := 0; i < len(keys); i++ {
keys[i], _ = crypto.GenerateKey()
}
key, oracle, sim := setupReleaseTest(t, keys...)

// Authorize all the keys as valid signers and verify cardinality
keys = append([]*ecdsa.PrivateKey{key}, keys...)
for i := 1; i < len(keys); i++ {
for j := 0; j <= i/2; j++ {
if _, err := oracle.Promote(bind.NewKeyedTransactor(keys[j]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid promotion attempt: %v", i, err)
}
}
sim.Commit()
}
signers, err := oracle.Signers(nil)
if err != nil {
t.Fatalf("Failed to retrieve list of signers: %v", err)
}
if len(signers) != len(keys) {
t.Fatalf("Signer count mismatch: have %v, want %v", len(signers), len(keys))
}
// Gradually demote users until we run out of signers
for i := len(keys) - 1; i >= 0; i-- {
// Demote with half - 1 voters and check that the user's not yet dropped
for j := 0; j < (i+1)/2; j++ {
if _, err = oracle.Demote(bind.NewKeyedTransactor(keys[j]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
```

```
t.Fatalf("Iter #%d: failed valid demotion attempt: %v", len(keys)-i, err)
}
}
sim.Commit()

signers, err := oracle.Signers(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve list of signers: %v", len(keys)-i, err)
}
if len(signers) != i+1 {
t.Fatalf("Iter #%d: signer count mismatch: have %v, want %v", len(keys)-i, len(signers), i+1)
}
// Demote with the last one needed to pass the demotion
if _, err = oracle.Demote(bind.NewKeyedTransactor(keys[(i+1)/2]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid demotion completion attempt: %v", i, err)
}
sim.Commit()

signers, err = oracle.Signers(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve list of signers: %v", len(keys)-i, err)
}
if len(signers) != i {
t.Fatalf("Iter #%d: signer count mismatch: have %v, want %v", len(keys)-i, len(signers), i)
}
// Check that no votes are accepted from the already demoted users
if _, err = oracle.Promote(bind.NewKeyedTransactor(keys[i]), common.Address{}); err != nil {
t.Fatalf("Iter #%d: failed invalid promotion attempt: %v", i, err)
}
sim.Commit()

pend, err := oracle.AuthProposals(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve active proposals: %v", i, err)
}
if len(pend) != 0 {
t.Fatalf("Iter #%d: proposal count mismatch: have %d, want 0", i, len(pend))
}
}
}
```

```
// Tests that new versions can be released, honouring both voting rights as well
// as the minimum required vote count.
func TestVersionRelease(t *testing.T) {
// Prefund a few accounts to authorize with and create the oracle
keys := make([]*ecdsa.PrivateKey, 5)
for i := 0; i < len(keys); i++ {
keys[i], _ = crypto.GenerateKey()
}
key, oracle, sim := setupReleaseTest(t, keys...)

// Track the "current release"
var (
verMajor  = uint32(0)
verMinor  = uint32(0)
verPatch  = uint32(0)
verCommit = [20]byte{}
)
// Gradually push releases, always requiring more signers than previously
keys = append([]*ecdsa.PrivateKey{key}, keys...)
for i := 1; i < len(keys); i++ {
// Check that no votes are accepted from the not yet authorized user
if _, err := oracle.Release(bind.NewKeyedTransactor(keys[i]), 0, 0, 0, [20]byte{0}); err != nil {
t.Fatalf("Iter #%d: failed invalid release attempt: %v", i, err)
}
sim.Commit()

prop, err := oracle.ProposedVersion(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve active proposal: %v", i, err)
}
if len(prop.Pass) != 0 {
t.Fatalf("Iter #%d: proposal vote count mismatch: have %d, want 0", i, len(prop.Pass))
}
// Authorize the user to make releases
for j := 0; j <= i/2; j++ {
if _, err = oracle.Promote(bind.NewKeyedTransactor(keys[j]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid promotion attempt: %v", i, err)
}
}
sim.Commit()
```

```
// Propose release with half voters and check that the release does not yet go through
for j := 0; j < (i+1)/2; j++ {
if _, err = oracle.Release(bind.NewKeyedTransactor(keys[j]), uint32(i), uint32(i+1), uint32(i+2),
[20]byte{byte(i + 3)}); err != nil {
t.Fatalf("Iter #%d: failed valid release attempt: %v", i, err)
}
}
sim.Commit()

ver, err := oracle.CurrentVersion(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve current version: %v", i, err)
}
if ver.Major != verMajor || ver.Minor != verMinor || ver.Patch != verPatch || ver.Commit !=
verCommit {
t.Fatalf("Iter #%d: version mismatch: have %d.%d.%d-%x, want %d.%d.%d-%x", i, ver.Major,
ver.Minor, ver.Patch, ver.Commit, verMajor, verMinor, verPatch, verCommit)
}

// Pass the release and check that it became the next version
verMajor, verMinor, verPatch, verCommit = uint32(i), uint32(i+1), uint32(i+2), [20]byte{byte(i + 3)}
if _, err = oracle.Release(bind.NewKeyedTransactor(keys[(i+1)/2]), uint32(i), uint32(i+1),
uint32(i+2), [20]byte{byte(i + 3)}); err != nil {
t.Fatalf("Iter #%d: failed valid release completion attempt: %v", i, err)
}
sim.Commit()

ver, err = oracle.CurrentVersion(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve current version: %v", i, err)
}
if ver.Major != verMajor || ver.Minor != verMinor || ver.Patch != verPatch || ver.Commit !=
verCommit {
t.Fatalf("Iter #%d: version mismatch: have %d.%d.%d-%x, want %d.%d.%d-%x", i, ver.Major,
ver.Minor, ver.Patch, ver.Commit, verMajor, verMinor, verPatch, verCommit)
}
}
}

// Tests that proposed versions can be nuked out of existence.
func TestVersionNuking(t *testing.T) {
// Prefund a few accounts to authorize with and create the oracle
```

```
keys := make([]*ecdsa.PrivateKey, 9)
for i := 0; i < len(keys); i++ {
keys[i], _ = crypto.GenerateKey()
}
key, oracle, sim := setupReleaseTest(t, keys...)

// Authorize all the keys as valid signers
keys = append([]*ecdsa.PrivateKey{key}, keys...)
for i := 1; i < len(keys); i++ {
for j := 0; j <= i/2; j++ {
if _, err := oracle.Promote(bind.NewKeyedTransactor(keys[j]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid promotion attempt: %v", i, err)
}
}
sim.Commit()
}
// Propose releases with more and more keys, always retaining enough users to nuke the
proposals
for i := 1; i < (len(keys)+1)/2; i++ {
// Propose release with an initial set of signers
for j := 0; j < i; j++ {
if _, err := oracle.Release(bind.NewKeyedTransactor(keys[j]), uint32(i), uint32(i+1), uint32(i+2),
[20]byte{byte(i + 3)}); err != nil {
t.Fatalf("Iter #%d: failed valid proposal attempt: %v", i, err)
}
}
sim.Commit()

prop, err := oracle.ProposedVersion(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve active proposal: %v", i, err)
}
if len(prop.Pass) != i {
t.Fatalf("Iter #%d: proposal vote count mismatch: have %d, want %d", i, len(prop.Pass), i)
}
// Nuke the release with half+1 voters
for j := i; j <= i+(len(keys)+1)/2; j++ {
if _, err := oracle.Nuke(bind.NewKeyedTransactor(keys[j])); err != nil {
t.Fatalf("Iter #%d: failed valid nuke attempt: %v", i, err)
}
}
```

```
sim.Commit()

prop, err = oracle.ProposedVersion(nil)
if err != nil {
t.Fatalf("Iter #%d: failed to retrieve active proposal: %v", i, err)
}
if len(prop.Pass) != 0 || len(prop.Fail) != 0 {
t.Fatalf("Iter #%d: proposal vote count mismatch: have %d/%d pass/fail, want 0/0", i,
len(prop.Pass), len(prop.Fail))
}
}
}

// Tests that demoting a signer will auto-nuke the currently pending release.
func TestVersionAutoNuke(t *testing.T) {
// Prefund a few accounts to authorize with and create the oracle
keys := make([]*ecdsa.PrivateKey, 5)
for i := 0; i < len(keys); i++ {
keys[i], _ = crypto.GenerateKey()
}
key, oracle, sim := setupReleaseTest(t, keys...)

// Authorize all the keys as valid signers
keys = append([]*ecdsa.PrivateKey{key}, keys...)
for i := 1; i < len(keys); i++ {
for j := 0; j <= i/2; j++ {
if _, err := oracle.Promote(bind.NewKeyedTransactor(keys[j]),
crypto.PubkeyToAddress(keys[i].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid promotion attempt: %v", i, err)
}
}
sim.Commit()
}
// Make a release proposal and check it's existence
if _, err := oracle.Release(bind.NewKeyedTransactor(keys[0]), 1, 2, 3, [20]byte{4}); err != nil {
t.Fatalf("Failed valid proposal attempt: %v", err)
}
sim.Commit()

prop, err := oracle.ProposedVersion(nil)
if err != nil {
t.Fatalf("Failed to retrieve active proposal: %v", err)
```

```
}
if len(prop.Pass) != 1 {
t.Fatalf("Proposal vote count mismatch: have %d, want 1", len(prop.Pass))
}
// Demote a signer and check release proposal deletion
for i := 0; i <= len(keys)/2; i++ {
if _, err := oracle.Demote(bind.NewKeyedTransactor(keys[i]),
crypto.PubkeyToAddress(keys[len(keys)-1].PublicKey)); err != nil {
t.Fatalf("Iter #%d: failed valid demotion attempt: %v", i, err)
}
}
sim.Commit()

prop, err = oracle.ProposedVersion(nil)
if err != nil {
t.Fatalf("Failed to retrieve active proposal: %v", err)
}
if len(prop.Pass) != 0 {
t.Fatalf("Proposal vote count mismatch: have %d, want 0", len(prop.Pass))
}
}

55:F:\git\coin\ethereum\go-ethereum\contracts\release\release.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package release contains the node service that tracks client releases.
package release

//go:generate abigen --sol ./contract.sol --pkg release --out ./contract.go

import (
"context"
"fmt"
"strings"
"time"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/eth"
"github.com/ethereum/go-ethereum/internal/ethapi"
"github.com/ethereum/go-ethereum/les"
"github.com/ethereum/go-ethereum/log"
```

```
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/p2p"
"github.com/ethereum/go-ethereum/rpc"
)

// Interval to check for new releases
const releaseRecheckInterval = time.Hour

// Config contains the configurations of the release service.
type Config struct {
Oracle common.Address // Ethereum address of the release oracle
Major  uint32         // Major version component of the release
Minor  uint32         // Minor version component of the release
Patch  uint32         // Patch version component of the release
Commit [20]byte       // Git SHA1 commit hash of the release
}

// ReleaseService is a node service that periodically checks the blockchain for
// newly released versions of the client being run and issues a warning to the
// user about it.
type ReleaseService struct {
config Config          // Current version to check releases against
oracle *ReleaseOracle  // Native binding to the release oracle contract
quit   chan chan error // Quit channel to terminate the version checker
}

// NewReleaseService creates a new service to periodically check for new client
// releases and notify the user of such.
func NewReleaseService(ctx *node.ServiceContext, config Config) (node.Service, error) {
// Retrieve the Ethereum service dependency to access the blockchain
var apiBackend ethapi.Backend
var ethereum *eth.Ethereum
if err := ctx.Service(&ethereum); err == nil {
apiBackend = ethereum.ApiBackend
} else {
var ethereum *les.LightEthereum
if err := ctx.Service(&ethereum); err == nil {
apiBackend = ethereum.ApiBackend
} else {
return nil, err
}
}
```

```
// Construct the release service
contract, err := NewReleaseOracle(config.Oracle, eth.NewContractBackend(apiBackend))
if err != nil {
return nil, err
}
return &ReleaseService{
config: config,
oracle: contract,
quit:   make(chan chan error),
}, nil
}

// Protocols returns an empty list of P2P protocols as the release service does
// not have a networking component.
func (r *ReleaseService) Protocols() []p2p.Protocol { return nil }

// APIs returns an empty list of RPC descriptors as the release service does not
// expose any functioanlity to the outside world.
func (r *ReleaseService) APIs() []rpc.API { return nil }

// Start spawns the periodic version checker goroutine
func (r *ReleaseService) Start(server *p2p.Server) error {
go r.checker()
return nil
}

// Stop terminates all goroutines belonging to the service, blocking until they
// are all terminated.
func (r *ReleaseService) Stop() error {
errc := make(chan error)
r.quit <- errc
return <-errc
}

// checker runs indefinitely in the background, periodically checking for new
// client releases.
func (r *ReleaseService) checker() {
// Set up the timers to periodically check for releases
timer := time.NewTimer(0) // Immediately fire a version check
defer timer.Stop()

for {
```

```
select {
case <-timer.C:
// Rechedule the timer before continuing
timer.Reset(releaseRecheckInterval)
r.checkVersion()
case errc := <-r.quit:
errc <- nil
return
}
}
}

func (r *ReleaseService) checkVersion() {
// Retrieve the current version, and handle missing contracts gracefully
ctx, cancel := context.WithTimeout(context.Background(), time.Second*5)
opts := &bind.CallOpts{Context: ctx}
defer cancel()

version, err := r.oracle.CurrentVersion(opts)
if err != nil {
if err == bind.ErrNoCode {
log.Debug("Release oracle not found", "contract", r.config.Oracle)
} else {
log.Error("Failed to retrieve current release", "err", err)
}
return
}
// Version was successfully retrieved, notify if newer than ours
if version.Major > r.config.Major ||
(version.Major == r.config.Major && version.Minor > r.config.Minor) ||
(version.Major == r.config.Major && version.Minor == r.config.Minor && version.Patch >
r.config.Patch) {

warning := fmt.Sprintf("Client v%d.%d.%d-%x seems older than the latest upstream release
v%d.%d.%d-%x",
r.config.Major, r.config.Minor, r.config.Patch, r.config.Commit[:4], version.Major, version.Minor,
version.Patch, version.Commit[:4])
howtofix := fmt.Sprintf("Please check https://github.com/ethereum/go-ethereum/releases for new
releases")
separator := strings.Repeat("-", len(warning))

log.Warn(separator)
```

```
log.Warn(warning)
log.Warn(howtofix)
log.Warn(separator)
} else {
log.Debug("Client seems up to date with upstream",
"local", fmt.Sprintf("v%d.%d.%d-%x", r.config.Major, r.config.Minor, r.config.Patch,
r.config.Commit[:4]),
"upstream", fmt.Sprintf("v%d.%d.%d-%x", version.Major, version.Minor, version.Patch,
version.Commit[:4]))
}
}
```

56:F:\git\coin\ethereum\go-ethereum\core\asm\asm.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Provides support for dealing with EVM assembly instructions (e.g., disassembling them).
package asm

import (
"encoding/hex"
"fmt"

"github.com/ethereum/go-ethereum/core/vm"
)

// Iterator for disassembled EVM instructions
type instructionIterator struct {
code    []byte
pc      uint64
arg     []byte
op      vm.OpCode
error   error
started bool
}

// Create a new instruction iterator.
func NewInstructionIterator(code []byte) *instructionIterator {
it := new(instructionIterator)
it.code = code
return it
}
```

```
// Returns true if there is a next instruction and moves on.
func (it *instructionIterator) Next() bool {
if it.error != nil || uint64(len(it.code)) <= it.pc {
// We previously reached an error or the end.
return false
}

if it.started {
// Since the iteration has been already started we move to the next instruction.
if it.arg != nil {
it.pc += uint64(len(it.arg))
}
it.pc++
} else {
// We start the iteration from the first instruction.
it.started = true
}

if uint64(len(it.code)) <= it.pc {
// We reached the end.
return false
}

it.op = vm.OpCode(it.code[it.pc])
if it.op.IsPush() {
a := uint64(it.op) - uint64(vm.PUSH1) + 1
u := it.pc + 1 + a
if uint64(len(it.code)) <= it.pc || uint64(len(it.code)) < u {
it.error = fmt.Errorf("incomplete push instruction at %v", it.pc)
return false
}
it.arg = it.code[it.pc+1 : u]
} else {
it.arg = nil
}
return true
}

// Returns any error that may have been encountered.
func (it *instructionIterator) Error() error {
return it.error
}
```

```
// Returns the PC of the current instruction.
func (it *instructionIterator) PC() uint64 {
return it.pc
}

// Returns the opcode of the current instruction.
func (it *instructionIterator) Op() vm.OpCode {
return it.op
}

// Returns the argument of the current instruction.
func (it *instructionIterator) Arg() []byte {
return it.arg
}

// Pretty-print all disassembled EVM instructions to stdout.
func PrintDisassembled(code string) error {
script, err := hex.DecodeString(code)
if err != nil {
return err
}

it := NewInstructionIterator(script)
for it.Next() {
if it.Arg() != nil && 0 < len(it.Arg()) {
fmt.Printf("%06v: %v 0x%x\n", it.PC(), it.Op(), it.Arg())
} else {
fmt.Printf("%06v: %v\n", it.PC(), it.Op())
}
}
if err := it.Error(); err != nil {
return err
}
return nil
}

// Return all disassembled EVM instructions in human-readable format.
func Disassemble(script []byte) ([]string, error) {
instrs := make([]string, 0)

it := NewInstructionIterator(script)
```

```
for it.Next() {
if it.Arg() != nil && 0 < len(it.Arg()) {
instrs = append(instrs, fmt.Sprintf("%06v: %v 0x%x\n", it.PC(), it.Op(), it.Arg()))
} else {
instrs = append(instrs, fmt.Sprintf("%06v: %v\n", it.PC(), it.Op()))
}
}
if err := it.Error(); err != nil {
return nil, err
}
return instrs, nil
}

57:F:\git\coin\ethereum\go-ethereum\core\asm\asm_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package asm

import (
"testing"

"encoding/hex"
)

// Tests disassembling the instructions for valid evm code
func TestInstructionIteratorValid(t *testing.T) {
cnt := 0
script, _ := hex.DecodeString("61000000")

it := NewInstructionIterator(script)
for it.Next() {
cnt++
}

if err := it.Error(); err != nil {
t.Errorf("Expected 2, but encountered error %v instead.", err)
}
if cnt != 2 {
t.Errorf("Expected 2, but got %v instead.", cnt)
}
}
```

```
// Tests disassembling the instructions for invalid evm code
func TestInstructionIteratorInvalid(t *testing.T) {
cnt := 0
script, _ := hex.DecodeString("6100")

it := NewInstructionIterator(script)
for it.Next() {
cnt++
}

if it.Error() == nil {
t.Errorf("Expected an error, but got %v instead.", cnt)
}
}

// Tests disassembling the instructions for empty evm code
func TestInstructionIteratorEmpty(t *testing.T) {
cnt := 0
script, _ := hex.DecodeString("")

it := NewInstructionIterator(script)
for it.Next() {
cnt++
}

if err := it.Error(); err != nil {
t.Errorf("Expected 0, but encountered error %v instead.", err)
}
if cnt != 0 {
t.Errorf("Expected 0, but got %v instead.", cnt)
}
}

58:F:\git\coin\ethereum\go-ethereum\core\asm\compiler.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package asm

import (
"errors"
"fmt"
"math/big"
```

```
"os"
"strings"

"github.com/ethereum/go-ethereum/common/math"
"github.com/ethereum/go-ethereum/core/vm"
)

// Compiler contains information about the parsed source
// and holds the tokens for the program.
type Compiler struct {
tokens []token
binary []interface{}

labels map[string]int

pc, pos int

debug bool
}

// newCompiler returns a new allocated compiler.
func NewCompiler(debug bool) *Compiler {
return &Compiler{
labels: make(map[string]int),
debug:  debug,
}
}

// Feed feeds tokens in to ch and are interpreted by
// the compiler.
//
// feed is the first pass in the compile stage as it
// collect the used labels in the program and keeps a
// program counter which is used to determine the locations
// of the jump dests. The labels can than be used in the
// second stage to push labels and determine the right
// position.
func (c *Compiler) Feed(ch <-chan token) {
for i := range ch {
switch i.typ {
case number:
num := math.MustParseBig256(i.text).Bytes()
```

```
if len(num) == 0 {
num = []byte{0}
}
c.pc += len(num)
case stringValue:
c.pc += len(i.text) - 2
case element:
c.pc++
case labelDef:
c.labels[i.text] = c.pc
c.pc++
case label:
c.pc += 5
}

c.tokens = append(c.tokens, i)
}
if c.debug {
fmt.Fprintln(os.Stderr, "found", len(c.labels), "labels")
}
}

// Compile compiles the current tokens and returns a
// binary string that can be interpreted by the EVM
// and an error if it failed.
//
// compile is the second stage in the compile phase
// which compiles the tokens to EVM instructions.
func (c *Compiler) Compile() (string, []error) {
var errors []error
// continue looping over the tokens until
// the stack has been exhausted.
for c.pos < len(c.tokens) {
if err := c.compileLine(); err != nil {
errors = append(errors, err)
}
}

// turn the binary to hex
var bin string
for _, v := range c.binary {
switch v := v.(type) {
```

```
case vm.OpCode:
bin += fmt.Sprintf("%x", []byte{byte(v)})
case []byte:
bin += fmt.Sprintf("%x", v)
}
}
return bin, errors
}

// next returns the next token and increments the
// posititon.
func (c *Compiler) next() token {
token := c.tokens[c.pos]
c.pos++
return token
}

// compile line compiles a single line instruction e.g.
// "push 1", "jump @labal".
func (c *Compiler) compileLine() error {
n := c.next()
if n.typ != lineStart {
return compileErr(n, n.typ.String(), lineStart.String())
}

lvalue := c.next()
switch lvalue.typ {
case eof:
return nil
case element:
if err := c.compileElement(lvalue); err != nil {
return err
}
case labelDef:
c.compileLabel()
case lineEnd:
return nil
default:
return compileErr(lvalue, lvalue.text, fmt.Sprintf("%v or %v", labelDef, element))
}

if n := c.next(); n.typ != lineEnd {
```

```
return compileErr(n, n.text, lineEnd.String())
}

return nil
}

// compileNumber compiles the number to bytes
func (c *Compiler) compileNumber(element token) (int, error) {
num := math.MustParseBig256(element.text).Bytes()
if len(num) == 0 {
num = []byte{0}
}
c.pushBin(num)
return len(num), nil
}

// compileElement compiles the element (push & label or both)
// to a binary representation and may error if incorrect statements
// where fed.
func (c *Compiler) compileElement(element token) error {
// check for a jump. jumps must be read and compiled
// from right to left.
if isJump(element.text) {
rvalue := c.next()
switch rvalue.typ {
case number:
// TODO figure out how to return the error properly
c.compileNumber(rvalue)
case stringValue:
// strings are quoted, remove them.
c.pushBin(rvalue.text[1 : len(rvalue.text)-2])
case label:
c.pushBin(vm.PUSH4)
pos := big.NewInt(int64(c.labels[rvalue.text])).Bytes()
pos = append(make([]byte, 4-len(pos)), pos...)
c.pushBin(pos)
default:
return compileErr(rvalue, rvalue.text, "number, string or label")
}
// push the operation
c.pushBin(toBinary(element.text))
return nil
```

```
} else if isPush(element.text) {
// handle pushes. pushes are read from left to right.
var value []byte

rvalue := c.next()
switch rvalue.typ {
case number:
value = math.MustParseBig256(rvalue.text).Bytes()
if len(value) == 0 {
value = []byte{0}
}
case stringValue:
value = []byte(rvalue.text[1 : len(rvalue.text)-1])
case label:
value = make([]byte, 4)
copy(value, big.NewInt(int64(c.labels[rvalue.text])).Bytes())
default:
return compileErr(rvalue, rvalue.text, "number, string or label")
}

if len(value) > 32 {
return fmt.Errorf("%d type error: unsupported string or number with size > 32", rvalue.lineno)
}

c.pushBin(vm.OpCode(int(vm.PUSH1) - 1 + len(value)))
c.pushBin(value)
} else {
c.pushBin(toBinary(element.text))
}

return nil
}

// compileLabel pushes a jumpdest to the binary slice.
func (c *Compiler) compileLabel() {
c.pushBin(vm.JUMPDEST)
}

// pushBin pushes the value v to the binary stack.
func (c *Compiler) pushBin(v interface{}) {
if c.debug {
fmt.Printf("%d: %v\n", len(c.binary), v)
```

```
}
c.binary = append(c.binary, v)
}

// isPush returns whether the string op is either any of
// push(N).
func isPush(op string) bool {
if op == "push" {
return true
}
return false
}

// isJump returns whether the string op is jump(i)
func isJump(op string) bool {
return op == "jumpi" || op == "jump"
}

// toBinary converts text to a vm.OpCode
func toBinary(text string) vm.OpCode {
if isPush(text) {
text = "push1"
}
return vm.StringToOp(strings.ToUpper(text))
}

type compileError struct {
got  string
want string

lineno int
}

func (err compileError) Error() string {
return fmt.Sprintf("%d syntax error: unexpected %v, expected %v", err.lineno, err.got, err.want)
}

var (
errExpBol           = errors.New("expected beginning of line")
errExpElementOrLabel = errors.New("expected beginning of line")
)
```

```
func compileErr(c token, got, want string) error {
return compileError{
got:    got,
want:   want,
lineno: c.lineno,
}
}
```

59:F:\git\coin\ethereum\go-ethereum\core\asm\lexer.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package asm

import (
"fmt"
"os"
"strings"
"unicode"
"unicode/utf8"
)

// stateFn is used through the lifetime of the
// lexer to parse the different values at the
// current state.
type stateFn func(*lexer) stateFn

// token is emitted when the lexer has discovered
// a new parsable token. These are delivered over
// the tokens channels of the lexer
type token struct {
typ    tokenType
lineno int
text   string
}

// tokenType are the different types the lexer
// is able to parse and return.
type tokenType int

const (
eof             tokenType = iota // end of file
lineStart                         // emitted when a line starts
```

```
lineEnd                 // emitted when a line ends
invalidStatement        // any invalid statement
element                 // any element during element parsing
label                   // label is emitted when a labal is found
labelDef                // label definition is emitted when a new label is found
number                  // number is emitted when a number is found
stringValue             // stringValue is emitted when a string has been found

Numbers           = "1234567890"                          // characters representing any decimal
number
HexadecimalNumbers = Numbers + "aAbBcCdDeEfF"             // characters
representing any hexadecimal
Alpha             = "abcdefghijklmnopqrstuwvxyzABCDEFGHIJKLMNOPQRSTUWVXYZ" //
characters representing alphanumeric
)

// String implements stringer
func (it tokenType) String() string {
if int(it) > len(stringtokenTypes) {
return "invalid"
}
return stringtokenTypes[it]
}

var stringtokenTypes = []string{
eof:             "EOF",
invalidStatement: "invalid statement",
element:         "element",
lineEnd:         "end of line",
lineStart:       "new line",
label:           "label",
labelDef:        "label definition",
number:          "number",
stringValue:     "string",
}

// lexer is the basic construct for parsing
// source code and turning them in to tokens.
// Tokens are interpreted by the compiler.
type lexer struct {
input string // input contains the source code of the program
```

```
tokens chan token // tokens is used to deliver tokens to the listener
state  stateFn    // the current state function

lineno           int // current line number in the source file
start, pos, width int // positions for lexing and returning value

debug bool // flag for triggering debug output
}

// lex lexes the program by name with the given source. It returns a
// channel on which the tokens are delivered.
func Lex(name string, source []byte, debug bool) <-chan token {
ch := make(chan token)
l := &lexer{
input:  string(source),
tokens: ch,
state:  lexLine,
debug:  debug,
}
go func() {
l.emit(lineStart)
for l.state != nil {
l.state = l.state(l)
}
l.emit(eof)
close(l.tokens)
}()

return ch
}

// next returns the next rune in the program's source.
func (l *lexer) next() (rune rune) {
if l.pos >= len(l.input) {
l.width = 0
return 0
}
rune, l.width = utf8.DecodeRuneInString(l.input[l.pos:])
l.pos += l.width
return rune
}
```

```
// backup backsup the last parsed element (multi-character)
func (l *lexer) backup() {
l.pos -= l.width
}

// peek returns the next rune but does not advance the seeker
func (l *lexer) peek() rune {
r := l.next()
l.backup()
return r
}

// ignore advances the seeker and ignores the value
func (l *lexer) ignore() {
l.start = l.pos
}

// Accepts checks whether the given input matches the next rune
func (l *lexer) accept(valid string) bool {
if strings.IndexRune(valid, l.next()) >= 0 {
return true
}

l.backup()

return false
}

// acceptRun will continue to advance the seeker until valid
// can no longer be met.
func (l *lexer) acceptRun(valid string) {
for strings.IndexRune(valid, l.next()) >= 0 {
}
l.backup()
}

// acceptRunUntil is the inverse of acceptRun and will continue
// to advance the seeker until the rune has been found.
func (l *lexer) acceptRunUntil(until rune) bool {
// Continues running until a rune is found
for i := l.next(); strings.IndexRune(string(until), i) == -1; i = l.next() {
if i == 0 {
```

```
return false
}
}

return true
}

// blob returns the current value
func (l *lexer) blob() string {
return l.input[l.start:l.pos]
}

// Emits a new token on to token channel for processing
func (l *lexer) emit(t tokenType) {
token := token{t, l.lineno, l.blob()}

if l.debug {
fmt.Fprintf(os.Stderr, "%04d: (%-20v) %s\n", token.lineno, token.typ, token.text)
}

l.tokens <- token
l.start = l.pos
}

// lexLine is state function for lexing lines
func lexLine(l *lexer) stateFn {
for {
switch r := l.next(); {
case r == '\n':
l.emit(lineEnd)
l.ignore()
l.lineno++

l.emit(lineStart)
case r == ';' && l.peek() == ';':
return lexComment
case isSpace(r):
l.ignore()
case isAlphaNumeric(r) || r == '_':
return lexElement
case isNumber(r):
return lexNumber
```

```
case r == '@':
l.ignore()
return lexLabel
case r == "":
return lexInsideString
default:
return nil
}
}
}

// lexComment parses the current position until the end
// of the line and discards the text.
func lexComment(l *lexer) stateFn {
l.acceptRunUntil('\n')
l.ignore()

return lexLine
}

// lexLabel parses the current label, emits and returns
// the lex text state function to advance the parsing
// process.
func lexLabel(l *lexer) stateFn {
l.acceptRun(Alpha + "_")

l.emit(label)

return lexLine
}

// lexInsideString lexes the inside of a string until
// until the state function finds the closing quote.
// It returns the lex text state function.
func lexInsideString(l *lexer) stateFn {
if l.acceptRunUntil("") {
l.emit(stringValue)
}

return lexLine
}
```

```
func lexNumber(l *lexer) stateFn {
acceptance := Numbers
if l.accept("0") && l.accept("xX") {
acceptance = HexadecimalNumbers
}
l.acceptRun(acceptance)

l.emit(number)

return lexLine
}

func lexElement(l *lexer) stateFn {
l.acceptRun(Alpha + "_" + Numbers)

if l.peek() == ':' {
l.emit(labelDef)

l.accept(":")
l.ignore()
} else {
l.emit(element)
}
return lexLine
}

func isAlphaNumeric(t rune) bool {
return unicode.IsLetter(t)
}

func isSpace(t rune) bool {
return unicode.IsSpace(t)
}

func isNumber(t rune) bool {
return unicode.IsNumber(t)
}

60:F:\git\coin\ethereum\go-ethereum\core\asm\lex_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package asm
```

```
import "testing"

func lexAll(src string) []token {
ch := Lex("test.asm", []byte(src), false)

var tokens []token
for i := range ch {
tokens = append(tokens, i)
}
return tokens
}

func TestComment(t *testing.T) {
tokens := lexAll(";; this is a comment")
if len(tokens) != 2 { // {new line, EOF}
t.Error("expected no tokens")
}
}

61:F:\git\coin\ethereum\go-ethereum\core\bench_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"crypto/ecdsa"
"io/ioutil"
"math/big"
"os"
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/math"
"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/params"
)
```

```
func BenchmarkInsertChain_empty_memdb(b *testing.B) {
benchInsertChain(b, false, nil)
}
func BenchmarkInsertChain_empty_diskdb(b *testing.B) {
benchInsertChain(b, true, nil)
}
func BenchmarkInsertChain_valueTx_memdb(b *testing.B) {
benchInsertChain(b, false, genValueTx(0))
}
func BenchmarkInsertChain_valueTx_diskdb(b *testing.B) {
benchInsertChain(b, true, genValueTx(0))
}
func BenchmarkInsertChain_valueTx_100kB_memdb(b *testing.B) {
benchInsertChain(b, false, genValueTx(100*1024))
}
func BenchmarkInsertChain_valueTx_100kB_diskdb(b *testing.B) {
benchInsertChain(b, true, genValueTx(100*1024))
}
func BenchmarkInsertChain_uncles_memdb(b *testing.B) {
benchInsertChain(b, false, genUncles)
}
func BenchmarkInsertChain_uncles_diskdb(b *testing.B) {
benchInsertChain(b, true, genUncles)
}
func BenchmarkInsertChain_ring200_memdb(b *testing.B) {
benchInsertChain(b, false, genTxRing(200))
}
func BenchmarkInsertChain_ring200_diskdb(b *testing.B) {
benchInsertChain(b, true, genTxRing(200))
}
func BenchmarkInsertChain_ring1000_memdb(b *testing.B) {
benchInsertChain(b, false, genTxRing(1000))
}
func BenchmarkInsertChain_ring1000_diskdb(b *testing.B) {
benchInsertChain(b, true, genTxRing(1000))
}

var (
// This is the content of the genesis block used by the benchmarks.
benchRootKey, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
```

```
291")
benchRootAddr   = crypto.PubkeyToAddress(benchRootKey.PublicKey)
benchRootFunds  = math.BigPow(2, 100)
)

// genValueTx returns a block generator that includes a single
// value-transfer transaction with n bytes of extra data in each
// block.
func genValueTx(nbytes int) func(int, *BlockGen) {
return func(i int, gen *BlockGen) {
toaddr := common.Address{}
data := make([]byte, nbytes)
gas := IntrinsicGas(data, false, false)
tx, _ := types.SignTx(types.NewTransaction(gen.TxNonce(benchRootAddr), toaddr, big.NewInt(1),
gas, nil, data), types.HomesteadSigner{}, benchRootKey)
gen.AddTx(tx)
}
}

var (
ringKeys  = make([]*ecdsa.PrivateKey, 1000)
ringAddrs = make([]common.Address, len(ringKeys))
bigTxGas  = new(big.Int).SetUint64(params.TxGas)
)

func init() {
ringKeys[0] = benchRootKey
ringAddrs[0] = benchRootAddr
for i := 1; i < len(ringKeys); i++ {
ringKeys[i], _ = crypto.GenerateKey()
ringAddrs[i] = crypto.PubkeyToAddress(ringKeys[i].PublicKey)
}
}

// genTxRing returns a block generator that sends ether in a ring
// among n accounts. This is creates n entries in the state database
// and fills the blocks with many small transactions.
func genTxRing(naccounts int) func(int, *BlockGen) {
from := 0
return func(i int, gen *BlockGen) {
gas := CalcGasLimit(gen.PrevBlock(i - 1))
for {
```

```
gas.Sub(gas, bigTxGas)
if gas.Cmp(bigTxGas) < 0 {
break
}
to := (from + 1) % naccounts
tx := types.NewTransaction(
gen.TxNonce(ringAddrs[from]),
ringAddrs[to],
benchRootFunds,
bigTxGas,
nil,
nil,
)
tx, _ = types.SignTx(tx, types.HomesteadSigner{}, ringKeys[from])
gen.AddTx(tx)
from = to
}
}
}

// genUncles generates blocks with two uncle headers.
func genUncles(i int, gen *BlockGen) {
if i >= 6 {
b2 := gen.PrevBlock(i - 6).Header()
b2.Extra = []byte("foo")
gen.AddUncle(b2)
b3 := gen.PrevBlock(i - 6).Header()
b3.Extra = []byte("bar")
gen.AddUncle(b3)
}
}

func benchInsertChain(b *testing.B, disk bool, gen func(int, *BlockGen)) {
// Create the database in memory or in a temporary directory.
var db ethdb.Database
if !disk {
db, _ = ethdb.NewMemDatabase()
} else {
dir, err := ioutil.TempDir("", "eth-core-bench")
if err != nil {
b.Fatalf("cannot create temporary directory: %v", err)
}
```

```
defer os.RemoveAll(dir)
db, err = ethdb.NewLDBDatabase(dir, 128, 128)
if err != nil {
b.Fatalf("cannot create temporary database: %v", err)
}
defer db.Close()
}

// Generate a chain of b.N blocks using the supplied block
// generator function.
gspec := Genesis{
Config: params.TestChainConfig,
Alloc:  GenesisAlloc{benchRootAddr: {Balance: benchRootFunds}},
}
genesis := gspec.MustCommit(db)
chain, _ := GenerateChain(gspec.Config, genesis, db, b.N, gen)

// Time the insertion of the new chain.
// State and blocks are stored in the same DB.
evmux := new(event.TypeMux)
chainman, _ := NewBlockChain(db, gspec.Config, ethash.NewFaker(), evmux, vm.Config{})
defer chainman.Stop()
b.ReportAllocs()
b.ResetTimer()
if i, err := chainman.InsertChain(chain); err != nil {
b.Fatalf("insert error (block %d): %v\n", i, err)
}
}

func BenchmarkChainRead_header_10k(b *testing.B) {
benchReadChain(b, false, 10000)
}
func BenchmarkChainRead_full_10k(b *testing.B) {
benchReadChain(b, true, 10000)
}
func BenchmarkChainRead_header_100k(b *testing.B) {
benchReadChain(b, false, 100000)
}
func BenchmarkChainRead_full_100k(b *testing.B) {
benchReadChain(b, true, 100000)
}
func BenchmarkChainRead_header_500k(b *testing.B) {
```

```
benchReadChain(b, false, 500000)
}
func BenchmarkChainRead_full_500k(b *testing.B) {
benchReadChain(b, true, 500000)
}
func BenchmarkChainWrite_header_10k(b *testing.B) {
benchWriteChain(b, false, 10000)
}
func BenchmarkChainWrite_full_10k(b *testing.B) {
benchWriteChain(b, true, 10000)
}
func BenchmarkChainWrite_header_100k(b *testing.B) {
benchWriteChain(b, false, 100000)
}
func BenchmarkChainWrite_full_100k(b *testing.B) {
benchWriteChain(b, true, 100000)
}
func BenchmarkChainWrite_header_500k(b *testing.B) {
benchWriteChain(b, false, 500000)
}
func BenchmarkChainWrite_full_500k(b *testing.B) {
benchWriteChain(b, true, 500000)
}

// makeChainForBench writes a given number of headers or empty blocks/receipts
// into a database.
func makeChainForBench(db ethdb.Database, full bool, count uint64) {
var hash common.Hash
for n := uint64(0); n < count; n++ {
header := &types.Header{
Coinbase:    common.Address{},
Number:      big.NewInt(int64(n)),
ParentHash:  hash,
Difficulty:  big.NewInt(1),
UncleHash:   types.EmptyUncleHash,
TxHash:      types.EmptyRootHash,
ReceiptHash: types.EmptyRootHash,
}
hash = header.Hash()
WriteHeader(db, header)
WriteCanonicalHash(db, hash, n)
WriteTd(db, hash, n, big.NewInt(int64(n+1)))
```

```
if full || n == 0 {
block := types.NewBlockWithHeader(header)
WriteBody(db, hash, n, block.Body())
WriteBlockReceipts(db, hash, n, nil)
}
}
}

func benchWriteChain(b *testing.B, full bool, count uint64) {
for i := 0; i < b.N; i++ {
dir, err := ioutil.TempDir("", "eth-chain-bench")
if err != nil {
b.Fatalf("cannot create temporary directory: %v", err)
}
db, err := ethdb.NewLDBDatabase(dir, 128, 1024)
if err != nil {
b.Fatalf("error opening database at %v: %v", dir, err)
}
makeChainForBench(db, full, count)
db.Close()
os.RemoveAll(dir)
}
}

func benchReadChain(b *testing.B, full bool, count uint64) {
dir, err := ioutil.TempDir("", "eth-chain-bench")
if err != nil {
b.Fatalf("cannot create temporary directory: %v", err)
}
defer os.RemoveAll(dir)

db, err := ethdb.NewLDBDatabase(dir, 128, 1024)
if err != nil {
b.Fatalf("error opening database at %v: %v", dir, err)
}
makeChainForBench(db, full, count)
db.Close()

b.ReportAllocs()
b.ResetTimer()

for i := 0; i < b.N; i++ {
```

```
db, err := ethdb.NewLDBDatabase(dir, 128, 1024)
if err != nil {
b.Fatalf("error opening database at %v: %v", dir, err)
}
chain, err := NewBlockChain(db, params.TestChainConfig, ethash.NewFaker(),
new(event.TypeMux), vm.Config{})
if err != nil {
b.Fatalf("error creating chain: %v", err)
}

for n := uint64(0); n < count; n++ {
header := chain.GetHeaderByNumber(n)
if full {
hash := header.Hash()
GetBody(db, hash, n)
GetBlockReceipts(db, hash, n)
}
}

db.Close()
}
}

62:F:\git\coin\ethereum\go-ethereum\core\blockchain.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package core implements the Ethereum consensus protocol.
package core

import (
"errors"
"fmt"
"io"
"math/big"
mrand "math/rand"
"runtime"
"sync"
"sync/atomic"
"time"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/mclock"
```

```
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/metrics"
"github.com/ethereum/go-ethereum/params"
"github.com/ethereum/go-ethereum/rlp"
"github.com/ethereum/go-ethereum/trie"
"github.com/hashicorp/golang-lru"
)

var (
blockInsertTimer = metrics.NewTimer("chain/inserts")

ErrNoGenesis = errors.New("Genesis not found in chain")
)

const (
bodyCacheLimit      = 256
blockCacheLimit     = 256
maxFutureBlocks     = 256
maxTimeFutureBlocks = 30
badBlockLimit       = 10

// BlockChainVersion ensures that an incompatible database forces a resync from scratch.
BlockChainVersion = 3
)

// BlockChain represents the canonical chain given a database with a genesis
// block. The Blockchain manages chain imports, reverts, chain reorganisations.
//
// Importing blocks in to the block chain happens according to the set of rules
// defined by the two stage Validator. Processing of blocks is done using the
// Processor which processes the included transaction. The validation of the state
// is done in the second part of the Validator. Failing results in aborting of
// the import.
//
// The BlockChain also helps in returning blocks from **any** chain included
```

```
// in the database as well as blocks that represents the canonical chain. It's
// important to note that GetBlock can return any block and does not need to be
// included in the canonical one where as GetBlockByNumber always represents the
// canonical chain.
type BlockChain struct {
config *params.ChainConfig // chain & network configuration

hc           *HeaderChain
chainDb      ethdb.Database
eventMux     *event.TypeMux
genesisBlock *types.Block

mu      sync.RWMutex // global mutex for locking chain operations
chainmu sync.RWMutex // blockchain insertion lock
procmu  sync.RWMutex // block processor lock

checkpoint       int          // checkpoint counts towards the new checkpoint
currentBlock     *types.Block // Current head of the block chain
currentFastBlock *types.Block // Current head of the fast-sync chain (may be above the block
chain!)

stateCache   state.Database // State database to reuse between imports (contains state cache)
bodyCache    *lru.Cache     // Cache for the most recent block bodies
bodyRLPCache *lru.Cache     // Cache for the most recent block bodies in RLP encoded format
blockCache   *lru.Cache     // Cache for the most recent entire blocks
futureBlocks *lru.Cache     // future blocks are blocks added for later processing

quit    chan struct{} // blockchain quit channel
running int32         // running must be called atomically
// procInterrupt must be atomically called
procInterrupt int32          // interrupt signaler for block processing
wg            sync.WaitGroup // chain processing wait group for shutting down

engine    consensus.Engine
processor Processor // block processor interface
validator Validator // block and state validator interface
vmConfig  vm.Config

badBlocks *lru.Cache // Bad block cache
}

// NewBlockChain returns a fully initialised block chain using information
```

```
// available in the database. It initialises the default Ethereum Validator and
// Processor.
func NewBlockChain(chainDb ethdb.Database, config *params.ChainConfig, engine
consensus.Engine, mux *event.TypeMux, vmConfig vm.Config) (*BlockChain, error) {
bodyCache, _ := lru.New(bodyCacheLimit)
bodyRLPCache, _ := lru.New(bodyCacheLimit)
blockCache, _ := lru.New(blockCacheLimit)
futureBlocks, _ := lru.New(maxFutureBlocks)
badBlocks, _ := lru.New(badBlockLimit)

bc := &BlockChain{
config:       config,
chainDb:      chainDb,
stateCache:   state.NewDatabase(chainDb),
eventMux:     mux,
quit:         make(chan struct{}),
bodyCache:    bodyCache,
bodyRLPCache: bodyRLPCache,
blockCache:   blockCache,
futureBlocks: futureBlocks,
engine:       engine,
vmConfig:     vmConfig,
badBlocks:    badBlocks,
}
bc.SetValidator(NewBlockValidator(config, bc, engine))
bc.SetProcessor(NewStateProcessor(config, bc, engine))

var err error
bc.hc, err = NewHeaderChain(chainDb, config, engine, bc.getProcInterrupt)
if err != nil {
return nil, err
}
bc.genesisBlock = bc.GetBlockByNumber(0)
if bc.genesisBlock == nil {
return nil, ErrNoGenesis
}
if err := bc.loadLastState(); err != nil {
return nil, err
}
// Check the current state of the block hashes and make sure that we do not have any of the bad
blocks in our chain
for hash := range BadHashes {
```

```
if header := bc.GetHeaderByHash(hash); header != nil {
// get the canonical block corresponding to the offending header's number
headerByNumber := bc.GetHeaderByNumber(header.Number.Uint64())
// make sure the headerByNumber (if present) is in our current canonical chain
if headerByNumber != nil && headerByNumber.Hash() == header.Hash() {
log.Error("Found bad hash, rewinding chain", "number", header.Number, "hash",
header.ParentHash)
bc.SetHead(header.Number.Uint64() - 1)
log.Error("Chain rewind was successful, resuming normal operation")
}
}
}
// Take ownership of this particular state
go bc.update()
return bc, nil
}

func (bc *BlockChain) getProcInterrupt() bool {
return atomic.LoadInt32(&bc.procInterrupt) == 1
}

// loadLastState loads the last known chain state from the database. This method
// assumes that the chain manager mutex is held.
func (bc *BlockChain) loadLastState() error {
// Restore the last known head block
head := GetHeadBlockHash(bc.chainDb)
if head == (common.Hash{}) {
// Corrupt or empty database, init from scratch
log.Warn("Empty database, resetting chain")
return bc.Reset()
}
// Make sure the entire head block is available
currentBlock := bc.GetBlockByHash(head)
if currentBlock == nil {
// Corrupt or empty database, init from scratch
log.Warn("Head block missing, resetting chain", "hash", head)
return bc.Reset()
}
// Make sure the state associated with the block is available
if _, err := state.New(currentBlock.Root(), bc.stateCache); err != nil {
// Dangling block without a state associated, init from scratch
log.Warn("Head state missing, resetting chain", "number", currentBlock.Number(), "hash",
```

```
currentBlock.Hash())
return bc.Reset()
}
// Everything seems to be fine, set as the head block
bc.currentBlock = currentBlock

// Restore the last known head header
currentHeader := bc.currentBlock.Header()
if head := GetHeadHeaderHash(bc.chainDb); head != (common.Hash{}) {
if header := bc.GetHeaderByHash(head); header != nil {
currentHeader = header
}
}
bc.hc.SetCurrentHeader(currentHeader)

// Restore the last known head fast block
bc.currentFastBlock = bc.currentBlock
if head := GetHeadFastBlockHash(bc.chainDb); head != (common.Hash{}) {
if block := bc.GetBlockByHash(head); block != nil {
bc.currentFastBlock = block
}
}

// Issue a status log for the user
headerTd := bc.GetTd(currentHeader.Hash(), currentHeader.Number.Uint64())
blockTd := bc.GetTd(bc.currentBlock.Hash(), bc.currentBlock.NumberU64())
fastTd := bc.GetTd(bc.currentFastBlock.Hash(), bc.currentFastBlock.NumberU64())

log.Info("Loaded most recent local header", "number", currentHeader.Number, "hash",
currentHeader.Hash(), "td", headerTd)
log.Info("Loaded most recent local full block", "number", bc.currentBlock.Number(), "hash",
bc.currentBlock.Hash(), "td", blockTd)
log.Info("Loaded most recent local fast block", "number", bc.currentFastBlock.Number(), "hash",
bc.currentFastBlock.Hash(), "td", fastTd)

return nil
}

// SetHead rewinds the local chain to a new head. In the case of headers, everything
// above the new head will be deleted and the new one set. In the case of blocks
// though, the head may be further rewound if block bodies are missing (non-archive
// nodes after a fast sync).
```

```
func (bc *BlockChain) SetHead(head uint64) error {
log.Warn("Rewinding blockchain", "target", head)

bc.mu.Lock()
defer bc.mu.Unlock()

// Rewind the header chain, deleting all block bodies until then
delFn := func(hash common.Hash, num uint64) {
DeleteBody(bc.chainDb, hash, num)
}
bc.hc.SetHead(head, delFn)
currentHeader := bc.hc.CurrentHeader()

// Clear out any stale content from the caches
bc.bodyCache.Purge()
bc.bodyRLPCache.Purge()
bc.blockCache.Purge()
bc.futureBlocks.Purge()

// Rewind the block chain, ensuring we don't end up with a stateless head block
if bc.currentBlock != nil && currentHeader.Number.Uint64() < bc.currentBlock.NumberU64() {
bc.currentBlock = bc.GetBlock(currentHeader.Hash(), currentHeader.Number.Uint64())
}
if bc.currentBlock != nil {
if _, err := state.New(bc.currentBlock.Root(), bc.stateCache); err != nil {
// Rewound state missing, rolled back to before pivot, reset to genesis
bc.currentBlock = nil
}
}
// Rewind the fast block in a simpleton way to the target head
if bc.currentFastBlock != nil && currentHeader.Number.Uint64() <
bc.currentFastBlock.NumberU64() {
bc.currentFastBlock = bc.GetBlock(currentHeader.Hash(), currentHeader.Number.Uint64())
}
// If either blocks reached nil, reset to the genesis state
if bc.currentBlock == nil {
bc.currentBlock = bc.genesisBlock
}
if bc.currentFastBlock == nil {
bc.currentFastBlock = bc.genesisBlock
}
if err := WriteHeadBlockHash(bc.chainDb, bc.currentBlock.Hash()); err != nil {
```

```
log.Crit("Failed to reset head full block", "err", err)
}
if err := WriteHeadFastBlockHash(bc.chainDb, bc.currentFastBlock.Hash()); err != nil {
log.Crit("Failed to reset head fast block", "err", err)
}
return bc.loadLastState()
}

// FastSyncCommitHead sets the current head block to the one defined by the hash
// irrelevant what the chain contents were prior.
func (bc *BlockChain) FastSyncCommitHead(hash common.Hash) error {
// Make sure that both the block as well at its state trie exists
block := bc.GetBlockByHash(hash)
if block == nil {
return fmt.Errorf("non existent block [%x…]", hash[:4])
}
if _, err := trie.NewSecure(block.Root(), bc.chainDb, 0); err != nil {
return err
}
// If all checks out, manually set the head block
bc.mu.Lock()
bc.currentBlock = block
bc.mu.Unlock()

log.Info("Committed new head block", "number", block.Number(), "hash", hash)
return nil
}

// GasLimit returns the gas limit of the current HEAD block.
func (bc *BlockChain) GasLimit() *big.Int {
bc.mu.RLock()
defer bc.mu.RUnlock()

return bc.currentBlock.GasLimit()
}

// LastBlockHash return the hash of the HEAD block.
func (bc *BlockChain) LastBlockHash() common.Hash {
bc.mu.RLock()
defer bc.mu.RUnlock()

return bc.currentBlock.Hash()
```

```
}

// CurrentBlock retrieves the current head block of the canonical chain. The
// block is retrieved from the blockchain's internal cache.
func (bc *BlockChain) CurrentBlock() *types.Block {
bc.mu.RLock()
defer bc.mu.RUnlock()

return bc.currentBlock
}

// CurrentFastBlock retrieves the current fast-sync head block of the canonical
// chain. The block is retrieved from the blockchain's internal cache.
func (bc *BlockChain) CurrentFastBlock() *types.Block {
bc.mu.RLock()
defer bc.mu.RUnlock()

return bc.currentFastBlock
}

// Status returns status information about the current chain such as the HEAD Td,
// the HEAD hash and the hash of the genesis block.
func (bc *BlockChain) Status() (td *big.Int, currentBlock common.Hash, genesisBlock
common.Hash) {
bc.mu.RLock()
defer bc.mu.RUnlock()

return bc.GetTd(bc.currentBlock.Hash(), bc.currentBlock.NumberU64()), bc.currentBlock.Hash(),
bc.genesisBlock.Hash()
}

// SetProcessor sets the processor required for making state modifications.
func (bc *BlockChain) SetProcessor(processor Processor) {
bc.procmu.Lock()
defer bc.procmu.Unlock()
bc.processor = processor
}

// SetValidator sets the validator which is used to validate incoming blocks.
func (bc *BlockChain) SetValidator(validator Validator) {
bc.procmu.Lock()
defer bc.procmu.Unlock()
```

```
bc.validator = validator
}

// Validator returns the current validator.
func (bc *BlockChain) Validator() Validator {
bc.procmu.RLock()
defer bc.procmu.RUnlock()
return bc.validator
}

// Processor returns the current processor.
func (bc *BlockChain) Processor() Processor {
bc.procmu.RLock()
defer bc.procmu.RUnlock()
return bc.processor
}

// State returns a new mutable state based on the current HEAD block.
func (bc *BlockChain) State() (*state.StateDB, error) {
return bc.StateAt(bc.CurrentBlock().Root())
}

// StateAt returns a new mutable state based on a particular point in time.
func (bc *BlockChain) StateAt(root common.Hash) (*state.StateDB, error) {
return state.New(root, bc.stateCache)
}

// Reset purges the entire blockchain, restoring it to its genesis state.
func (bc *BlockChain) Reset() error {
return bc.ResetWithGenesisBlock(bc.genesisBlock)
}

// ResetWithGenesisBlock purges the entire blockchain, restoring it to the
// specified genesis state.
func (bc *BlockChain) ResetWithGenesisBlock(genesis *types.Block) error {
// Dump the entire block chain and purge the caches
if err := bc.SetHead(0); err != nil {
return err
}
bc.mu.Lock()
defer bc.mu.Unlock()
```

```
// Prepare the genesis block and reinitialise the chain
if err := bc.hc.WriteTd(genesis.Hash(), genesis.NumberU64(), genesis.Difficulty()); err != nil {
log.Crit("Failed to write genesis block TD", "err", err)
}
if err := WriteBlock(bc.chainDb, genesis); err != nil {
log.Crit("Failed to write genesis block", "err", err)
}
bc.genesisBlock = genesis
bc.insert(bc.genesisBlock)
bc.currentBlock = bc.genesisBlock
bc.hc.SetGenesis(bc.genesisBlock.Header())
bc.hc.SetCurrentHeader(bc.genesisBlock.Header())
bc.currentFastBlock = bc.genesisBlock

return nil
}

// Export writes the active chain to the given writer.
func (bc *BlockChain) Export(w io.Writer) error {
return bc.ExportN(w, uint64(0), bc.currentBlock.NumberU64())
}

// ExportN writes a subset of the active chain to the given writer.
func (bc *BlockChain) ExportN(w io.Writer, first uint64, last uint64) error {
bc.mu.RLock()
defer bc.mu.RUnlock()

if first > last {
return fmt.Errorf("export failed: first (%d) is greater than last (%d)", first, last)
}
log.Info("Exporting batch of blocks", "count", last-first+1)

for nr := first; nr <= last; nr++ {
block := bc.GetBlockByNumber(nr)
if block == nil {
return fmt.Errorf("export failed on #%d: not found", nr)
}

if err := block.EncodeRLP(w); err != nil {
return err
}
}
```

```
return nil
}

// insert injects a new head block into the current block chain. This method
// assumes that the block is indeed a true head. It will also reset the head
// header and the head fast sync block to this very same block if they are older
// or if they are on a different side chain.
//
// Note, this function assumes that the `mu` mutex is held!
func (bc *BlockChain) insert(block *types.Block) {
// If the block is on a side chain or an unknown one, force other heads onto it too
updateHeads := GetCanonicalHash(bc.chainDb, block.NumberU64()) != block.Hash()

// Add the block to the canonical chain number scheme and mark as the head
if err := WriteCanonicalHash(bc.chainDb, block.Hash(), block.NumberU64()); err != nil {
log.Crit("Failed to insert block number", "err", err)
}
if err := WriteHeadBlockHash(bc.chainDb, block.Hash()); err != nil {
log.Crit("Failed to insert head block hash", "err", err)
}
bc.currentBlock = block

// If the block is better than out head or is on a different chain, force update heads
if updateHeads {
bc.hc.SetCurrentHeader(block.Header())

if err := WriteHeadFastBlockHash(bc.chainDb, block.Hash()); err != nil {
log.Crit("Failed to insert head fast block hash", "err", err)
}
bc.currentFastBlock = block
}
}

// Genesis retrieves the chain's genesis block.
func (bc *BlockChain) Genesis() *types.Block {
return bc.genesisBlock
}

// GetBody retrieves a block body (transactions and uncles) from the database by
// hash, caching it if found.
func (bc *BlockChain) GetBody(hash common.Hash) *types.Body {
```

```
// Short circuit if the body's already in the cache, retrieve otherwise
if cached, ok := bc.bodyCache.Get(hash); ok {
body := cached.(*types.Body)
return body
}
body := GetBody(bc.chainDb, hash, bc.hc.GetBlockNumber(hash))
if body == nil {
return nil
}
// Cache the found body for next time and return
bc.bodyCache.Add(hash, body)
return body
}

// GetBodyRLP retrieves a block body in RLP encoding from the database by hash,
// caching it if found.
func (bc *BlockChain) GetBodyRLP(hash common.Hash) rlp.RawValue {
// Short circuit if the body's already in the cache, retrieve otherwise
if cached, ok := bc.bodyRLPCache.Get(hash); ok {
return cached.(rlp.RawValue)
}
body := GetBodyRLP(bc.chainDb, hash, bc.hc.GetBlockNumber(hash))
if len(body) == 0 {
return nil
}
// Cache the found body for next time and return
bc.bodyRLPCache.Add(hash, body)
return body
}

// HasBlock checks if a block is fully present in the database or not, caching
// it if present.
func (bc *BlockChain) HasBlock(hash common.Hash) bool {
return bc.GetBlockByHash(hash) != nil
}

// HasBlockAndState checks if a block and associated state trie is fully present
// in the database or not, caching it if present.
func (bc *BlockChain) HasBlockAndState(hash common.Hash) bool {
// Check first that the block itself is known
block := bc.GetBlockByHash(hash)
if block == nil {
```

```
return false
}
// Ensure the associated state is also present
_, err := bc.stateCache.OpenTrie(block.Root())
return err == nil
}

// GetBlock retrieves a block from the database by hash and number,
// caching it if found.
func (bc *BlockChain) GetBlock(hash common.Hash, number uint64) *types.Block {
// Short circuit if the block's already in the cache, retrieve otherwise
if block, ok := bc.blockCache.Get(hash); ok {
return block.(*types.Block)
}
block := GetBlock(bc.chainDb, hash, number)
if block == nil {
return nil
}
// Cache the found block for next time and return
bc.blockCache.Add(block.Hash(), block)
return block
}

// GetBlockByHash retrieves a block from the database by hash, caching it if found.
func (bc *BlockChain) GetBlockByHash(hash common.Hash) *types.Block {
return bc.GetBlock(hash, bc.hc.GetBlockNumber(hash))
}

// GetBlockByNumber retrieves a block from the database by number, caching it
// (associated with its hash) if found.
func (bc *BlockChain) GetBlockByNumber(number uint64) *types.Block {
hash := GetCanonicalHash(bc.chainDb, number)
if hash == (common.Hash{}) {
return nil
}
return bc.GetBlock(hash, number)
}

// GetBlocksFromHash returns the block corresponding to hash and up to n-1 ancestors.
// [deprecated by eth/62]
func (bc *BlockChain) GetBlocksFromHash(hash common.Hash, n int) (blocks []*types.Block) {
number := bc.hc.GetBlockNumber(hash)
```

```
for i := 0; i < n; i++ {
block := bc.GetBlock(hash, number)
if block == nil {
break
}
blocks = append(blocks, block)
hash = block.ParentHash()
number--
}
return
}

// GetUnclesInChain retrieves all the uncles from a given block backwards until
// a specific distance is reached.
func (bc *BlockChain) GetUnclesInChain(block *types.Block, length int) []*types.Header {
uncles := []*types.Header{}
for i := 0; block != nil && i < length; i++ {
uncles = append(uncles, block.Uncles()...)
block = bc.GetBlock(block.ParentHash(), block.NumberU64()-1)
}
return uncles
}

// Stop stops the blockchain service. If any imports are currently in progress
// it will abort them using the procInterrupt.
func (bc *BlockChain) Stop() {
if !atomic.CompareAndSwapInt32(&bc.running, 0, 1) {
return
}
close(bc.quit)
atomic.StoreInt32(&bc.procInterrupt, 1)

bc.wg.Wait()
log.Info("Blockchain manager stopped")
}

func (bc *BlockChain) procFutureBlocks() {
blocks := make([]*types.Block, 0, bc.futureBlocks.Len())
for _, hash := range bc.futureBlocks.Keys() {
if block, exist := bc.futureBlocks.Peek(hash); exist {
blocks = append(blocks, block.(*types.Block))
}
```

```
}
if len(blocks) > 0 {
types.BlockBy(types.Number).Sort(blocks)

// Insert one by one as chain insertion needs contiguous ancestry between blocks
for i := range blocks {
bc.InsertChain(blocks[i : i+1])
}
}
}

// WriteStatus status of write
type WriteStatus byte

const (
NonStatTy WriteStatus = iota
CanonStatTy
SideStatTy
)

// Rollback is designed to remove a chain of links from the database that aren't
// certain enough to be valid.
func (bc *BlockChain) Rollback(chain []common.Hash) {
bc.mu.Lock()
defer bc.mu.Unlock()

for i := len(chain) - 1; i >= 0; i-- {
hash := chain[i]

currentHeader := bc.hc.CurrentHeader()
if currentHeader.Hash() == hash {
bc.hc.SetCurrentHeader(bc.GetHeader(currentHeader.ParentHash,
currentHeader.Number.Uint64()-1))
}
if bc.currentFastBlock.Hash() == hash {
bc.currentFastBlock = bc.GetBlock(bc.currentFastBlock.ParentHash(),
bc.currentFastBlock.NumberU64()-1)
WriteHeadFastBlockHash(bc.chainDb, bc.currentFastBlock.Hash())
}
if bc.currentBlock.Hash() == hash {
bc.currentBlock = bc.GetBlock(bc.currentBlock.ParentHash(), bc.currentBlock.NumberU64()-1)
WriteHeadBlockHash(bc.chainDb, bc.currentBlock.Hash())
```

```
}
}
}

// SetReceiptsData computes all the non-consensus fields of the receipts
func SetReceiptsData(config *params.ChainConfig, block *types.Block, receipts types.Receipts) {
signer := types.MakeSigner(config, block.Number())

transactions, logIndex := block.Transactions(), uint(0)

for j := 0; j < len(receipts); j++ {
// The transaction hash can be retrieved from the transaction itself
receipts[j].TxHash = transactions[j].Hash()

// The contract address can be derived from the transaction itself
if transactions[j].To() == nil {
// Deriving the signer is expensive, only do if it's actually needed
from, _ := types.Sender(signer, transactions[j])
receipts[j].ContractAddress = crypto.CreateAddress(from, transactions[j].Nonce())
}
// The used gas can be calculated based on previous receipts
if j == 0 {
receipts[j].GasUsed = new(big.Int).Set(receipts[j].CumulativeGasUsed)
} else {
receipts[j].GasUsed = new(big.Int).Sub(receipts[j].CumulativeGasUsed, receipts[j-
1].CumulativeGasUsed)
}
// The derived log fields can simply be set from the block and transaction
for k := 0; k < len(receipts[j].Logs); k++ {
receipts[j].Logs[k].BlockNumber = block.NumberU64()
receipts[j].Logs[k].BlockHash = block.Hash()
receipts[j].Logs[k].TxHash = receipts[j].TxHash
receipts[j].Logs[k].TxIndex = uint(j)
receipts[j].Logs[k].Index = logIndex
logIndex++
}
}
}

// InsertReceiptChain attempts to complete an already existing header chain with
// transaction and receipt data.
// XXX should this be moved to the test?
```

```
func (bc *BlockChain) InsertReceiptChain(blockChain types.Blocks, receiptChain []types.Receipts)
(int, error) {
// Do a sanity check that the provided chain is actually ordered and linked
for i := 1; i < len(blockChain); i++ {
if blockChain[i].NumberU64() != blockChain[i-1].NumberU64()+1 || blockChain[i].ParentHash() !=
blockChain[i-1].Hash() {
// Chain broke ancestry, log a messge (programming error) and skip insertion
log.Error("Non contiguous receipt insert", "number", blockChain[i].Number(), "hash",
blockChain[i].Hash(), "parent", blockChain[i].ParentHash(),
"prevnumber", blockChain[i-1].Number(), "prevhash", blockChain[i-1].Hash())

return 0, fmt.Errorf("non contiguous insert: item %d is #%d [%x…], item %d is #%d [%x…] (parent
[%x…])", i-1, blockChain[i-1].NumberU64(),
blockChain[i-1].Hash().Bytes()[:4], i, blockChain[i].NumberU64(), blockChain[i].Hash().Bytes()[:4],
blockChain[i].ParentHash().Bytes()[:4])
}
}
// Pre-checks passed, start the block body and receipt imports
bc.wg.Add(1)
defer bc.wg.Done()

// Collect some import statistics to report on
stats := struct{ processed, ignored int32 }{}
start := time.Now()

// Create the block importing task queue and worker functions
tasks := make(chan int, len(blockChain))
for i := 0; i < len(blockChain) && i < len(receiptChain); i++ {
tasks <- i
}
close(tasks)

errs, failed := make([]error, len(tasks)), int32(0)
process := func(worker int) {
for index := range tasks {
block, receipts := blockChain[index], receiptChain[index]

// Short circuit insertion if shutting down or processing failed
if atomic.LoadInt32(&bc.procInterrupt) == 1 {
return
}
if atomic.LoadInt32(&failed) > 0 {
```

```
return
}
// Short circuit if the owner header is unknown
if !bc.HasHeader(block.Hash()) {
errs[index] = fmt.Errorf("containing header #%d [%x…] unknown", block.Number(),
block.Hash().Bytes()[:4])
atomic.AddInt32(&failed, 1)
return
}
// Skip if the entire data is already known
if bc.HasBlock(block.Hash()) {
atomic.AddInt32(&stats.ignored, 1)
continue
}
// Compute all the non-consensus fields of the receipts
SetReceiptsData(bc.config, block, receipts)
// Write all the data out into the database
if err := WriteBody(bc.chainDb, block.Hash(), block.NumberU64(), block.Body()); err != nil {
errs[index] = fmt.Errorf("failed to write block body: %v", err)
atomic.AddInt32(&failed, 1)
log.Crit("Failed to write block body", "err", err)
return
}
if err := WriteBlockReceipts(bc.chainDb, block.Hash(), block.NumberU64(), receipts); err != nil {
errs[index] = fmt.Errorf("failed to write block receipts: %v", err)
atomic.AddInt32(&failed, 1)
log.Crit("Failed to write block receipts", "err", err)
return
}
if err := WriteMipmapBloom(bc.chainDb, block.NumberU64(), receipts); err != nil {
errs[index] = fmt.Errorf("failed to write log blooms: %v", err)
atomic.AddInt32(&failed, 1)
log.Crit("Failed to write log blooms", "err", err)
return
}
if err := WriteTransactions(bc.chainDb, block); err != nil {
errs[index] = fmt.Errorf("failed to write individual transactions: %v", err)
atomic.AddInt32(&failed, 1)
log.Crit("Failed to write individual transactions", "err", err)
return
}
if err := WriteReceipts(bc.chainDb, receipts); err != nil {
```

```
errs[index] = fmt.Errorf("failed to write individual receipts: %v", err)
atomic.AddInt32(&failed, 1)
log.Crit("Failed to write individual receipts", "err", err)
return
}
atomic.AddInt32(&stats.processed, 1)
}
}
// Start as many worker threads as goroutines allowed
pending := new(sync.WaitGroup)
for i := 0; i < runtime.GOMAXPROCS(0); i++ {
pending.Add(1)
go func(id int) {
defer pending.Done()
process(id)
}(i)
}
pending.Wait()

// If anything failed, report
if failed > 0 {
for i, err := range errs {
if err != nil {
return i, err
}
}
}
if atomic.LoadInt32(&bc.procInterrupt) == 1 {
log.Debug("Premature abort during receipts processing")
return 0, nil
}
// Update the head fast sync block if better
bc.mu.Lock()

head := blockChain[len(errs)-1]
if td := bc.GetTd(head.Hash(), head.NumberU64()); td != nil { // Rewind may have occurred, skip in that case
if bc.GetTd(bc.currentFastBlock.Hash(), bc.currentFastBlock.NumberU64()).Cmp(td) < 0 {
if err := WriteHeadFastBlockHash(bc.chainDb, head.Hash()); err != nil {
log.Crit("Failed to update head fast block hash", "err", err)
}
bc.currentFastBlock = head
```

```
}
}
bc.mu.Unlock()

// Report some public statistics so the user has a clue what's going on
last := blockChain[len(blockChain)-1]
log.Info("Imported new block receipts", "count", stats.processed, "elapsed",
common.PrettyDuration(time.Since(start)),
"number", last.Number(), "hash", last.Hash(), "ignored", stats.ignored)

return 0, nil
}

// WriteBlock writes the block to the chain.
func (bc *BlockChain) WriteBlock(block *types.Block) (status WriteStatus, err error) {
bc.wg.Add(1)
defer bc.wg.Done()

// Calculate the total difficulty of the block
ptd := bc.GetTd(block.ParentHash(), block.NumberU64()-1)
if ptd == nil {
return NonStatTy, consensus.ErrUnknownAncestor
}
// Make sure no inconsistent state is leaked during insertion
bc.mu.Lock()
defer bc.mu.Unlock()

localTd := bc.GetTd(bc.currentBlock.Hash(), bc.currentBlock.NumberU64())
externTd := new(big.Int).Add(block.Difficulty(), ptd)

// Irrelevant of the canonical status, write the block itself to the database
if err := bc.hc.WriteTd(block.Hash(), block.NumberU64(), externTd); err != nil {
log.Crit("Failed to write block total difficulty", "err", err)
}
if err := WriteBlock(bc.chainDb, block); err != nil {
log.Crit("Failed to write block contents", "err", err)
}

// If the total difficulty is higher than our known, add it to the canonical chain
// Second clause in the if statement reduces the vulnerability to selfish mining.
// Please refer to http://www.cs.cornell.edu/~ie53/publications/btcProcFC.pdf
if externTd.Cmp(localTd) > 0 || (externTd.Cmp(localTd) == 0 && mrand.Float64() < 0.5) {
```

```
// Reorganise the chain if the parent is not the head block
if block.ParentHash() != bc.currentBlock.Hash() {
if err := bc.reorg(bc.currentBlock, block); err != nil {
return NonStatTy, err
}
}
bc.insert(block) // Insert the block as the new head of the chain
status = CanonStatTy
} else {
status = SideStatTy
}

bc.futureBlocks.Remove(block.Hash())

return
}

// InsertChain will attempt to insert the given chain in to the canonical chain or, otherwise, create a fork. If an error is returned
// it will return the index number of the failing block as well an error describing what went wrong (for possible errors see core/errors.go).
func (bc *BlockChain) InsertChain(chain types.Blocks) (int, error) {
// Do a sanity check that the provided chain is actually ordered and linked
for i := 1; i < len(chain); i++ {
if chain[i].NumberU64() != chain[i-1].NumberU64()+1 || chain[i].ParentHash() != chain[i-1].Hash() {
// Chain broke ancestry, log a messge (programming error) and skip insertion
log.Error("Non contiguous block insert", "number", chain[i].Number(), "hash", chain[i].Hash(),
"parent", chain[i].ParentHash(), "prevnumber", chain[i-1].Number(), "prevhash", chain[i-1].Hash())

return 0, fmt.Errorf("non contiguous insert: item %d is #%d [%x…], item %d is #%d [%x…] (parent [%x…])", i-1, chain[i-1].NumberU64(),
chain[i-1].Hash().Bytes()[:4], i, chain[i].NumberU64(), chain[i].Hash().Bytes()[:4],
chain[i].ParentHash().Bytes()[:4])
}
}
// Pre-checks passed, start the full block imports
bc.wg.Add(1)
defer bc.wg.Done()

bc.chainmu.Lock()
defer bc.chainmu.Unlock()
```

```
// A queued approach to delivering events. This is generally
// faster than direct delivery and requires much less mutex
// acquiring.
var (
stats         = insertStats{startTime: mclock.Now()}
events        = make([]interface{}, 0, len(chain))
coalescedLogs []*types.Log
)
// Start the parallel header verifier
headers := make([]*types.Header, len(chain))
seals := make([]bool, len(chain))

for i, block := range chain {
headers[i] = block.Header()
seals[i] = true
}
abort, results := bc.engine.VerifyHeaders(bc, headers, seals)
defer close(abort)

// Iterate over the blocks and insert when the verifier permits
for i, block := range chain {
// If the chain is terminating, stop processing blocks
if atomic.LoadInt32(&bc.procInterrupt) == 1 {
log.Debug("Premature abort during blocks processing")
break
}
// If the header is a banned one, straight out abort
if BadHashes[block.Hash()] {
bc.reportBlock(block, nil, ErrBlacklistedHash)
return i, ErrBlacklistedHash
}
// Wait for the block's verification to complete
bstart := time.Now()

err := <-results
if err == nil {
err = bc.Validator().ValidateBody(block)
}
if err != nil {
if err == ErrKnownBlock {
stats.ignored++
continue
```

```
}

if err == consensus.ErrFutureBlock {
// Allow up to MaxFuture second in the future blocks. If this limit
// is exceeded the chain is discarded and processed at a later time
// if given.
max := big.NewInt(time.Now().Unix() + maxTimeFutureBlocks)
if block.Time().Cmp(max) > 0 {
return i, fmt.Errorf("future block: %v > %v", block.Time(), max)
}
bc.futureBlocks.Add(block.Hash(), block)
stats.queued++
continue
}

if err == consensus.ErrUnknownAncestor && bc.futureBlocks.Contains(block.ParentHash()) {
bc.futureBlocks.Add(block.Hash(), block)
stats.queued++
continue
}

bc.reportBlock(block, nil, err)
return i, err
}
// Create a new statedb using the parent block and report an
// error if it fails.
var parent *types.Block
if i == 0 {
parent = bc.GetBlock(block.ParentHash(), block.NumberU64()-1)
} else {
parent = chain[i-1]
}
state, err := state.New(parent.Root(), bc.stateCache)
if err != nil {
return i, err
}
// Process block using the parent state as reference point.
receipts, logs, usedGas, err := bc.processor.Process(block, state, bc.vmConfig)
if err != nil {
bc.reportBlock(block, receipts, err)
return i, err
}
```

```
// Validate the state using the default validator
err = bc.Validator().ValidateState(block, parent, state, receipts, usedGas)
if err != nil {
bc.reportBlock(block, receipts, err)
return i, err
}
// Write state changes to database
if _, err = state.CommitTo(bc.chainDb, bc.config.IsEIP158(block.Number())); err != nil {
return i, err
}

// coalesce logs for later processing
coalescedLogs = append(coalescedLogs, logs...)

if err = WriteBlockReceipts(bc.chainDb, block.Hash(), block.NumberU64(), receipts); err != nil {
return i, err
}

// write the block to the chain and get the status
status, err := bc.WriteBlock(block)
if err != nil {
return i, err
}

switch status {
case CanonStatTy:
log.Debug("Inserted new block", "number", block.Number(), "hash", block.Hash(), "uncles",
len(block.Uncles()),
"txs", len(block.Transactions()), "gas", block.GasUsed(), "elapsed",
common.PrettyDuration(time.Since(bstart)))

blockInsertTimer.UpdateSince(bstart)
events = append(events, ChainEvent{block, block.Hash(), logs})

// This puts transactions in a extra db for rpc
if err := WriteTransactions(bc.chainDb, block); err != nil {
return i, err
}
// store the receipts
if err := WriteReceipts(bc.chainDb, receipts); err != nil {
return i, err
}
```

```
// Write map map bloom filters
if err := WriteMipmapBloom(bc.chainDb, block.NumberU64(), receipts); err != nil {
return i, err
}
// Write hash preimages
if err := WritePreimages(bc.chainDb, block.NumberU64(), state.Preimages()); err != nil {
return i, err
}
case SideStatTy:
log.Debug("Inserted forked block", "number", block.Number(), "hash", block.Hash(), "diff",
block.Difficulty(), "elapsed",
common.PrettyDuration(time.Since(bstart)), "txs", len(block.Transactions()), "gas",
block.GasUsed(), "uncles", len(block.Uncles()))

blockInsertTimer.UpdateSince(bstart)
events = append(events, ChainSideEvent{block})
}
stats.processed++
stats.usedGas += usedGas.Uint64()
stats.report(chain, i)
}
go bc.postChainEvents(events, coalescedLogs)

return 0, nil
}

// insertStats tracks and reports on block insertion.
type insertStats struct {
queued, processed, ignored int
usedGas                    uint64
lastIndex                  int
startTime                  mclock.AbsTime
}

// statsReportLimit is the time limit during import after which we always print
// out progress. This avoids the user wondering what's going on.
const statsReportLimit = 8 * time.Second

// report prints statistics if some number of blocks have been processed
// or more than a few seconds have passed since the last message.
func (st *insertStats) report(chain []*types.Block, index int) {
// Fetch the timings for the batch
```

```
var (
now     = mclock.Now()
elapsed = time.Duration(now) - time.Duration(st.startTime)
)
// If we're at the last block of the batch or report period reached, log
if index == len(chain)-1 || elapsed >= statsReportLimit {
var (
end = chain[index]
txs = countTransactions(chain[st.lastIndex : index+1])
)
context := []interface{}{
"blocks", st.processed, "txs", txs, "mgas", float64(st.usedGas) / 1000000,
"elapsed", common.PrettyDuration(elapsed), "mgasps", float64(st.usedGas) * 1000 /
float64(elapsed),
"number", end.Number(), "hash", end.Hash(),
}
if st.queued > 0 {
context = append(context, []interface{}{"queued", st.queued}...)
}
if st.ignored > 0 {
context = append(context, []interface{}{"ignored", st.ignored}...)
}
log.Info("Imported new chain segment", context...)

*st = insertStats{startTime: now, lastIndex: index + 1}
}
}

func countTransactions(chain []*types.Block) (c int) {
for _, b := range chain {
c += len(b.Transactions())
}
return c
}

// reorgs takes two blocks, an old chain and a new chain and will reconstruct the blocks and inserts
them
// to be part of the new canonical chain and accumulates potential missing transactions and post
an
// event about them
func (bc *BlockChain) reorg(oldBlock, newBlock *types.Block) error {
var (
```

```
newChain    types.Blocks
oldChain    types.Blocks
commonBlock *types.Block
deletedTxs  types.Transactions
deletedLogs []*types.Log
// collectLogs collects the logs that were generated during the
// processing of the block that corresponds with the given hash.
// These logs are later announced as deleted.
collectLogs = func(h common.Hash) {
// Coalesce logs and set 'Removed'.
receipts := GetBlockReceipts(bc.chainDb, h, bc.hc.GetBlockNumber(h))
for _, receipt := range receipts {
for _, log := range receipt.Logs {
del := *log
del.Removed = true
deletedLogs = append(deletedLogs, &del)
}
}
}
)

// first reduce whoever is higher bound
if oldBlock.NumberU64() > newBlock.NumberU64() {
// reduce old chain
for ; oldBlock != nil && oldBlock.NumberU64() != newBlock.NumberU64(); oldBlock =
bc.GetBlock(oldBlock.ParentHash(), oldBlock.NumberU64()-1) {
oldChain = append(oldChain, oldBlock)
deletedTxs = append(deletedTxs, oldBlock.Transactions()...)

collectLogs(oldBlock.Hash())
}
} else {
// reduce new chain and append new chain blocks for inserting later on
for ; newBlock != nil && newBlock.NumberU64() != oldBlock.NumberU64(); newBlock =
bc.GetBlock(newBlock.ParentHash(), newBlock.NumberU64()-1) {
newChain = append(newChain, newBlock)
}
}
if oldBlock == nil {
return fmt.Errorf("Invalid old chain")
}
if newBlock == nil {
```

```
return fmt.Errorf("Invalid new chain")
}

for {
if oldBlock.Hash() == newBlock.Hash() {
commonBlock = oldBlock
break
}

oldChain = append(oldChain, oldBlock)
newChain = append(newChain, newBlock)
deletedTxs = append(deletedTxs, oldBlock.Transactions()...)
collectLogs(oldBlock.Hash())

oldBlock, newBlock = bc.GetBlock(oldBlock.ParentHash(), oldBlock.NumberU64()-1),
bc.GetBlock(newBlock.ParentHash(), newBlock.NumberU64()-1)
if oldBlock == nil {
return fmt.Errorf("Invalid old chain")
}
if newBlock == nil {
return fmt.Errorf("Invalid new chain")
}
}
// Ensure the user sees large reorgs
if len(oldChain) > 0 && len(newChain) > 0 {
logFn := log.Debug
if len(oldChain) > 63 {
logFn = log.Warn
}
logFn("Chain split detected", "number", commonBlock.Number(), "hash", commonBlock.Hash(),
"drop", len(oldChain), "dropfrom", oldChain[0].Hash(), "add", len(newChain), "addfrom",
newChain[0].Hash())
} else {
log.Error("Impossible reorg, please file an issue", "oldnum", oldBlock.Number(), "oldhash",
oldBlock.Hash(), "newnum", newBlock.Number(), "newhash", newBlock.Hash())
}
var addedTxs types.Transactions
// insert blocks. Order does not matter. Last block will be written in ImportChain itself which creates
the new head properly
for _, block := range newChain {
// insert the block in the canonical way, re-writing history
bc.insert(block)
```

```
// write canonical receipts and transactions
if err := WriteTransactions(bc.chainDb, block); err != nil {
return err
}
receipts := GetBlockReceipts(bc.chainDb, block.Hash(), block.NumberU64())
// write receipts
if err := WriteReceipts(bc.chainDb, receipts); err != nil {
return err
}
// Write map map bloom filters
if err := WriteMipmapBloom(bc.chainDb, block.NumberU64(), receipts); err != nil {
return err
}
addedTxs = append(addedTxs, block.Transactions()...)
}

// calculate the difference between deleted and added transactions
diff := types.TxDifference(deletedTxs, addedTxs)
// When transactions get deleted from the database that means the
// receipts that were created in the fork must also be deleted
for _, tx := range diff {
DeleteReceipt(bc.chainDb, tx.Hash())
DeleteTransaction(bc.chainDb, tx.Hash())
}
// Must be posted in a goroutine because of the transaction pool trying
// to acquire the chain manager lock
if len(diff) > 0 {
go bc.eventMux.Post(RemovedTransactionEvent{diff})
}
if len(deletedLogs) > 0 {
go bc.eventMux.Post(RemovedLogsEvent{deletedLogs})
}

if len(oldChain) > 0 {
go func() {
for _, block := range oldChain {
bc.eventMux.Post(ChainSideEvent{Block: block})
}
}()
}

return nil
```

```
}

// postChainEvents iterates over the events generated by a chain insertion and
// posts them into the event mux.
func (bc *BlockChain) postChainEvents(events []interface{}, logs []*types.Log) {
// post event logs for further processing
bc.eventMux.Post(logs)
for _, event := range events {
if event, ok := event.(ChainEvent); ok {
// We need some control over the mining operation. Acquiring locks and waiting
// for the miner to create new block takes too long and in most cases isn't
// even necessary.
if bc.LastBlockHash() == event.Hash {
bc.eventMux.Post(ChainHeadEvent{event.Block})
}
}
// Fire the insertion events individually too
bc.eventMux.Post(event)
}
}

func (bc *BlockChain) update() {
futureTimer := time.Tick(5 * time.Second)
for {
select {
case <-futureTimer:
bc.procFutureBlocks()
case <-bc.quit:
return
}
}
}

// BadBlockArgs represents the entries in the list returned when bad blocks are queried.
type BadBlockArgs struct {
Hash   common.Hash   `json:"hash"`
Header *types.Header `json:"header"`
}

// BadBlocks returns a list of the last 'bad blocks' that the client has seen on the network
func (bc *BlockChain) BadBlocks() ([]BadBlockArgs, error) {
headers := make([]BadBlockArgs, 0, bc.badBlocks.Len())
```

```
for _, hash := range bc.badBlocks.Keys() {
if hdr, exist := bc.badBlocks.Peek(hash); exist {
header := hdr.(*types.Header)
headers = append(headers, BadBlockArgs{header.Hash(), header})
}
}
return headers, nil
}

// addBadBlock adds a bad block to the bad-block LRU cache
func (bc *BlockChain) addBadBlock(block *types.Block) {
bc.badBlocks.Add(block.Header().Hash(), block.Header())
}

// reportBlock logs a bad block error.
func (bc *BlockChain) reportBlock(block *types.Block, receipts types.Receipts, err error) {
bc.addBadBlock(block)

var receiptString string
for _, receipt := range receipts {
receiptString += fmt.Sprintf("\t%v\n", receipt)
}
log.Error(fmt.Sprintf(`
########## BAD BLOCK #########
Chain config: %v

Number: %v
Hash: 0x%x
%v

Error: %v
##############################
`, bc.config, block.Number(), block.Hash(), receiptString, err))
}

// InsertHeaderChain attempts to insert the given header chain in to the local
// chain, possibly creating a reorg. If an error is returned, it will return the
// index number of the failing header as well an error describing what went wrong.
//
// The verify parameter can be used to fine tune whether nonce verification
// should be done or not. The reason behind the optional check is because some
// of the header retrieval mechanisms already need to verify nonces, as well as
```

```
// because nonces can be verified sparsely, not needing to check each.
func (bc *BlockChain) InsertHeaderChain(chain []*types.Header, checkFreq int) (int, error) {
start := time.Now()
if i, err := bc.hc.ValidateHeaderChain(chain, checkFreq); err != nil {
return i, err
}

// Make sure only one thread manipulates the chain at once
bc.chainmu.Lock()
defer bc.chainmu.Unlock()

bc.wg.Add(1)
defer bc.wg.Done()

whFunc := func(header *types.Header) error {
bc.mu.Lock()
defer bc.mu.Unlock()

_, err := bc.hc.WriteHeader(header)
return err
}

return bc.hc.InsertHeaderChain(chain, whFunc, start)
}

// writeHeader writes a header into the local chain, given that its parent is
// already known. If the total difficulty of the newly inserted header becomes
// greater than the current known TD, the canonical chain is re-routed.
//
// Note: This method is not concurrent-safe with inserting blocks simultaneously
// into the chain, as side effects caused by reorganisations cannot be emulated
// without the real blocks. Hence, writing headers directly should only be done
// in two scenarios: pure-header mode of operation (light clients), or properly
// separated header/block phases (non-archive clients).
func (bc *BlockChain) writeHeader(header *types.Header) error {
bc.wg.Add(1)
defer bc.wg.Done()

bc.mu.Lock()
defer bc.mu.Unlock()

_, err := bc.hc.WriteHeader(header)
```

```
return err
}

// CurrentHeader retrieves the current head header of the canonical chain. The
// header is retrieved from the HeaderChain's internal cache.
func (bc *BlockChain) CurrentHeader() *types.Header {
bc.mu.RLock()
defer bc.mu.RUnlock()

return bc.hc.CurrentHeader()
}

// GetTd retrieves a block's total difficulty in the canonical chain from the
// database by hash and number, caching it if found.
func (bc *BlockChain) GetTd(hash common.Hash, number uint64) *big.Int {
return bc.hc.GetTd(hash, number)
}

// GetTdByHash retrieves a block's total difficulty in the canonical chain from the
// database by hash, caching it if found.
func (bc *BlockChain) GetTdByHash(hash common.Hash) *big.Int {
return bc.hc.GetTdByHash(hash)
}

// GetHeader retrieves a block header from the database by hash and number,
// caching it if found.
func (bc *BlockChain) GetHeader(hash common.Hash, number uint64) *types.Header {
return bc.hc.GetHeader(hash, number)
}

// GetHeaderByHash retrieves a block header from the database by hash, caching it if
// found.
func (bc *BlockChain) GetHeaderByHash(hash common.Hash) *types.Header {
return bc.hc.GetHeaderByHash(hash)
}

// HasHeader checks if a block header is present in the database or not, caching
// it if present.
func (bc *BlockChain) HasHeader(hash common.Hash) bool {
return bc.hc.HasHeader(hash)
}
```

```
// GetBlockHashesFromHash retrieves a number of block hashes starting at a given
// hash, fetching towards the genesis block.
func (bc *BlockChain) GetBlockHashesFromHash(hash common.Hash, max uint64)
[]common.Hash {
return bc.hc.GetBlockHashesFromHash(hash, max)
}

// GetHeaderByNumber retrieves a block header from the database by number,
// caching it (associated with its hash) if found.
func (bc *BlockChain) GetHeaderByNumber(number uint64) *types.Header {
return bc.hc.GetHeaderByNumber(number)
}

// Config retrieves the blockchain's chain configuration.
func (bc *BlockChain) Config() *params.ChainConfig { return bc.config }

// Engine retrieves the blockchain's consensus engine.
func (bc *BlockChain) Engine() consensus.Engine { return bc.engine }

63:F:\git\coin\ethereum\go-ethereum\core\blockchain_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"fmt"
"math/big"
"math/rand"
"testing"
"time"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/params"
)
```

```
// newTestBlockChain creates a blockchain without validation.
func newTestBlockChain(fake bool) *BlockChain {
db, _ := ethdb.NewMemDatabase()
gspec := &Genesis{
Config:     params.TestChainConfig,
Difficulty: big.NewInt(1),
}
gspec.MustCommit(db)
engine := ethash.NewFullFaker()
if !fake {
engine = ethash.NewTester()
}
blockchain, err := NewBlockChain(db, gspec.Config, engine, new(event.TypeMux), vm.Config{})
if err != nil {
panic(err)
}
blockchain.SetValidator(bproc{})
return blockchain
}

// Test fork of length N starting from block i
func testFork(t *testing.T, blockchain *BlockChain, i, n int, full bool, comparator func(td1, td2
*big.Int)) {
// Copy old chain up to #i into a new db
db, blockchain2, err := newCanonical(i, full)
if err != nil {
t.Fatal("could not make new canonical in testFork", err)
}
// Assert the chains have the same header/block at #i
var hash1, hash2 common.Hash
if full {
hash1 = blockchain.GetBlockByNumber(uint64(i)).Hash()
hash2 = blockchain2.GetBlockByNumber(uint64(i)).Hash()
} else {
hash1 = blockchain.GetHeaderByNumber(uint64(i)).Hash()
hash2 = blockchain2.GetHeaderByNumber(uint64(i)).Hash()
}
if hash1 != hash2 {
t.Errorf("chain content mismatch at %d: have hash %v, want hash %v", i, hash2, hash1)
}
// Extend the newly created chain
var (
```

```
blockChainB  []*types.Block
headerChainB []*types.Header
)
if full {
blockChainB = makeBlockChain(blockchain2.CurrentBlock(), n, db, forkSeed)
if _, err := blockchain2.InsertChain(blockChainB); err != nil {
t.Fatalf("failed to insert forking chain: %v", err)
}
} else {
headerChainB = makeHeaderChain(blockchain2.CurrentHeader(), n, db, forkSeed)
if _, err := blockchain2.InsertHeaderChain(headerChainB, 1); err != nil {
t.Fatalf("failed to insert forking chain: %v", err)
}
}
// Sanity check that the forked chain can be imported into the original
var tdPre, tdPost *big.Int

if full {
tdPre = blockchain.GetTdByHash(blockchain.CurrentBlock().Hash())
if err := testBlockChainImport(blockChainB, blockchain); err != nil {
t.Fatalf("failed to import forked block chain: %v", err)
}
tdPost = blockchain.GetTdByHash(blockChainB[len(blockChainB)-1].Hash())
} else {
tdPre = blockchain.GetTdByHash(blockchain.CurrentHeader().Hash())
if err := testHeaderChainImport(headerChainB, blockchain); err != nil {
t.Fatalf("failed to import forked header chain: %v", err)
}
tdPost = blockchain.GetTdByHash(headerChainB[len(headerChainB)-1].Hash())
}
// Compare the total difficulties of the chains
comparator(tdPre, tdPost)
}

func printChain(bc *BlockChain) {
for i := bc.CurrentBlock().Number().Uint64(); i > 0; i-- {
b := bc.GetBlockByNumber(uint64(i))
fmt.Printf("\t%x %v\n", b.Hash(), b.Difficulty())
}
}

// testBlockChainImport tries to process a chain of blocks, writing them into
```

```
// the database if successful.
func testBlockChainImport(chain types.Blocks, blockchain *BlockChain) error {
for _, block := range chain {
// Try and process the block
err := blockchain.engine.VerifyHeader(blockchain, block.Header(), true)
if err == nil {
err = blockchain.validator.ValidateBody(block)
}
if err != nil {
if err == ErrKnownBlock {
continue
}
return err
}
statedb, err := state.New(blockchain.GetBlockByHash(block.ParentHash()).Root(),
blockchain.stateCache)
if err != nil {
return err
}
receipts, _, usedGas, err := blockchain.Processor().Process(block, statedb, vm.Config{})
if err != nil {
blockchain.reportBlock(block, receipts, err)
return err
}
err = blockchain.validator.ValidateState(block, blockchain.GetBlockByHash(block.ParentHash()),
statedb, receipts, usedGas)
if err != nil {
blockchain.reportBlock(block, receipts, err)
return err
}
blockchain.mu.Lock()
WriteTd(blockchain.chainDb, block.Hash(), block.NumberU64(), new(big.Int).Add(block.Difficulty(),
blockchain.GetTdByHash(block.ParentHash())))
WriteBlock(blockchain.chainDb, block)
statedb.CommitTo(blockchain.chainDb, false)
blockchain.mu.Unlock()
}
return nil
}

// testHeaderChainImport tries to process a chain of header, writing them into
// the database if successful.
```

```
func testHeaderChainImport(chain []*types.Header, blockchain *BlockChain) error {
for _, header := range chain {
// Try and validate the header
if err := blockchain.engine.VerifyHeader(blockchain, header, false); err != nil {
return err
}
// Manually insert the header into the database, but don't reorganise (allows subsequent testing)
blockchain.mu.Lock()
WriteTd(blockchain.chainDb, header.Hash(), header.Number.Uint64(),
new(big.Int).Add(header.Difficulty, blockchain.GetTdByHash(header.ParentHash)))
WriteHeader(blockchain.chainDb, header)
blockchain.mu.Unlock()
}
return nil
}

func insertChain(done chan bool, blockchain *BlockChain, chain types.Blocks, t *testing.T) {
_, err := blockchain.InsertChain(chain)
if err != nil {
fmt.Println(err)
t.FailNow()
}
done <- true
}

func TestLastBlock(t *testing.T) {
bchain := newTestBlockChain(false)
block := makeBlockChain(bchain.CurrentBlock(), 1, bchain.chainDb, 0)[0]
bchain.insert(block)
if block.Hash() != GetHeadBlockHash(bchain.chainDb) {
t.Errorf("Write/Get HeadBlockHash failed")
}
}

// Tests that given a starting canonical chain of a given size, it can be extended
// with various length chains.
func TestExtendCanonicalHeaders(t *testing.T) { testExtendCanonical(t, false) }
func TestExtendCanonicalBlocks(t *testing.T)  { testExtendCanonical(t, true) }

func testExtendCanonical(t *testing.T, full bool) {
length := 5
```

```
// Make first chain starting from genesis
_, processor, err := newCanonical(length, full)
if err != nil {
t.Fatalf("failed to make new canonical chain: %v", err)
}
// Define the difficulty comparator
better := func(td1, td2 *big.Int) {
if td2.Cmp(td1) <= 0 {
t.Errorf("total difficulty mismatch: have %v, expected more than %v", td2, td1)
}
}
// Start fork from current height
testFork(t, processor, length, 1, full, better)
testFork(t, processor, length, 2, full, better)
testFork(t, processor, length, 5, full, better)
testFork(t, processor, length, 10, full, better)
}

// Tests that given a starting canonical chain of a given size, creating shorter
// forks do not take canonical ownership.
func TestShorterForkHeaders(t *testing.T) { testShorterFork(t, false) }
func TestShorterForkBlocks(t *testing.T)  { testShorterFork(t, true) }

func testShorterFork(t *testing.T, full bool) {
length := 10

// Make first chain starting from genesis
_, processor, err := newCanonical(length, full)
if err != nil {
t.Fatalf("failed to make new canonical chain: %v", err)
}
// Define the difficulty comparator
worse := func(td1, td2 *big.Int) {
if td2.Cmp(td1) >= 0 {
t.Errorf("total difficulty mismatch: have %v, expected less than %v", td2, td1)
}
}
// Sum of numbers must be less than `length` for this to be a shorter fork
testFork(t, processor, 0, 3, full, worse)
testFork(t, processor, 0, 7, full, worse)
testFork(t, processor, 1, 1, full, worse)
testFork(t, processor, 1, 7, full, worse)
```

```
testFork(t, processor, 5, 3, full, worse)
testFork(t, processor, 5, 4, full, worse)
}

// Tests that given a starting canonical chain of a given size, creating longer
// forks do take canonical ownership.
func TestLongerForkHeaders(t *testing.T) { testLongerFork(t, false) }
func TestLongerForkBlocks(t *testing.T)  { testLongerFork(t, true) }

func testLongerFork(t *testing.T, full bool) {
length := 10

// Make first chain starting from genesis
_, processor, err := newCanonical(length, full)
if err != nil {
t.Fatalf("failed to make new canonical chain: %v", err)
}
// Define the difficulty comparator
better := func(td1, td2 *big.Int) {
if td2.Cmp(td1) <= 0 {
t.Errorf("total difficulty mismatch: have %v, expected more than %v", td2, td1)
}
}
// Sum of numbers must be greater than `length` for this to be a longer fork
testFork(t, processor, 0, 11, full, better)
testFork(t, processor, 0, 15, full, better)
testFork(t, processor, 1, 10, full, better)
testFork(t, processor, 1, 12, full, better)
testFork(t, processor, 5, 6, full, better)
testFork(t, processor, 5, 8, full, better)
}

// Tests that given a starting canonical chain of a given size, creating equal
// forks do take canonical ownership.
func TestEqualForkHeaders(t *testing.T) { testEqualFork(t, false) }
func TestEqualForkBlocks(t *testing.T)  { testEqualFork(t, true) }

func testEqualFork(t *testing.T, full bool) {
length := 10

// Make first chain starting from genesis
_, processor, err := newCanonical(length, full)
```

```
if err != nil {
t.Fatalf("failed to make new canonical chain: %v", err)
}
// Define the difficulty comparator
equal := func(td1, td2 *big.Int) {
if td2.Cmp(td1) != 0 {
t.Errorf("total difficulty mismatch: have %v, want %v", td2, td1)
}
}
// Sum of numbers must be equal to `length` for this to be an equal fork
testFork(t, processor, 0, 10, full, equal)
testFork(t, processor, 1, 9, full, equal)
testFork(t, processor, 2, 8, full, equal)
testFork(t, processor, 5, 5, full, equal)
testFork(t, processor, 6, 4, full, equal)
testFork(t, processor, 9, 1, full, equal)
}

// Tests that chains missing links do not get accepted by the processor.
func TestBrokenHeaderChain(t *testing.T) { testBrokenChain(t, false) }
func TestBrokenBlockChain(t *testing.T)  { testBrokenChain(t, true) }

func testBrokenChain(t *testing.T, full bool) {
// Make chain starting from genesis
db, blockchain, err := newCanonical(10, full)
if err != nil {
t.Fatalf("failed to make new canonical chain: %v", err)
}
// Create a forked chain, and try to insert with a missing link
if full {
chain := makeBlockChain(blockchain.CurrentBlock(), 5, db, forkSeed)[1:]
if err := testBlockChainImport(chain, blockchain); err == nil {
t.Errorf("broken block chain not reported")
}
} else {
chain := makeHeaderChain(blockchain.CurrentHeader(), 5, db, forkSeed)[1:]
if err := testHeaderChainImport(chain, blockchain); err == nil {
t.Errorf("broken header chain not reported")
}
}
}
```

```
type bproc struct{}

func (bproc) ValidateBody(*types.Block) error { return nil }
func (bproc) ValidateState(block, parent *types.Block, state *state.StateDB, receipts
types.Receipts, usedGas *big.Int) error {
return nil
}
func (bproc) Process(block *types.Block, statedb *state.StateDB, cfg vm.Config) (types.Receipts,
[]*types.Log, *big.Int, error) {
return nil, nil, new(big.Int), nil
}

func makeHeaderChainWithDiff(genesis *types.Block, d []int, seed byte) []*types.Header {
blocks := makeBlockChainWithDiff(genesis, d, seed)
headers := make([]*types.Header, len(blocks))
for i, block := range blocks {
headers[i] = block.Header()
}
return headers
}

func makeBlockChainWithDiff(genesis *types.Block, d []int, seed byte) []*types.Block {
var chain []*types.Block
for i, difficulty := range d {
header := &types.Header{
Coinbase:    common.Address{seed},
Number:      big.NewInt(int64(i + 1)),
Difficulty:  big.NewInt(int64(difficulty)),
UncleHash:   types.EmptyUncleHash,
TxHash:      types.EmptyRootHash,
ReceiptHash: types.EmptyRootHash,
Time:        big.NewInt(int64(i) + 1),
}
if i == 0 {
header.ParentHash = genesis.Hash()
} else {
header.ParentHash = chain[i-1].Hash()
}
block := types.NewBlockWithHeader(header)
chain = append(chain, block)
}
return chain
```

```
}

// Tests that reorganising a long difficult chain after a short easy one
// overwrites the canonical numbers and links in the database.
func TestReorgLongHeaders(t *testing.T) { testReorgLong(t, false) }
func TestReorgLongBlocks(t *testing.T)  { testReorgLong(t, true) }

func testReorgLong(t *testing.T, full bool) {
testReorg(t, []int{1, 2, 4}, []int{1, 2, 3, 4}, 10, full)
}

// Tests that reorganising a short difficult chain after a long easy one
// overwrites the canonical numbers and links in the database.
func TestReorgShortHeaders(t *testing.T) { testReorgShort(t, false) }
func TestReorgShortBlocks(t *testing.T)  { testReorgShort(t, true) }

func testReorgShort(t *testing.T, full bool) {
testReorg(t, []int{1, 2, 3, 4}, []int{1, 10}, 11, full)
}

func testReorg(t *testing.T, first, second []int, td int64, full bool) {
bc := newTestBlockChain(true)

// Insert an easy and a difficult chain afterwards
if full {
bc.InsertChain(makeBlockChainWithDiff(bc.genesisBlock, first, 11))
bc.InsertChain(makeBlockChainWithDiff(bc.genesisBlock, second, 22))
} else {
bc.InsertHeaderChain(makeHeaderChainWithDiff(bc.genesisBlock, first, 11), 1)
bc.InsertHeaderChain(makeHeaderChainWithDiff(bc.genesisBlock, second, 22), 1)
}
// Check that the chain is valid number and link wise
if full {
prev := bc.CurrentBlock()
for block := bc.GetBlockByNumber(bc.CurrentBlock().NumberU64() - 1); block.NumberU64() != 0;
prev, block = block, bc.GetBlockByNumber(block.NumberU64()-1) {
if prev.ParentHash() != block.Hash() {
t.Errorf("parent block hash mismatch: have %x, want %x", prev.ParentHash(), block.Hash())
}
}
} else {
prev := bc.CurrentHeader()
```

```
for header := bc.GetHeaderByNumber(bc.CurrentHeader().Number.Uint64() - 1);
header.Number.Uint64() != 0; prev, header = header,
bc.GetHeaderByNumber(header.Number.Uint64()-1) {
if prev.ParentHash != header.Hash() {
t.Errorf("parent header hash mismatch: have %x, want %x", prev.ParentHash, header.Hash())
}
}
}
// Make sure the chain total difficulty is the correct one
want := new(big.Int).Add(bc.genesisBlock.Difficulty(), big.NewInt(td))
if full {
if have := bc.GetTdByHash(bc.CurrentBlock().Hash()); have.Cmp(want) != 0 {
t.Errorf("total difficulty mismatch: have %v, want %v", have, want)
}
} else {
if have := bc.GetTdByHash(bc.CurrentHeader().Hash()); have.Cmp(want) != 0 {
t.Errorf("total difficulty mismatch: have %v, want %v", have, want)
}
}
}

// Tests that the insertion functions detect banned hashes.
func TestBadHeaderHashes(t *testing.T) { testBadHashes(t, false) }
func TestBadBlockHashes(t *testing.T)  { testBadHashes(t, true) }

func testBadHashes(t *testing.T, full bool) {
bc := newTestBlockChain(true)

// Create a chain, ban a hash and try to import
var err error
if full {
blocks := makeBlockChainWithDiff(bc.genesisBlock, []int{1, 2, 4}, 10)
BadHashes[blocks[2].Header().Hash()] = true
_, err = bc.InsertChain(blocks)
} else {
headers := makeHeaderChainWithDiff(bc.genesisBlock, []int{1, 2, 4}, 10)
BadHashes[headers[2].Hash()] = true
_, err = bc.InsertHeaderChain(headers, 1)
}
if err != ErrBlacklistedHash {
t.Errorf("error mismatch: have: %v, want: %v", err, ErrBlacklistedHash)
}
```

```
}

// Tests that bad hashes are detected on boot, and the chain rolled back to a
// good state prior to the bad hash.
func TestReorgBadHeaderHashes(t *testing.T) { testReorgBadHashes(t, false) }
func TestReorgBadBlockHashes(t *testing.T)  { testReorgBadHashes(t, true) }

func testReorgBadHashes(t *testing.T, full bool) {
bc := newTestBlockChain(true)

// Create a chain, import and ban afterwards
headers := makeHeaderChainWithDiff(bc.genesisBlock, []int{1, 2, 3, 4}, 10)
blocks := makeBlockChainWithDiff(bc.genesisBlock, []int{1, 2, 3, 4}, 10)

if full {
if _, err := bc.InsertChain(blocks); err != nil {
t.Fatalf("failed to import blocks: %v", err)
}
if bc.CurrentBlock().Hash() != blocks[3].Hash() {
t.Errorf("last block hash mismatch: have: %x, want %x", bc.CurrentBlock().Hash(),
blocks[3].Header().Hash())
}
BadHashes[blocks[3].Header().Hash()] = true
defer func() { delete(BadHashes, blocks[3].Header().Hash()) }()
} else {
if _, err := bc.InsertHeaderChain(headers, 1); err != nil {
t.Fatalf("failed to import headers: %v", err)
}
if bc.CurrentHeader().Hash() != headers[3].Hash() {
t.Errorf("last header hash mismatch: have: %x, want %x", bc.CurrentHeader().Hash(),
headers[3].Hash())
}
BadHashes[headers[3].Hash()] = true
defer func() { delete(BadHashes, headers[3].Hash()) }()
}

// Create a new BlockChain and check that it rolled back the state.
ncm, err := NewBlockChain(bc.chainDb, bc.config, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})
if err != nil {
t.Fatalf("failed to create new chain manager: %v", err)
}
```

```
if full {
if ncm.CurrentBlock().Hash() != blocks[2].Header().Hash() {
t.Errorf("last block hash mismatch: have: %x, want %x", ncm.CurrentBlock().Hash(),
blocks[2].Header().Hash())
}
if blocks[2].Header().GasLimit.Cmp(ncm.GasLimit()) != 0 {
t.Errorf("last  block gasLimit mismatch: have: %x, want %x", ncm.GasLimit(),
blocks[2].Header().GasLimit)
}
} else {
if ncm.CurrentHeader().Hash() != headers[2].Hash() {
t.Errorf("last header hash mismatch: have: %x, want %x", ncm.CurrentHeader().Hash(),
headers[2].Hash())
}
}
}

// Tests chain insertions in the face of one entity containing an invalid nonce.
func TestHeadersInsertNonceError(t *testing.T) { testInsertNonceError(t, false) }
func TestBlocksInsertNonceError(t *testing.T)  { testInsertNonceError(t, true) }

func testInsertNonceError(t *testing.T, full bool) {
for i := 1; i < 25 && !t.Failed(); i++ {
// Create a pristine chain and database
db, blockchain, err := newCanonical(0, full)
if err != nil {
t.Fatalf("failed to create pristine chain: %v", err)
}
// Create and insert a chain with a failing nonce
var (
failAt  int
failRes int
failNum uint64
)
if full {
blocks := makeBlockChain(blockchain.CurrentBlock(), i, db, 0)

failAt = rand.Int() % len(blocks)
failNum = blocks[failAt].NumberU64()

blockchain.engine = ethash.NewFakeFailer(failNum)
failRes, err = blockchain.InsertChain(blocks)
```

```
} else {
headers := makeHeaderChain(blockchain.CurrentHeader(), i, db, 0)

failAt = rand.Int() % len(headers)
failNum = headers[failAt].Number.Uint64()

blockchain.engine = ethash.NewFakeFailer(failNum)
blockchain.hc.engine = blockchain.engine
failRes, err = blockchain.InsertHeaderChain(headers, 1)
}
// Check that the returned error indicates the failure.
if failRes != failAt {
t.Errorf("test %d: failure index mismatch: have %d, want %d", i, failRes, failAt)
}
// Check that all no blocks after the failing block have been inserted.
for j := 0; j < i-failAt; j++ {
if full {
if block := blockchain.GetBlockByNumber(failNum + uint64(j)); block != nil {
t.Errorf("test %d: invalid block in chain: %v", i, block)
}
} else {
if header := blockchain.GetHeaderByNumber(failNum + uint64(j)); header != nil {
t.Errorf("test %d: invalid header in chain: %v", i, header)
}
}
}
}
}

// Tests that fast importing a block chain produces the same chain data as the
// classical full block processing.
func TestFastVsFullChains(t *testing.T) {
// Configure and generate a sample block chain
var (
gendb, _ = ethdb.NewMemDatabase()
key, _   =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
address  = crypto.PubkeyToAddress(key.PublicKey)
funds    = big.NewInt(1000000000)
gspec    = &Genesis{
Config: params.TestChainConfig,
```

```
Alloc:  GenesisAlloc{address: {Balance: funds}},
}
genesis = gspec.MustCommit(gendb)
signer  = types.NewEIP155Signer(gspec.Config.ChainId)
)
blocks, receipts := GenerateChain(gspec.Config, genesis, gendb, 1024, func(i int, block
*BlockGen) {
block.SetCoinbase(common.Address{0x00})

// If the block number is multiple of 3, send a few bonus transactions to the miner
if i%3 == 2 {
for j := 0; j < i%4+1; j++ {
tx, err := types.SignTx(types.NewTransaction(block.TxNonce(address), common.Address{0x00},
big.NewInt(1000), bigTxGas, nil, nil), signer, key)
if err != nil {
panic(err)
}
block.AddTx(tx)
}
}
// If the block number is a multiple of 5, add a few bonus uncles to the block
if i%5 == 5 {
block.AddUncle(&types.Header{ParentHash: block.PrevBlock(i - 1).Hash(), Number:
big.NewInt(int64(i - 1))})
}
})
// Import the chain as an archive node for the comparison baseline
archiveDb, _ := ethdb.NewMemDatabase()
gspec.MustCommit(archiveDb)
archive, _ := NewBlockChain(archiveDb, gspec.Config, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})

if n, err := archive.InsertChain(blocks); err != nil {
t.Fatalf("failed to process block %d: %v", n, err)
}

// Fast import the chain as a non-archive node to test
fastDb, _ := ethdb.NewMemDatabase()
gspec.MustCommit(fastDb)
fast, _ := NewBlockChain(fastDb, gspec.Config, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})
```

```
headers := make([]*types.Header, len(blocks))
for i, block := range blocks {
headers[i] = block.Header()
}
if n, err := fast.InsertHeaderChain(headers, 1); err != nil {
t.Fatalf("failed to insert header %d: %v", n, err)
}
if n, err := fast.InsertReceiptChain(blocks, receipts); err != nil {
t.Fatalf("failed to insert receipt %d: %v", n, err)
}
// Iterate over all chain data components, and cross reference
for i := 0; i < len(blocks); i++ {
num, hash := blocks[i].NumberU64(), blocks[i].Hash()

if ftd, atd := fast.GetTdByHash(hash), archive.GetTdByHash(hash); ftd.Cmp(atd) != 0 {
t.Errorf("block #%d [%x]: td mismatch: have %v, want %v", num, hash, ftd, atd)
}
if fheader, aheader := fast.GetHeaderByHash(hash), archive.GetHeaderByHash(hash);
fheader.Hash() != aheader.Hash() {
t.Errorf("block #%d [%x]: header mismatch: have %v, want %v", num, hash, fheader, aheader)
}
if fblock, ablock := fast.GetBlockByHash(hash), archive.GetBlockByHash(hash); fblock.Hash() !=
ablock.Hash() {
t.Errorf("block #%d [%x]: block mismatch: have %v, want %v", num, hash, fblock, ablock)
} else if types.DeriveSha(fblock.Transactions()) != types.DeriveSha(ablock.Transactions()) {
t.Errorf("block #%d [%x]: transactions mismatch: have %v, want %v", num, hash,
fblock.Transactions(), ablock.Transactions())
} else if types.CalcUncleHash(fblock.Uncles()) != types.CalcUncleHash(ablock.Uncles()) {
t.Errorf("block #%d [%x]: uncles mismatch: have %v, want %v", num, hash, fblock.Uncles(),
ablock.Uncles())
}
if freceipts, areceipts := GetBlockReceipts(fastDb, hash, GetBlockNumber(fastDb, hash)),
GetBlockReceipts(archiveDb, hash, GetBlockNumber(archiveDb, hash));
types.DeriveSha(freceipts) != types.DeriveSha(areceipts) {
t.Errorf("block #%d [%x]: receipts mismatch: have %v, want %v", num, hash, freceipts, areceipts)
}
}
// Check that the canonical chains are the same between the databases
for i := 0; i < len(blocks)+1; i++ {
if fhash, ahash := GetCanonicalHash(fastDb, uint64(i)), GetCanonicalHash(archiveDb, uint64(i));
fhash != ahash {
t.Errorf("block #%d: canonical hash mismatch: have %v, want %v", i, fhash, ahash)
```

```
}
}
}

// Tests that various import methods move the chain head pointers to the correct
// positions.
func TestLightVsFastVsFullChainHeads(t *testing.T) {
// Configure and generate a sample block chain
var (
gendb, _ = ethdb.NewMemDatabase()
key, _   =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
address  = crypto.PubkeyToAddress(key.PublicKey)
funds    = big.NewInt(1000000000)
gspec    = &Genesis{Config: params.TestChainConfig, Alloc: GenesisAlloc{address: {Balance:
funds}}}
genesis  = gspec.MustCommit(gendb)
)
height := uint64(1024)
blocks, receipts := GenerateChain(gspec.Config, genesis, gendb, int(height), nil)

// Configure a subchain to roll back
remove := []common.Hash{}
for _, block := range blocks[height/2:] {
remove = append(remove, block.Hash())
}
// Create a small assertion method to check the three heads
assert := func(t *testing.T, kind string, chain *BlockChain, header uint64, fast uint64, block uint64)
{
if num := chain.CurrentBlock().NumberU64(); num != block {
t.Errorf("%s head block mismatch: have #%v, want #%v", kind, num, block)
}
if num := chain.CurrentFastBlock().NumberU64(); num != fast {
t.Errorf("%s head fast-block mismatch: have #%v, want #%v", kind, num, fast)
}
if num := chain.CurrentHeader().Number.Uint64(); num != header {
t.Errorf("%s head header mismatch: have #%v, want #%v", kind, num, header)
}
}
// Import the chain as an archive node and ensure all pointers are updated
archiveDb, _ := ethdb.NewMemDatabase()
```

```
gspec.MustCommit(archiveDb)

archive, _ := NewBlockChain(archiveDb, gspec.Config, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})
if n, err := archive.InsertChain(blocks); err != nil {
t.Fatalf("failed to process block %d: %v", n, err)
}
assert(t, "archive", archive, height, height, height)
archive.Rollback(remove)
assert(t, "archive", archive, height/2, height/2, height/2)

// Import the chain as a non-archive node and ensure all pointers are updated
fastDb, _ := ethdb.NewMemDatabase()
gspec.MustCommit(fastDb)
fast, _ := NewBlockChain(fastDb, gspec.Config, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})

headers := make([]*types.Header, len(blocks))
for i, block := range blocks {
headers[i] = block.Header()
}
if n, err := fast.InsertHeaderChain(headers, 1); err != nil {
t.Fatalf("failed to insert header %d: %v", n, err)
}
if n, err := fast.InsertReceiptChain(blocks, receipts); err != nil {
t.Fatalf("failed to insert receipt %d: %v", n, err)
}
assert(t, "fast", fast, height, height, 0)
fast.Rollback(remove)
assert(t, "fast", fast, height/2, height/2, 0)

// Import the chain as a light node and ensure all pointers are updated
lightDb, _ := ethdb.NewMemDatabase()
gspec.MustCommit(lightDb)

light, _ := NewBlockChain(lightDb, gspec.Config, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})
if n, err := light.InsertHeaderChain(headers, 1); err != nil {
t.Fatalf("failed to insert header %d: %v", n, err)
}
assert(t, "light", light, height, 0, 0)
light.Rollback(remove)
```

```
assert(t, "light", light, height/2, 0, 0)
}

// Tests that chain reorganisations handle transaction removals and reinsertions.
func TestChainTxReorgs(t *testing.T) {
var (
key1, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
key2, _ =
crypto.HexToECDSA("8a1f9a8f95be41cd7ccb6168179afb4504aefe388d1e14474d32c45c72ce7b
7a")
key3, _ =
crypto.HexToECDSA("49a7b37aa6f6645917e7b807e9d1c00d4fa71f18343b0d4122a4d2df64dd6f
ee")
addr1   = crypto.PubkeyToAddress(key1.PublicKey)
addr2   = crypto.PubkeyToAddress(key2.PublicKey)
addr3   = crypto.PubkeyToAddress(key3.PublicKey)
db, _   = ethdb.NewMemDatabase()
gspec   = &Genesis{
Config:   params.TestChainConfig,
GasLimit: 3141592,
Alloc: GenesisAlloc{
addr1: {Balance: big.NewInt(1000000)},
addr2: {Balance: big.NewInt(1000000)},
addr3: {Balance: big.NewInt(1000000)},
},
}
genesis = gspec.MustCommit(db)
signer  = types.NewEIP155Signer(gspec.Config.ChainId)
)

// Create two transactions shared between the chains:
//  - postponed: transaction included at a later block in the forked chain
//  - swapped: transaction included at the same block number in the forked chain
postponed, _ := types.SignTx(types.NewTransaction(0, addr1, big.NewInt(1000), bigTxGas, nil,
nil), signer, key1)
swapped, _ := types.SignTx(types.NewTransaction(1, addr1, big.NewInt(1000), bigTxGas, nil, nil),
signer, key1)

// Create two transactions that will be dropped by the forked chain:
//  - pastDrop: transaction dropped retroactively from a past block
```

```
//  - freshDrop: transaction dropped exactly at the block where the reorg is detected
var pastDrop, freshDrop *types.Transaction

// Create three transactions that will be added in the forked chain:
//  - pastAdd:   transaction added before the reorganization is detected
//  - freshAdd:  transaction added at the exact block the reorg is detected
//  - futureAdd: transaction added after the reorg has already finished
var pastAdd, freshAdd, futureAdd *types.Transaction

chain, _ := GenerateChain(gspec.Config, genesis, db, 3, func(i int, gen *BlockGen) {
switch i {
case 0:
pastDrop, _ = types.SignTx(types.NewTransaction(gen.TxNonce(addr2), addr2, big.NewInt(1000),
bigTxGas, nil, nil), signer, key2)

gen.AddTx(pastDrop)  // This transaction will be dropped in the fork from below the split point
gen.AddTx(postponed) // This transaction will be postponed till block #3 in the fork

case 2:
freshDrop, _ = types.SignTx(types.NewTransaction(gen.TxNonce(addr2), addr2,
big.NewInt(1000), bigTxGas, nil, nil), signer, key2)

gen.AddTx(freshDrop) // This transaction will be dropped in the fork from exactly at the split point
gen.AddTx(swapped)   // This transaction will be swapped out at the exact height

gen.OffsetTime(9) // Lower the block difficulty to simulate a weaker chain
}
})
// Import the chain. This runs all block validation rules.
evmux := &event.TypeMux{}
blockchain, _ := NewBlockChain(db, gspec.Config, ethash.NewFaker(), evmux, vm.Config{})
if i, err := blockchain.InsertChain(chain); err != nil {
t.Fatalf("failed to insert original chain[%d]: %v", i, err)
}

// overwrite the old chain
chain, _ = GenerateChain(gspec.Config, genesis, db, 5, func(i int, gen *BlockGen) {
switch i {
case 0:
pastAdd, _ = types.SignTx(types.NewTransaction(gen.TxNonce(addr3), addr3, big.NewInt(1000),
bigTxGas, nil, nil), signer, key3)
gen.AddTx(pastAdd) // This transaction needs to be injected during reorg
```

```
case 2:
gen.AddTx(postponed) // This transaction was postponed from block #1 in the original chain
gen.AddTx(swapped)   // This transaction was swapped from the exact current spot in the original chain

freshAdd, _ = types.SignTx(types.NewTransaction(gen.TxNonce(addr3), addr3, big.NewInt(1000), bigTxGas, nil, nil), signer, key3)
gen.AddTx(freshAdd) // This transaction will be added exactly at reorg time

case 3:
futureAdd, _ = types.SignTx(types.NewTransaction(gen.TxNonce(addr3), addr3, big.NewInt(1000), bigTxGas, nil, nil), signer, key3)
gen.AddTx(futureAdd) // This transaction will be added after a full reorg
}
})
if _, err := blockchain.InsertChain(chain); err != nil {
t.Fatalf("failed to insert forked chain: %v", err)
}

// removed tx
for i, tx := range (types.Transactions{pastDrop, freshDrop}) {
if txn, _, _, _ := GetTransaction(db, tx.Hash()); txn != nil {
t.Errorf("drop %d: tx %v found while shouldn't have been", i, txn)
}
if GetReceipt(db, tx.Hash()) != nil {
t.Errorf("drop %d: receipt found while shouldn't have been", i)
}
}
// added tx
for i, tx := range (types.Transactions{pastAdd, freshAdd, futureAdd}) {
if txn, _, _, _ := GetTransaction(db, tx.Hash()); txn == nil {
t.Errorf("add %d: expected tx to be found", i)
}
if GetReceipt(db, tx.Hash()) == nil {
t.Errorf("add %d: expected receipt to be found", i)
}
}
// shared tx
for i, tx := range (types.Transactions{postponed, swapped}) {
if txn, _, _, _ := GetTransaction(db, tx.Hash()); txn == nil {
t.Errorf("share %d: expected tx to be found", i)
```

```
}
if GetReceipt(db, tx.Hash()) == nil {
t.Errorf("share %d: expected receipt to be found", i)
}
}
}

func TestLogReorgs(t *testing.T) {

var (
key1, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
addr1   = crypto.PubkeyToAddress(key1.PublicKey)
db, _   = ethdb.NewMemDatabase()
// this code generates a log
code    =
common.Hex2Bytes("60606040525b7f24ec1d3ff24c2f6ff210738839dbc339cd45a5294d85c79361
016243157aae7b6040518090506040518090390a15b600a8060416000396000f3606060405260
08565b00")
gspec   = &Genesis{Config: params.TestChainConfig, Alloc: GenesisAlloc{addr1: {Balance:
big.NewInt(10000000000000)}}}
genesis = gspec.MustCommit(db)
signer  = types.NewEIP155Signer(gspec.Config.ChainId)
)

var evmux event.TypeMux
blockchain, _ := NewBlockChain(db, gspec.Config, ethash.NewFaker(), &evmux, vm.Config{})

subs := evmux.Subscribe(RemovedLogsEvent{})
chain, _ := GenerateChain(params.TestChainConfig, genesis, db, 2, func(i int, gen *BlockGen) {
if i == 1 {
tx, err := types.SignTx(types.NewContractCreation(gen.TxNonce(addr1), new(big.Int),
big.NewInt(1000000), new(big.Int), code), signer, key1)
if err != nil {
t.Fatalf("failed to create tx: %v", err)
}
gen.AddTx(tx)
}
})
if _, err := blockchain.InsertChain(chain); err != nil {
t.Fatalf("failed to insert chain: %v", err)
```

```
}

chain, _ = GenerateChain(params.TestChainConfig, genesis, db, 3, func(i int, gen *BlockGen) {})
if _, err := blockchain.InsertChain(chain); err != nil {
t.Fatalf("failed to insert forked chain: %v", err)
}

ev := <-subs.Chan()
if len(ev.Data.(RemovedLogsEvent).Logs) == 0 {
t.Error("expected logs")
}
}

func TestReorgSideEvent(t *testing.T) {
var (
db, _   = ethdb.NewMemDatabase()
key1, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
addr1   = crypto.PubkeyToAddress(key1.PublicKey)
gspec   = &Genesis{
Config: params.TestChainConfig,
Alloc:  GenesisAlloc{addr1: {Balance: big.NewInt(10000000000000)}},
}
genesis = gspec.MustCommit(db)
signer  = types.NewEIP155Signer(gspec.Config.ChainId)
)

evmux := &event.TypeMux{}
blockchain, _ := NewBlockChain(db, gspec.Config, ethash.NewFaker(), evmux, vm.Config{})

chain, _ := GenerateChain(gspec.Config, genesis, db, 3, func(i int, gen *BlockGen) {})
if _, err := blockchain.InsertChain(chain); err != nil {
t.Fatalf("failed to insert chain: %v", err)
}

replacementBlocks, _ := GenerateChain(gspec.Config, genesis, db, 4, func(i int, gen *BlockGen) {
tx, err := types.SignTx(types.NewContractCreation(gen.TxNonce(addr1), new(big.Int),
big.NewInt(1000000), new(big.Int), nil), signer, key1)
if i == 2 {
gen.OffsetTime(-1)
}
```

```
if err != nil {
t.Fatalf("failed to create tx: %v", err)
}
gen.AddTx(tx)
})
subs := evmux.Subscribe(ChainSideEvent{})
if _, err := blockchain.InsertChain(replacementBlocks); err != nil {
t.Fatalf("failed to insert chain: %v", err)
}

// first two block of the secondary chain are for a brief moment considered
// side chains because up to that point the first one is considered the
// heavier chain.
expectedSideHashes := map[common.Hash]bool{
replacementBlocks[0].Hash(): true,
replacementBlocks[1].Hash(): true,
chain[0].Hash():             true,
chain[1].Hash():             true,
chain[2].Hash():             true,
}

i := 0

const timeoutDura = 10 * time.Second
timeout := time.NewTimer(timeoutDura)
done:
for {
select {
case ev := <-subs.Chan():
block := ev.Data.(ChainSideEvent).Block
if _, ok := expectedSideHashes[block.Hash()]; !ok {
t.Errorf("%d: didn't expect %x to be in side chain", i, block.Hash())
}
i++

if i == len(expectedSideHashes) {
timeout.Stop()

break done
}
timeout.Reset(timeoutDura)
```

```
case <-timeout.C:
t.Fatal("Timeout. Possibly not all blocks were triggered for sideevent")
}
}

// make sure no more events are fired
select {
case e := <-subs.Chan():
t.Errorf("unexpected event fired: %v", e)
case <-time.After(250 * time.Millisecond):
}

}

// Tests if the canonical block can be fetched from the database during chain insertion.
func TestCanonicalBlockRetrieval(t *testing.T) {
bc := newTestBlockChain(false)
chain, _ := GenerateChain(bc.config, bc.genesisBlock, bc.chainDb, 10, func(i int, gen *BlockGen)
{})

for i := range chain {
go func(block *types.Block) {
// try to retrieve a block by its canonical hash and see if the block data can be retrieved.
for {
ch := GetCanonicalHash(bc.chainDb, block.NumberU64())
if ch == (common.Hash{}) {
continue // busy wait for canonical hash to be written
}
if ch != block.Hash() {
t.Fatalf("unknown canonical hash, want %s, got %s", block.Hash().Hex(), ch.Hex())
}
fb := GetBlock(bc.chainDb, ch, block.NumberU64())
if fb == nil {
t.Fatalf("unable to retrieve block %d for canonical hash: %s", block.NumberU64(), ch.Hex())
}
if fb.Hash() != block.Hash() {
t.Fatalf("invalid block hash for block %d, want %s, got %s", block.NumberU64(),
block.Hash().Hex(), fb.Hash().Hex())
}
return
}
}(chain[i])
```

```
bc.InsertChain(types.Blocks{chain[i]})
}
}

func TestEIP155Transition(t *testing.T) {
// Configure and generate a sample block chain
var (
db, _      = ethdb.NewMemDatabase()
key, _     =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
address    = crypto.PubkeyToAddress(key.PublicKey)
funds      = big.NewInt(1000000000)
deleteAddr = common.Address{1}
gspec      = &Genesis{
Config: &params.ChainConfig{ChainId: big.NewInt(1), EIP155Block: big.NewInt(2),
HomesteadBlock: new(big.Int)},
Alloc:  GenesisAlloc{address: {Balance: funds}, deleteAddr: {Balance: new(big.Int)}},
}
genesis = gspec.MustCommit(db)
mux     event.TypeMux
)

blockchain, _ := NewBlockChain(db, gspec.Config, ethash.NewFaker(), &mux, vm.Config{})
blocks, _ := GenerateChain(gspec.Config, genesis, db, 4, func(i int, block *BlockGen) {
var (
tx      *types.Transaction
err     error
basicTx = func(signer types.Signer) (*types.Transaction, error) {
return types.SignTx(types.NewTransaction(block.TxNonce(address), common.Address{},
new(big.Int), big.NewInt(21000), new(big.Int), nil), signer, key)
}
)
switch i {
case 0:
tx, err = basicTx(types.HomesteadSigner{})
if err != nil {
t.Fatal(err)
}
block.AddTx(tx)
case 2:
```

```
tx, err = basicTx(types.HomesteadSigner{})
if err != nil {
t.Fatal(err)
}
block.AddTx(tx)

tx, err = basicTx(types.NewEIP155Signer(gspec.Config.ChainId))
if err != nil {
t.Fatal(err)
}
block.AddTx(tx)
case 3:
tx, err = basicTx(types.HomesteadSigner{})
if err != nil {
t.Fatal(err)
}
block.AddTx(tx)

tx, err = basicTx(types.NewEIP155Signer(gspec.Config.ChainId))
if err != nil {
t.Fatal(err)
}
block.AddTx(tx)
}
})

if _, err := blockchain.InsertChain(blocks); err != nil {
t.Fatal(err)
}
block := blockchain.GetBlockByNumber(1)
if block.Transactions()[0].Protected() {
t.Error("Expected block[0].txs[0] to not be replay protected")
}

block = blockchain.GetBlockByNumber(3)
if block.Transactions()[0].Protected() {
t.Error("Expected block[3].txs[0] to not be replay protected")
}
if !block.Transactions()[1].Protected() {
t.Error("Expected block[3].txs[1] to be replay protected")
}
if _, err := blockchain.InsertChain(blocks[4:]); err != nil {
```

```
t.Fatal(err)
}

// generate an invalid chain id transaction
config := &params.ChainConfig{ChainId: big.NewInt(2), EIP155Block: big.NewInt(2),
HomesteadBlock: new(big.Int)}
blocks, _ = GenerateChain(config, blocks[len(blocks)-1], db, 4, func(i int, block *BlockGen) {
var (
tx      *types.Transaction
err     error
basicTx = func(signer types.Signer) (*types.Transaction, error) {
return types.SignTx(types.NewTransaction(block.TxNonce(address), common.Address{},
new(big.Int), big.NewInt(21000), new(big.Int), nil), signer, key)
}
)
switch i {
case 0:
tx, err = basicTx(types.NewEIP155Signer(big.NewInt(2)))
if err != nil {
t.Fatal(err)
}
block.AddTx(tx)
}
})
_, err := blockchain.InsertChain(blocks)
if err != types.ErrInvalidChainId {
t.Error("expected error:", types.ErrInvalidChainId)
}
}

func TestEIP161AccountRemoval(t *testing.T) {
// Configure and generate a sample block chain
var (
db, _   = ethdb.NewMemDatabase()
key, _  =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
address = crypto.PubkeyToAddress(key.PublicKey)
funds   = big.NewInt(1000000000)
theAddr = common.Address{1}
gspec   = &Genesis{
Config: &params.ChainConfig{
```

```
ChainId:        big.NewInt(1),
HomesteadBlock: new(big.Int),
EIP155Block:    new(big.Int),
EIP158Block:    big.NewInt(2),
},
Alloc: GenesisAlloc{address: {Balance: funds}},
}
genesis       = gspec.MustCommit(db)
mux           event.TypeMux
blockchain, _ = NewBlockChain(db, gspec.Config, ethash.NewFaker(), &mux, vm.Config{})
)
blocks, _ := GenerateChain(gspec.Config, genesis, db, 3, func(i int, block *BlockGen) {
var (
tx     *types.Transaction
err    error
signer = types.NewEIP155Signer(gspec.Config.ChainId)
)
switch i {
case 0:
tx, err = types.SignTx(types.NewTransaction(block.TxNonce(address), theAddr, new(big.Int),
big.NewInt(21000), new(big.Int), nil), signer, key)
case 1:
tx, err = types.SignTx(types.NewTransaction(block.TxNonce(address), theAddr, new(big.Int),
big.NewInt(21000), new(big.Int), nil), signer, key)
case 2:
tx, err = types.SignTx(types.NewTransaction(block.TxNonce(address), theAddr, new(big.Int),
big.NewInt(21000), new(big.Int), nil), signer, key)
}
if err != nil {
t.Fatal(err)
}
block.AddTx(tx)
})
// account must exist pre eip 161
if _, err := blockchain.InsertChain(types.Blocks{blocks[0]}); err != nil {
t.Fatal(err)
}
if st, _ := blockchain.State(); !st.Exist(theAddr) {
t.Error("expected account to exist")
}

// account needs to be deleted post eip 161
```

```
if _, err := blockchain.InsertChain(types.Blocks{blocks[1]}); err != nil {
t.Fatal(err)
}
if st, _ := blockchain.State(); st.Exist(theAddr) {
t.Error("account should not exist")
}

// account musn't be created post eip 161
if _, err := blockchain.InsertChain(types.Blocks{blocks[2]}); err != nil {
t.Fatal(err)
}
if st, _ := blockchain.State(); st.Exist(theAddr) {
t.Error("account should not exist")
}
}
```

64:F:\git\coin\ethereum\go-ethereum\core\blocks.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import "github.com/ethereum/go-ethereum/common"

// BadHashes represent a set of manually tracked bad hashes (usually hard forks)
var BadHashes = map[common.Hash]bool{
common.HexToHash("05bef30ef572270f654746da22639a7a0c97dd97a7050b9e252391996aaeb
689"): true,
common.HexToHash("7d05d08cbc596a2e5e4f13b80a743e53e09221b5323c3a61946b20873e585
83f"): true,
}
```

65:F:\git\coin\ethereum\go-ethereum\core\block_validator.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"fmt"
"math/big"

"github.com/ethereum/go-ethereum/common/math"
"github.com/ethereum/go-ethereum/consensus"
```

```
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/params"
)

// BlockValidator is responsible for validating block headers, uncles and
// processed state.
//
// BlockValidator implements Validator.
type BlockValidator struct {
config *params.ChainConfig // Chain configuration options
bc     *BlockChain         // Canonical block chain
engine consensus.Engine    // Consensus engine used for validating
}

// NewBlockValidator returns a new block validator which is safe for re-use
func NewBlockValidator(config *params.ChainConfig, blockchain *BlockChain, engine
consensus.Engine) *BlockValidator {
validator := &BlockValidator{
config: config,
engine: engine,
bc:     blockchain,
}
return validator
}

// ValidateBody validates the given block's uncles and verifies the the block
// header's transaction and uncle roots. The headers are assumed to be already
// validated at this point.
func (v *BlockValidator) ValidateBody(block *types.Block) error {
// Check whether the block's known, and if not, that it's linkable
if v.bc.HasBlockAndState(block.Hash()) {
return ErrKnownBlock
}
if !v.bc.HasBlockAndState(block.ParentHash()) {
return consensus.ErrUnknownAncestor
}
// Header validity is known at this point, check the uncles and transactions
header := block.Header()
if err := v.engine.VerifyUncles(v.bc, block); err != nil {
return err
}
```

```
if hash := types.CalcUncleHash(block.Uncles()); hash != header.UncleHash {
return fmt.Errorf("uncle root hash mismatch: have %x, want %x", hash, header.UncleHash)
}
if hash := types.DeriveSha(block.Transactions()); hash != header.TxHash {
return fmt.Errorf("transaction root hash mismatch: have %x, want %x", hash, header.TxHash)
}
return nil
}

// ValidateState validates the various changes that happen after a state
// transition, such as amount of used gas, the receipt roots and the state root
// itself. ValidateState returns a database batch if the validation was a success
// otherwise nil and an error is returned.
func (v *BlockValidator) ValidateState(block, parent *types.Block, statedb *state.StateDB, receipts
types.Receipts, usedGas *big.Int) error {
header := block.Header()
if block.GasUsed().Cmp(usedGas) != 0 {
return fmt.Errorf("invalid gas used (remote: %v local: %v)", block.GasUsed(), usedGas)
}
// Validate the received block's bloom with the one derived from the generated receipts.
// For valid blocks this should always validate to true.
rbloom := types.CreateBloom(receipts)
if rbloom != header.Bloom {
return fmt.Errorf("invalid bloom (remote: %x  local: %x)", header.Bloom, rbloom)
}
// Tre receipt Trie's root (R = (Tr [[H1, R1], ... [Hn, R1]]))
receiptSha := types.DeriveSha(receipts)
if receiptSha != header.ReceiptHash {
return fmt.Errorf("invalid receipt root hash (remote: %x local: %x)", header.ReceiptHash,
receiptSha)
}
// Validate the state root against the received state root and throw
// an error if they don't match.
if root := statedb.IntermediateRoot(v.config.IsEIP158(header.Number)); header.Root != root {
return fmt.Errorf("invalid merkle root (remote: %x local: %x)", header.Root, root)
}
return nil
}

// CalcGasLimit computes the gas limit of the next block after parent.
// The result may be modified by the caller.
// This is miner strategy, not consensus protocol.
```

```
func CalcGasLimit(parent *types.Block) *big.Int {
// contrib = (parentGasUsed * 3 / 2) / 1024
contrib := new(big.Int).Mul(parent.GasUsed(), big.NewInt(3))
contrib = contrib.Div(contrib, big.NewInt(2))
contrib = contrib.Div(contrib, params.GasLimitBoundDivisor)

// decay = parentGasLimit / 1024 -1
decay := new(big.Int).Div(parent.GasLimit(), params.GasLimitBoundDivisor)
decay.Sub(decay, big.NewInt(1))

/*
strategy: gasLimit of block-to-mine is set based on parent's
gasUsed value.  if parentGasUsed > parentGasLimit * (2/3) then we
increase it, otherwise lower it (or leave it unchanged if it's right
at that usage) the amount increased/decreased depends on how far away
from parentGasLimit * (2/3) parentGasUsed is.
*/
gl := new(big.Int).Sub(parent.GasLimit(), decay)
gl = gl.Add(gl, contrib)
gl.Set(math.BigMax(gl, params.MinGasLimit))

// however, if we're now below the target (TargetGasLimit) we increase the
// limit as much as we can (parentGasLimit / 1024 -1)
if gl.Cmp(params.TargetGasLimit) < 0 {
gl.Add(parent.GasLimit(), decay)
gl.Set(math.BigMin(gl, params.TargetGasLimit))
}
return gl
}

66:F:\git\coin\ethereum\go-ethereum\core\block_validator_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"runtime"
"testing"
"time"

"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/core/types"
```

```
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/params"
)

// Tests that simple header verification works, for both good and bad blocks.
func TestHeaderVerification(t *testing.T) {
// Create a simple chain to verify
var (
testdb, _ = ethdb.NewMemDatabase()
gspec     = &Genesis{Config: params.TestChainConfig}
genesis   = gspec.MustCommit(testdb)
blocks, _ = GenerateChain(params.TestChainConfig, genesis, testdb, 8, nil)
)
headers := make([]*types.Header, len(blocks))
for i, block := range blocks {
headers[i] = block.Header()
}
// Run the header checker for blocks one-by-one, checking for both valid and invalid nonces
chain, _ := NewBlockChain(testdb, params.TestChainConfig, ethash.NewFaker(),
new(event.TypeMux), vm.Config{})

for i := 0; i < len(blocks); i++ {
for j, valid := range []bool{true, false} {
var results <-chan error

if valid {
engine := ethash.NewFaker()
_, results = engine.VerifyHeaders(chain, []*types.Header{headers[i]}, []bool{true})
} else {
engine := ethash.NewFakeFailer(headers[i].Number.Uint64())
_, results = engine.VerifyHeaders(chain, []*types.Header{headers[i]}, []bool{true})
}
// Wait for the verification result
select {
case result := <-results:
if (result == nil) != valid {
t.Errorf("test %d.%d: validity mismatch: have %v, want %v", i, j, result, valid)
}
case <-time.After(time.Second):
t.Fatalf("test %d.%d: verification timeout", i, j)
```

```
}
// Make sure no more data is returned
select {
case result := <-results:
t.Fatalf("test %d.%d: unexpected result returned: %v", i, j, result)
case <-time.After(25 * time.Millisecond):
}
}
chain.InsertChain(blocks[i : i+1])
}
}

// Tests that concurrent header verification works, for both good and bad blocks.
func TestHeaderConcurrentVerification2(t *testing.T)  { testHeaderConcurrentVerification(t, 2) }
func TestHeaderConcurrentVerification8(t *testing.T)  { testHeaderConcurrentVerification(t, 8) }
func TestHeaderConcurrentVerification32(t *testing.T) { testHeaderConcurrentVerification(t, 32) }

func testHeaderConcurrentVerification(t *testing.T, threads int) {
// Create a simple chain to verify
var (
testdb, _ = ethdb.NewMemDatabase()
gspec     = &Genesis{Config: params.TestChainConfig}
genesis   = gspec.MustCommit(testdb)
blocks, _ = GenerateChain(params.TestChainConfig, genesis, testdb, 8, nil)
)
headers := make([]*types.Header, len(blocks))
seals := make([]bool, len(blocks))

for i, block := range blocks {
headers[i] = block.Header()
seals[i] = true
}
// Set the number of threads to verify on
old := runtime.GOMAXPROCS(threads)
defer runtime.GOMAXPROCS(old)

// Run the header checker for the entire block chain at once both for a valid and
// also an invalid chain (enough if one arbitrary block is invalid).
for i, valid := range []bool{true, false} {
var results <-chan error

if valid {
```

```
chain, _ := NewBlockChain(testdb, params.TestChainConfig, ethash.NewFaker(),
new(event.TypeMux), vm.Config{})
_, results = chain.engine.VerifyHeaders(chain, headers, seals)
} else {
chain, _ := NewBlockChain(testdb, params.TestChainConfig,
ethash.NewFakeFailer(uint64(len(headers)-1)), new(event.TypeMux), vm.Config{})
_, results = chain.engine.VerifyHeaders(chain, headers, seals)
}
// Wait for all the verification results
checks := make(map[int]error)
for j := 0; j < len(blocks); j++ {
select {
case result := <-results:
checks[j] = result

case <-time.After(time.Second):
t.Fatalf("test %d.%d: verification timeout", i, j)
}
}
// Check nonce check validity
for j := 0; j < len(blocks); j++ {
want := valid || (j < len(blocks)-2) // We chose the last-but-one nonce in the chain to fail
if (checks[j] == nil) != want {
t.Errorf("test %d.%d: validity mismatch: have %v, want %v", i, j, checks[j], want)
}
if !want {
// A few blocks after the first error may pass verification due to concurrent
// workers. We don't care about those in this test, just that the correct block
// errors out.
break
}
}
// Make sure no more data is returned
select {
case result := <-results:
t.Fatalf("test %d: unexpected result returned: %v", i, result)
case <-time.After(25 * time.Millisecond):
}
}
}

// Tests that aborting a header validation indeed prevents further checks from being
```

```
// run, as well as checks that no left-over goroutines are leaked.
func TestHeaderConcurrentAbortion2(t *testing.T)  { testHeaderConcurrentAbortion(t, 2) }
func TestHeaderConcurrentAbortion8(t *testing.T)  { testHeaderConcurrentAbortion(t, 8) }
func TestHeaderConcurrentAbortion32(t *testing.T) { testHeaderConcurrentAbortion(t, 32) }

func testHeaderConcurrentAbortion(t *testing.T, threads int) {
// Create a simple chain to verify
var (
testdb, _ = ethdb.NewMemDatabase()
gspec     = &Genesis{Config: params.TestChainConfig}
genesis   = gspec.MustCommit(testdb)
blocks, _ = GenerateChain(params.TestChainConfig, genesis, testdb, 1024, nil)
)
headers := make([]*types.Header, len(blocks))
seals := make([]bool, len(blocks))

for i, block := range blocks {
headers[i] = block.Header()
seals[i] = true
}
// Set the number of threads to verify on
old := runtime.GOMAXPROCS(threads)
defer runtime.GOMAXPROCS(old)

// Start the verifications and immediately abort
chain, _ := NewBlockChain(testdb, params.TestChainConfig,
ethash.NewFakeDelayer(time.Millisecond), new(event.TypeMux), vm.Config{})
abort, results := chain.engine.VerifyHeaders(chain, headers, seals)
close(abort)

// Deplete the results channel
verified := 0
for depleted := false; !depleted; {
select {
case result := <-results:
if result != nil {
t.Errorf("header %d: validation failed: %v", verified, result)
}
verified++
case <-time.After(50 * time.Millisecond):
depleted = true
}
```

```
}
// Check that abortion was honored by not processing too many POWs
if verified > 2*threads {
t.Errorf("verification count too large: have %d, want below %d", verified, 2*threads)
}
}
```

67:F:\git\coin\ethereum\go-ethereum\core\chain_makers.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"fmt"
"math/big"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/consensus/misc"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/params"
)

// So we can deterministically seed different blockchains
var (
canonicalSeed = 1
forkSeed      = 2
)

// BlockGen creates blocks for testing.
// See GenerateChain for a detailed explanation.
type BlockGen struct {
i       int
parent  *types.Block
chain   []*types.Block
header  *types.Header
statedb *state.StateDB
```

```
gasPool  *GasPool
txs      []*types.Transaction
receipts []*types.Receipt
uncles   []*types.Header

config *params.ChainConfig
}

// SetCoinbase sets the coinbase of the generated block.
// It can be called at most once.
func (b *BlockGen) SetCoinbase(addr common.Address) {
if b.gasPool != nil {
if len(b.txs) > 0 {
panic("coinbase must be set before adding transactions")
}
panic("coinbase can only be set once")
}
b.header.Coinbase = addr
b.gasPool = new(GasPool).AddGas(b.header.GasLimit)
}

// SetExtra sets the extra data field of the generated block.
func (b *BlockGen) SetExtra(data []byte) {
b.header.Extra = data
}

// AddTx adds a transaction to the generated block. If no coinbase has
// been set, the block's coinbase is set to the zero address.
//
// AddTx panics if the transaction cannot be executed. In addition to
// the protocol-imposed limitations (gas limit, etc.), there are some
// further limitations on the content of transactions that can be
// added. Notably, contract code relying on the BLOCKHASH instruction
// will panic during execution.
func (b *BlockGen) AddTx(tx *types.Transaction) {
if b.gasPool == nil {
b.SetCoinbase(common.Address{})
}
b.statedb.Prepare(tx.Hash(), common.Hash{}, len(b.txs))
receipt, _, err := ApplyTransaction(b.config, nil, &b.header.Coinbase, b.gasPool, b.statedb,
b.header, tx, b.header.GasUsed, vm.Config{})
if err != nil {
```

```
panic(err)
}
b.txs = append(b.txs, tx)
b.receipts = append(b.receipts, receipt)
}

// Number returns the block number of the block being generated.
func (b *BlockGen) Number() *big.Int {
return new(big.Int).Set(b.header.Number)
}

// AddUncheckedReceipt forcefully adds a receipts to the block without a
// backing transaction.
//
// AddUncheckedReceipt will cause consensus failures when used during real
// chain processing. This is best used in conjunction with raw block insertion.
func (b *BlockGen) AddUncheckedReceipt(receipt *types.Receipt) {
b.receipts = append(b.receipts, receipt)
}

// TxNonce returns the next valid transaction nonce for the
// account at addr. It panics if the account does not exist.
func (b *BlockGen) TxNonce(addr common.Address) uint64 {
if !b.statedb.Exist(addr) {
panic("account does not exist")
}
return b.statedb.GetNonce(addr)
}

// AddUncle adds an uncle header to the generated block.
func (b *BlockGen) AddUncle(h *types.Header) {
b.uncles = append(b.uncles, h)
}

// PrevBlock returns a previously generated block by number. It panics if
// num is greater or equal to the number of the block being generated.
// For index -1, PrevBlock returns the parent block given to GenerateChain.
func (b *BlockGen) PrevBlock(index int) *types.Block {
if index >= b.i {
panic("block index out of range")
}
if index == -1 {
```

```
return b.parent
}
return b.chain[index]
}

// OffsetTime modifies the time instance of a block, implicitly changing its
// associated difficulty. It's useful to test scenarios where forking is not
// tied to chain length directly.
func (b *BlockGen) OffsetTime(seconds int64) {
b.header.Time.Add(b.header.Time, new(big.Int).SetInt64(seconds))
if b.header.Time.Cmp(b.parent.Header().Time) <= 0 {
panic("block time out of range")
}
b.header.Difficulty = ethash.CalcDifficulty(b.config, b.header.Time.Uint64(), b.parent.Header())
}

// GenerateChain creates a chain of n blocks. The first block's
// parent will be the provided parent. db is used to store
// intermediate states and should contain the parent's state trie.
//
// The generator function is called with a new block generator for
// every block. Any transactions and uncles added to the generator
// become part of the block. If gen is nil, the blocks will be empty
// and their coinbase will be the zero address.
//
// Blocks created by GenerateChain do not contain valid proof of work
// values. Inserting them into BlockChain requires use of FakePow or
// a similar non-validating proof of work implementation.
func GenerateChain(config *params.ChainConfig, parent *types.Block, db ethdb.Database, n int,
gen func(int, *BlockGen)) ([]*types.Block, []types.Receipts) {
if config == nil {
config = params.TestChainConfig
}
blocks, receipts := make(types.Blocks, n), make([]types.Receipts, n)
genblock := func(i int, h *types.Header, statedb *state.StateDB) (*types.Block, types.Receipts) {
b := &BlockGen{parent: parent, i: i, chain: blocks, header: h, statedb: statedb, config: config}
// Mutate the state and block according to any hard-fork specs
if daoBlock := config.DAOForkBlock; daoBlock != nil {
limit := new(big.Int).Add(daoBlock, params.DAOForkExtraRange)
if h.Number.Cmp(daoBlock) >= 0 && h.Number.Cmp(limit) < 0 {
if config.DAOForkSupport {
h.Extra = common.CopyBytes(params.DAOForkBlockExtra)
```

```
}
}
}
if config.DAOForkSupport && config.DAOForkBlock != nil &&
config.DAOForkBlock.Cmp(h.Number) == 0 {
misc.ApplyDAOHardFork(statedb)
}
// Execute any user modifications to the block and finalize it
if gen != nil {
gen(i, b)
}
ethash.AccumulateRewards(statedb, h, b.uncles)
root, err := statedb.CommitTo(db, config.IsEIP158(h.Number))
if err != nil {
panic(fmt.Sprintf("state write error: %v", err))
}
h.Root = root
return types.NewBlock(h, b.txs, b.uncles, b.receipts), b.receipts
}
for i := 0; i < n; i++ {
statedb, err := state.New(parent.Root(), state.NewDatabase(db))
if err != nil {
panic(err)
}
header := makeHeader(config, parent, statedb)
block, receipt := genblock(i, header, statedb)
blocks[i] = block
receipts[i] = receipt
parent = block
}
return blocks, receipts
}

func makeHeader(config *params.ChainConfig, parent *types.Block, state *state.StateDB)
*types.Header {
var time *big.Int
if parent.Time() == nil {
time = big.NewInt(10)
} else {
time = new(big.Int).Add(parent.Time(), big.NewInt(10)) // block time is fixed at 10 seconds
}
```

```
return &types.Header{
Root:       state.IntermediateRoot(config.IsEIP158(parent.Number())),
ParentHash: parent.Hash(),
Coinbase:   parent.Coinbase(),
Difficulty: ethash.CalcDifficulty(config, time.Uint64(), &types.Header{
Number:     parent.Number(),
Time:       new(big.Int).Sub(time, big.NewInt(10)),
Difficulty: parent.Difficulty(),
}),
GasLimit: CalcGasLimit(parent),
GasUsed:  new(big.Int),
Number:   new(big.Int).Add(parent.Number(), common.Big1),
Time:     time,
}
}

// newCanonical creates a chain database, and injects a deterministic canonical
// chain. Depending on the full flag, if creates either a full block chain or a
// header only chain.
func newCanonical(n int, full bool) (ethdb.Database, *BlockChain, error) {
// Initialize a fresh chain with only a genesis block
gspec := new(Genesis)
db, _ := ethdb.NewMemDatabase()
genesis := gspec.MustCommit(db)

blockchain, _ := NewBlockChain(db, params.AllProtocolChanges, ethash.NewFaker(),
new(event.TypeMux), vm.Config{})
// Create and inject the requested chain
if n == 0 {
return db, blockchain, nil
}
if full {
// Full block-chain requested
blocks := makeBlockChain(genesis, n, db, canonicalSeed)
_, err := blockchain.InsertChain(blocks)
return db, blockchain, err
}
// Header-only chain requested
headers := makeHeaderChain(genesis.Header(), n, db, canonicalSeed)
_, err := blockchain.InsertHeaderChain(headers, 1)
return db, blockchain, err
}
```

```
// makeHeaderChain creates a deterministic chain of headers rooted at parent.
func makeHeaderChain(parent *types.Header, n int, db ethdb.Database, seed int) []*types.Header
{
blocks := makeBlockChain(types.NewBlockWithHeader(parent), n, db, seed)
headers := make([]*types.Header, len(blocks))
for i, block := range blocks {
headers[i] = block.Header()
}
return headers
}

// makeBlockChain creates a deterministic chain of blocks rooted at parent.
func makeBlockChain(parent *types.Block, n int, db ethdb.Database, seed int) []*types.Block {
blocks, _ := GenerateChain(params.TestChainConfig, parent, db, n, func(i int, b *BlockGen) {
b.SetCoinbase(common.Address{0: byte(seed), 19: byte(i)})
})
return blocks
}

68:F:\git\coin\ethereum\go-ethereum\core\chain_makers_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"fmt"
"math/big"

"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/params"
)

func ExampleGenerateChain() {
var (
key1, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
```

```
291")
key2, _ =
crypto.HexToECDSA("8a1f9a8f95be41cd7ccb6168179afb4504aefe388d1e14474d32c45c72ce7b
7a")
key3, _ =
crypto.HexToECDSA("49a7b37aa6f6645917e7b807e9d1c00d4fa71f18343b0d4122a4d2df64dd6f
ee")
addr1   = crypto.PubkeyToAddress(key1.PublicKey)
addr2   = crypto.PubkeyToAddress(key2.PublicKey)
addr3   = crypto.PubkeyToAddress(key3.PublicKey)
db, _   = ethdb.NewMemDatabase()
)

// Ensure that key1 has some funds in the genesis block.
gspec := &Genesis{
Config: &params.ChainConfig{HomesteadBlock: new(big.Int)},
Alloc:  GenesisAlloc{addr1: {Balance: big.NewInt(1000000)}},
}
genesis := gspec.MustCommit(db)

// This call generates a chain of 5 blocks. The function runs for
// each block and adds different features to gen based on the
// block index.
signer := types.HomesteadSigner{}
chain, _ := GenerateChain(gspec.Config, genesis, db, 5, func(i int, gen *BlockGen) {
switch i {
case 0:
// In block 1, addr1 sends addr2 some ether.
tx, _ := types.SignTx(types.NewTransaction(gen.TxNonce(addr1), addr2, big.NewInt(10000),
bigTxGas, nil, nil), signer, key1)
gen.AddTx(tx)
case 1:
// In block 2, addr1 sends some more ether to addr2.
// addr2 passes it on to addr3.
tx1, _ := types.SignTx(types.NewTransaction(gen.TxNonce(addr1), addr2, big.NewInt(1000),
bigTxGas, nil, nil), signer, key1)
tx2, _ := types.SignTx(types.NewTransaction(gen.TxNonce(addr2), addr3, big.NewInt(1000),
bigTxGas, nil, nil), signer, key2)
gen.AddTx(tx1)
gen.AddTx(tx2)
case 2:
// Block 3 is empty but was mined by addr3.
```

```
gen.SetCoinbase(addr3)
gen.SetExtra([]byte("yeehaw"))
case 3:
// Block 4 includes blocks 2 and 3 as uncle headers (with modified extra data).
b2 := gen.PrevBlock(1).Header()
b2.Extra = []byte("foo")
gen.AddUncle(b2)
b3 := gen.PrevBlock(2).Header()
b3.Extra = []byte("foo")
gen.AddUncle(b3)
}
})

// Import the chain. This runs all block validation rules.
evmux := &event.TypeMux{}
blockchain, _ := NewBlockChain(db, gspec.Config, ethash.NewFaker(), evmux, vm.Config{})
if i, err := blockchain.InsertChain(chain); err != nil {
fmt.Printf("insert error (block %d): %v\n", chain[i].NumberU64(), err)
return
}

state, _ := blockchain.State()
fmt.Printf("last block: #%d\n", blockchain.CurrentBlock().Number())
fmt.Println("balance of addr1:", state.GetBalance(addr1))
fmt.Println("balance of addr2:", state.GetBalance(addr2))
fmt.Println("balance of addr3:", state.GetBalance(addr3))
// Output:
// last block: #5
// balance of addr1: 989000
// balance of addr2: 10000
// balance of addr3: 19687500000000001000
}

69:F:\git\coin\ethereum\go-ethereum\core\dao.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"bytes"
"fmt"
"math/big"
```

```
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/params"
)

// ValidateDAOHeaderExtraData validates the extra-data field of a block header to
// ensure it conforms to DAO hard-fork rules.
//
// DAO hard-fork extension to the header validity:
//   a) if the node is no-fork, do not accept blocks in the [fork, fork+10) range
//      with the fork specific extra-data set
//   b) if the node is pro-fork, require blocks in the specific range to have the
//      unique extra-data set.
func ValidateDAOHeaderExtraData(config *params.ChainConfig, header *types.Header) error {
// Short circuit validation if the node doesn't care about the DAO fork
if config.DAOForkBlock == nil {
return nil
}
// Make sure the block is within the fork's modified extra-data range
limit := new(big.Int).Add(config.DAOForkBlock, params.DAOForkExtraRange)
if header.Number.Cmp(config.DAOForkBlock) < 0 || header.Number.Cmp(limit) >= 0 {
return nil
}
// Depending whether we support or oppose the fork, validate the extra-data contents
if config.DAOForkSupport {
if !bytes.Equal(header.Extra, params.DAOForkBlockExtra) {
return fmt.Errorf("DAO pro-fork bad block extra-data: 0x%x", header.Extra)
}
} else {
if bytes.Equal(header.Extra, params.DAOForkBlockExtra) {
return fmt.Errorf("DAO no-fork bad block extra-data: 0x%x", header.Extra)
}
}
// All ok, header has the same extra-data we expect
return nil
}

// ApplyDAOHardFork modifies the state database according to the DAO hard-fork
// rules, transferring all balances of a set of DAO accounts to a single refund
// contract.
func ApplyDAOHardFork(statedb *state.StateDB) {
```

```
// Retrieve the contract to refund balances into
if !statedb.Exist(params.DAORefundContract) {
statedb.CreateAccount(params.DAORefundContract)
}

// Move every DAO account and extra-balance account funds into the refund contract
for _, addr := range params.DAODrainList() {
statedb.AddBalance(params.DAORefundContract, statedb.GetBalance(addr))
statedb.SetBalance(addr, new(big.Int))
}
}

70:F:\git\coin\ethereum\go-ethereum\core\dao_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"math/big"
"testing"

"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/params"
)

// Tests that DAO-fork enabled clients can properly filter out fork-commencing
// blocks based on their extradata fields.
func TestDAOForkRangeExtradata(t *testing.T) {
forkBlock := big.NewInt(32)

// Generate a common prefix for both pro-forkers and non-forkers
db, _ := ethdb.NewMemDatabase()
gspec := new(Genesis)
genesis := gspec.MustCommit(db)
prefix, _ := GenerateChain(params.TestChainConfig, genesis, db, int(forkBlock.Int64()-1), func(i
int, gen *BlockGen) {})

// Create the concurrent, conflicting two nodes
proDb, _ := ethdb.NewMemDatabase()
```

```
gspec.MustCommit(proDb)
proConf := &params.ChainConfig{HomesteadBlock: big.NewInt(0), DAOForkBlock: forkBlock,
DAOForkSupport: true}
proBc, _ := NewBlockChain(proDb, proConf, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})

conDb, _ := ethdb.NewMemDatabase()
gspec.MustCommit(conDb)
conConf := &params.ChainConfig{HomesteadBlock: big.NewInt(0), DAOForkBlock: forkBlock,
DAOForkSupport: false}
conBc, _ := NewBlockChain(conDb, conConf, ethash.NewFaker(), new(event.TypeMux),
vm.Config{})

if _, err := proBc.InsertChain(prefix); err != nil {
t.Fatalf("pro-fork: failed to import chain prefix: %v", err)
}
if _, err := conBc.InsertChain(prefix); err != nil {
t.Fatalf("con-fork: failed to import chain prefix: %v", err)
}
// Try to expand both pro-fork and non-fork chains iteratively with other camp's blocks
for i := int64(0); i < params.DAOForkExtraRange.Int64(); i++ {
// Create a pro-fork block, and try to feed into the no-fork chain
db, _ = ethdb.NewMemDatabase()
gspec.MustCommit(db)
bc, _ := NewBlockChain(db, conConf, ethash.NewFaker(), new(event.TypeMux), vm.Config{})

blocks := conBc.GetBlocksFromHash(conBc.CurrentBlock().Hash(),
int(conBc.CurrentBlock().NumberU64()))
for j := 0; j < len(blocks)/2; j++ {
blocks[j], blocks[len(blocks)-1-j] = blocks[len(blocks)-1-j], blocks[j]
}
if _, err := bc.InsertChain(blocks); err != nil {
t.Fatalf("failed to import contra-fork chain for expansion: %v", err)
}
blocks, _ = GenerateChain(proConf, conBc.CurrentBlock(), db, 1, func(i int, gen *BlockGen) {})
if _, err := conBc.InsertChain(blocks); err == nil {
t.Fatalf("contra-fork chain accepted pro-fork block: %v", blocks[0])
}
// Create a proper no-fork block for the contra-forker
blocks, _ = GenerateChain(conConf, conBc.CurrentBlock(), db, 1, func(i int, gen *BlockGen) {})
if _, err := conBc.InsertChain(blocks); err != nil {
t.Fatalf("contra-fork chain didn't accepted no-fork block: %v", err)
```

```
}
// Create a no-fork block, and try to feed into the pro-fork chain
db, _ = ethdb.NewMemDatabase()
gspec.MustCommit(db)
bc, _ = NewBlockChain(db, proConf, ethash.NewFaker(), new(event.TypeMux), vm.Config{})

blocks = proBc.GetBlocksFromHash(proBc.CurrentBlock().Hash(),
int(proBc.CurrentBlock().NumberU64()))
for j := 0; j < len(blocks)/2; j++ {
blocks[j], blocks[len(blocks)-1-j] = blocks[len(blocks)-1-j], blocks[j]
}
if _, err := bc.InsertChain(blocks); err != nil {
t.Fatalf("failed to import pro-fork chain for expansion: %v", err)
}
blocks, _ = GenerateChain(conConf, proBc.CurrentBlock(), db, 1, func(i int, gen *BlockGen) {})
if _, err := proBc.InsertChain(blocks); err == nil {
t.Fatalf("pro-fork chain accepted contra-fork block: %v", blocks[0])
}
// Create a proper pro-fork block for the pro-forker
blocks, _ = GenerateChain(proConf, proBc.CurrentBlock(), db, 1, func(i int, gen *BlockGen) {})
if _, err := proBc.InsertChain(blocks); err != nil {
t.Fatalf("pro-fork chain didn't accepted pro-fork block: %v", err)
}
}
// Verify that contra-forkers accept pro-fork extra-datas after forking finishes
db, _ = ethdb.NewMemDatabase()
gspec.MustCommit(db)
bc, _ := NewBlockChain(db, conConf, ethash.NewFaker(), new(event.TypeMux), vm.Config{})

blocks := conBc.GetBlocksFromHash(conBc.CurrentBlock().Hash(),
int(conBc.CurrentBlock().NumberU64()))
for j := 0; j < len(blocks)/2; j++ {
blocks[j], blocks[len(blocks)-1-j] = blocks[len(blocks)-1-j], blocks[j]
}
if _, err := bc.InsertChain(blocks); err != nil {
t.Fatalf("failed to import contra-fork chain for expansion: %v", err)
}
blocks, _ = GenerateChain(proConf, conBc.CurrentBlock(), db, 1, func(i int, gen *BlockGen) {})
if _, err := conBc.InsertChain(blocks); err != nil {
t.Fatalf("contra-fork chain didn't accept pro-fork block post-fork: %v", err)
}
// Verify that pro-forkers accept contra-fork extra-datas after forking finishes
```

```
db, _ = ethdb.NewMemDatabase()
gspec.MustCommit(db)
bc, _ = NewBlockChain(db, proConf, ethash.NewFaker(), new(event.TypeMux), vm.Config{})

blocks = proBc.GetBlocksFromHash(proBc.CurrentBlock().Hash(),
int(proBc.CurrentBlock().NumberU64()))
for j := 0; j < len(blocks)/2; j++ {
blocks[j], blocks[len(blocks)-1-j] = blocks[len(blocks)-1-j], blocks[j]
}
if _, err := bc.InsertChain(blocks); err != nil {
t.Fatalf("failed to import pro-fork chain for expansion: %v", err)
}
blocks, _ = GenerateChain(conConf, proBc.CurrentBlock(), db, 1, func(i int, gen *BlockGen) {})
if _, err := proBc.InsertChain(blocks); err != nil {
t.Fatalf("pro-fork chain didn't accept contra-fork block post-fork: %v", err)
}
}
```

71:F:\git\coin\ethereum\go-ethereum\core\database_util.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"bytes"
"encoding/binary"
"encoding/json"
"errors"
"fmt"
"math/big"
"sync"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/metrics"
"github.com/ethereum/go-ethereum/params"
"github.com/ethereum/go-ethereum/rlp"
)

var (
```

```
headHeaderKey = []byte("LastHeader")
headBlockKey  = []byte("LastBlock")
headFastKey   = []byte("LastFast")

headerPrefix       = []byte("h")   // headerPrefix + num (uint64 big endian) + hash -> header
tdSuffix           = []byte("t")   // headerPrefix + num (uint64 big endian) + hash + tdSuffix -> td
numSuffix          = []byte("n")   // headerPrefix + num (uint64 big endian) + numSuffix -> hash
blockHashPrefix    = []byte("H")   // blockHashPrefix + hash -> num (uint64 big endian)
bodyPrefix         = []byte("b")   // bodyPrefix + num (uint64 big endian) + hash -> block body
blockReceiptsPrefix = []byte("r")   // blockReceiptsPrefix + num (uint64 big endian) + hash -> block
receipts
preimagePrefix     = "secure-key-" // preimagePrefix + hash -> preimage

txMetaSuffix   = []byte{0x01}
receiptsPrefix = []byte("receipts-")

mipmapPre    = []byte("mipmap-log-bloom-")
MIPMapLevels = []uint64{1000000, 500000, 100000, 50000, 1000}

configPrefix = []byte("ethereum-config-") // config prefix for the db

// used by old (non-sequential keys) db, now only used for conversion
oldBlockPrefix        = []byte("block-")
oldHeaderSuffix       = []byte("-header")
oldTdSuffix           = []byte("-td") // headerPrefix + num (uint64 big endian) + hash + tdSuffix -> td
oldBodySuffix         = []byte("-body")
oldBlockNumPrefix     = []byte("block-num-")
oldBlockReceiptsPrefix = []byte("receipts-block-")
oldBlockHashPrefix    = []byte("block-hash-") // [deprecated by the header/block split, remove
eventually]

ErrChainConfigNotFound = errors.New("ChainConfig not found") // general config not found error

mipmapBloomMu sync.Mutex // protect against race condition when updating mipmap blooms

preimageCounter    = metrics.NewCounter("db/preimage/total")
preimageHitCounter = metrics.NewCounter("db/preimage/hits")
)

// encodeBlockNumber encodes a block number as big endian uint64
func encodeBlockNumber(number uint64) []byte {
enc := make([]byte, 8)
```

```
binary.BigEndian.PutUint64(enc, number)
return enc
}

// GetCanonicalHash retrieves a hash assigned to a canonical block number.
func GetCanonicalHash(db ethdb.Database, number uint64) common.Hash {
data, _ := db.Get(append(append(headerPrefix, encodeBlockNumber(number)...), numSuffix...))
if len(data) == 0 {
data, _ = db.Get(append(oldBlockNumPrefix, big.NewInt(int64(number)).Bytes()...))
if len(data) == 0 {
return common.Hash{}
}
}
return common.BytesToHash(data)
}

// missingNumber is returned by GetBlockNumber if no header with the
// given block hash has been stored in the database
const missingNumber = uint64(0xffffffffffffffff)

// GetBlockNumber returns the block number assigned to a block hash
// if the corresponding header is present in the database
func GetBlockNumber(db ethdb.Database, hash common.Hash) uint64 {
data, _ := db.Get(append(blockHashPrefix, hash.Bytes()...))
if len(data) != 8 {
data, _ := db.Get(append(append(oldBlockPrefix, hash.Bytes()...), oldHeaderSuffix...))
if len(data) == 0 {
return missingNumber
}
header := new(types.Header)
if err := rlp.Decode(bytes.NewReader(data), header); err != nil {
log.Crit("Failed to decode block header", "err", err)
}
return header.Number.Uint64()
}
return binary.BigEndian.Uint64(data)
}

// GetHeadHeaderHash retrieves the hash of the current canonical head block's
// header. The difference between this and GetHeadBlockHash is that whereas the
// last block hash is only updated upon a full block import, the last header
// hash is updated already at header import, allowing head tracking for the
```

```
// light synchronization mechanism.
func GetHeadHeaderHash(db ethdb.Database) common.Hash {
data, _ := db.Get(headHeaderKey)
if len(data) == 0 {
return common.Hash{}
}
return common.BytesToHash(data)
}

// GetHeadBlockHash retrieves the hash of the current canonical head block.
func GetHeadBlockHash(db ethdb.Database) common.Hash {
data, _ := db.Get(headBlockKey)
if len(data) == 0 {
return common.Hash{}
}
return common.BytesToHash(data)
}

// GetHeadFastBlockHash retrieves the hash of the current canonical head block during
// fast synchronization. The difference between this and GetHeadBlockHash is that
// whereas the last block hash is only updated upon a full block import, the last
// fast hash is updated when importing pre-processed blocks.
func GetHeadFastBlockHash(db ethdb.Database) common.Hash {
data, _ := db.Get(headFastKey)
if len(data) == 0 {
return common.Hash{}
}
return common.BytesToHash(data)
}

// GetHeaderRLP retrieves a block header in its raw RLP database encoding, or nil
// if the header's not found.
func GetHeaderRLP(db ethdb.Database, hash common.Hash, number uint64) rlp.RawValue {
data, _ := db.Get(append(append(headerPrefix, encodeBlockNumber(number)...), hash.Bytes()...))
if len(data) == 0 {
data, _ = db.Get(append(append(oldBlockPrefix, hash.Bytes()...), oldHeaderSuffix...))
}
return data
}

// GetHeader retrieves the block header corresponding to the hash, nil if none
// found.
```

```
func GetHeader(db ethdb.Database, hash common.Hash, number uint64) *types.Header {
data := GetHeaderRLP(db, hash, number)
if len(data) == 0 {
return nil
}
header := new(types.Header)
if err := rlp.Decode(bytes.NewReader(data), header); err != nil {
log.Error("Invalid block header RLP", "hash", hash, "err", err)
return nil
}
return header
}

// GetBodyRLP retrieves the block body (transactions and uncles) in RLP encoding.
func GetBodyRLP(db ethdb.Database, hash common.Hash, number uint64) rlp.RawValue {
data, _ := db.Get(append(append(bodyPrefix, encodeBlockNumber(number)...), hash.Bytes()...))
if len(data) == 0 {
data, _ = db.Get(append(append(oldBlockPrefix, hash.Bytes()...), oldBodySuffix...))
}
return data
}

// GetBody retrieves the block body (transactons, uncles) corresponding to the
// hash, nil if none found.
func GetBody(db ethdb.Database, hash common.Hash, number uint64) *types.Body {
data := GetBodyRLP(db, hash, number)
if len(data) == 0 {
return nil
}
body := new(types.Body)
if err := rlp.Decode(bytes.NewReader(data), body); err != nil {
log.Error("Invalid block body RLP", "hash", hash, "err", err)
return nil
}
return body
}

// GetTd retrieves a block's total difficulty corresponding to the hash, nil if
// none found.
func GetTd(db ethdb.Database, hash common.Hash, number uint64) *big.Int {
data, _ := db.Get(append(append(append(headerPrefix, encodeBlockNumber(number)...),
hash[:]...), tdSuffix...))
```

```
if len(data) == 0 {
data, _ = db.Get(append(append(oldBlockPrefix, hash.Bytes()...), oldTdSuffix...))
if len(data) == 0 {
return nil
}
}
td := new(big.Int)
if err := rlp.Decode(bytes.NewReader(data), td); err != nil {
log.Error("Invalid block total difficulty RLP", "hash", hash, "err", err)
return nil
}
return td
}

// GetBlock retrieves an entire block corresponding to the hash, assembling it
// back from the stored header and body. If either the header or body could not
// be retrieved nil is returned.
//
// Note, due to concurrent download of header and block body the header and thus
// canonical hash can be stored in the database but the body data not (yet).
func GetBlock(db ethdb.Database, hash common.Hash, number uint64) *types.Block {
// Retrieve the block header and body contents
header := GetHeader(db, hash, number)
if header == nil {
return nil
}
body := GetBody(db, hash, number)
if body == nil {
return nil
}
// Reassemble the block and return
return types.NewBlockWithHeader(header).WithBody(body.Transactions, body.Uncles)
}

// GetBlockReceipts retrieves the receipts generated by the transactions included
// in a block given by its hash.
func GetBlockReceipts(db ethdb.Database, hash common.Hash, number uint64) types.Receipts {
data, _ := db.Get(append(append(blockReceiptsPrefix, encodeBlockNumber(number)...),
hash[:]...))
if len(data) == 0 {
data, _ = db.Get(append(oldBlockReceiptsPrefix, hash.Bytes()...))
if len(data) == 0 {
```

```
return nil
}
}
storageReceipts := []*types.ReceiptForStorage{}
if err := rlp.DecodeBytes(data, &storageReceipts); err != nil {
log.Error("Invalid receipt array RLP", "hash", hash, "err", err)
return nil
}
receipts := make(types.Receipts, len(storageReceipts))
for i, receipt := range storageReceipts {
receipts[i] = (*types.Receipt)(receipt)
}
return receipts
}

// GetTransaction retrieves a specific transaction from the database, along with
// its added positional metadata.
func GetTransaction(db ethdb.Database, hash common.Hash) (*types.Transaction,
common.Hash, uint64, uint64) {
// Retrieve the transaction itself from the database
data, _ := db.Get(hash.Bytes())
if len(data) == 0 {
return nil, common.Hash{}, 0, 0
}
var tx types.Transaction
if err := rlp.DecodeBytes(data, &tx); err != nil {
return nil, common.Hash{}, 0, 0
}
// Retrieve the blockchain positional metadata
data, _ = db.Get(append(hash.Bytes(), txMetaSuffix...))
if len(data) == 0 {
return nil, common.Hash{}, 0, 0
}
var meta struct {
BlockHash  common.Hash
BlockIndex uint64
Index      uint64
}
if err := rlp.DecodeBytes(data, &meta); err != nil {
return nil, common.Hash{}, 0, 0
}
return &tx, meta.BlockHash, meta.BlockIndex, meta.Index
```

```
}

// GetReceipt returns a receipt by hash
func GetReceipt(db ethdb.Database, hash common.Hash) *types.Receipt {
data, _ := db.Get(append(receiptsPrefix, hash[:]...))
if len(data) == 0 {
return nil
}
var receipt types.ReceiptForStorage
err := rlp.DecodeBytes(data, &receipt)
if err != nil {
log.Error("Invalid receipt RLP", "hash", hash, "err", err)
}
return (*types.Receipt)(&receipt)
}

// WriteCanonicalHash stores the canonical hash for the given block number.
func WriteCanonicalHash(db ethdb.Database, hash common.Hash, number uint64) error {
key := append(append(headerPrefix, encodeBlockNumber(number)...), numSuffix...)
if err := db.Put(key, hash.Bytes()); err != nil {
log.Crit("Failed to store number to hash mapping", "err", err)
}
return nil
}

// WriteHeadHeaderHash stores the head header's hash.
func WriteHeadHeaderHash(db ethdb.Database, hash common.Hash) error {
if err := db.Put(headHeaderKey, hash.Bytes()); err != nil {
log.Crit("Failed to store last header's hash", "err", err)
}
return nil
}

// WriteHeadBlockHash stores the head block's hash.
func WriteHeadBlockHash(db ethdb.Database, hash common.Hash) error {
if err := db.Put(headBlockKey, hash.Bytes()); err != nil {
log.Crit("Failed to store last block's hash", "err", err)
}
return nil
}

// WriteHeadFastBlockHash stores the fast head block's hash.
```

```
func WriteHeadFastBlockHash(db ethdb.Database, hash common.Hash) error {
if err := db.Put(headFastKey, hash.Bytes()); err != nil {
log.Crit("Failed to store last fast block's hash", "err", err)
}
return nil
}

// WriteHeader serializes a block header into the database.
func WriteHeader(db ethdb.Database, header *types.Header) error {
data, err := rlp.EncodeToBytes(header)
if err != nil {
return err
}
hash := header.Hash().Bytes()
num := header.Number.Uint64()
encNum := encodeBlockNumber(num)
key := append(blockHashPrefix, hash...)
if err := db.Put(key, encNum); err != nil {
log.Crit("Failed to store hash to number mapping", "err", err)
}
key = append(append(headerPrefix, encNum...), hash...)
if err := db.Put(key, data); err != nil {
log.Crit("Failed to store header", "err", err)
}
return nil
}

// WriteBody serializes the body of a block into the database.
func WriteBody(db ethdb.Database, hash common.Hash, number uint64, body *types.Body) error
{
data, err := rlp.EncodeToBytes(body)
if err != nil {
return err
}
return WriteBodyRLP(db, hash, number, data)
}

// WriteBodyRLP writes a serialized body of a block into the database.
func WriteBodyRLP(db ethdb.Database, hash common.Hash, number uint64, rlp rlp.RawValue)
error {
key := append(append(bodyPrefix, encodeBlockNumber(number)...), hash.Bytes()...)
if err := db.Put(key, rlp); err != nil {
```

```
log.Crit("Failed to store block body", "err", err)
}
return nil
}

// WriteTd serializes the total difficulty of a block into the database.
func WriteTd(db ethdb.Database, hash common.Hash, number uint64, td *big.Int) error {
data, err := rlp.EncodeToBytes(td)
if err != nil {
return err
}
key := append(append(append(headerPrefix, encodeBlockNumber(number)...), hash.Bytes()...),
tdSuffix...)
if err := db.Put(key, data); err != nil {
log.Crit("Failed to store block total difficulty", "err", err)
}
return nil
}

// WriteBlock serializes a block into the database, header and body separately.
func WriteBlock(db ethdb.Database, block *types.Block) error {
// Store the body first to retain database consistency
if err := WriteBody(db, block.Hash(), block.NumberU64(), block.Body()); err != nil {
return err
}
// Store the header too, signaling full block ownership
if err := WriteHeader(db, block.Header()); err != nil {
return err
}
return nil
}

// WriteBlockReceipts stores all the transaction receipts belonging to a block
// as a single receipt slice. This is used during chain reorganisations for
// rescheduling dropped transactions.
func WriteBlockReceipts(db ethdb.Database, hash common.Hash, number uint64, receipts
types.Receipts) error {
// Convert the receipts into their storage form and serialize them
storageReceipts := make([]*types.ReceiptForStorage, len(receipts))
for i, receipt := range receipts {
storageReceipts[i] = (*types.ReceiptForStorage)(receipt)
}
```

```
bytes, err := rlp.EncodeToBytes(storageReceipts)
if err != nil {
return err
}
// Store the flattened receipt slice
key := append(append(blockReceiptsPrefix, encodeBlockNumber(number)...), hash.Bytes()...)
if err := db.Put(key, bytes); err != nil {
log.Crit("Failed to store block receipts", "err", err)
}
return nil
}

// WriteTransactions stores the transactions associated with a specific block
// into the given database. Beside writing the transaction, the function also
// stores a metadata entry along with the transaction, detailing the position
// of this within the blockchain.
func WriteTransactions(db ethdb.Database, block *types.Block) error {
batch := db.NewBatch()

// Iterate over each transaction and encode it with its metadata
for i, tx := range block.Transactions() {
// Encode and queue up the transaction for storage
data, err := rlp.EncodeToBytes(tx)
if err != nil {
return err
}
if err = batch.Put(tx.Hash().Bytes(), data); err != nil {
return err
}
// Encode and queue up the transaction metadata for storage
meta := struct {
BlockHash  common.Hash
BlockIndex uint64
Index      uint64
}{
BlockHash:  block.Hash(),
BlockIndex: block.NumberU64(),
Index:      uint64(i),
}
data, err = rlp.EncodeToBytes(meta)
if err != nil {
return err
```

```
}
if err := batch.Put(append(tx.Hash().Bytes(), txMetaSuffix...), data); err != nil {
return err
}
}
// Write the scheduled data into the database
if err := batch.Write(); err != nil {
log.Crit("Failed to store transactions", "err", err)
}
return nil
}

// WriteReceipt stores a single transaction receipt into the database.
func WriteReceipt(db ethdb.Database, receipt *types.Receipt) error {
storageReceipt := (*types.ReceiptForStorage)(receipt)
data, err := rlp.EncodeToBytes(storageReceipt)
if err != nil {
return err
}
return db.Put(append(receiptsPrefix, receipt.TxHash.Bytes()...), data)
}

// WriteReceipts stores a batch of transaction receipts into the database.
func WriteReceipts(db ethdb.Database, receipts types.Receipts) error {
batch := db.NewBatch()

// Iterate over all the receipts and queue them for database injection
for _, receipt := range receipts {
storageReceipt := (*types.ReceiptForStorage)(receipt)
data, err := rlp.EncodeToBytes(storageReceipt)
if err != nil {
return err
}
if err := batch.Put(append(receiptsPrefix, receipt.TxHash.Bytes()...), data); err != nil {
return err
}
}
// Write the scheduled data into the database
if err := batch.Write(); err != nil {
log.Crit("Failed to store receipts", "err", err)
}
return nil
```

```
}

// DeleteCanonicalHash removes the number to hash canonical mapping.
func DeleteCanonicalHash(db ethdb.Database, number uint64) {
db.Delete(append(append(headerPrefix, encodeBlockNumber(number)...), numSuffix...))
}

// DeleteHeader removes all block header data associated with a hash.
func DeleteHeader(db ethdb.Database, hash common.Hash, number uint64) {
db.Delete(append(blockHashPrefix, hash.Bytes()...))
db.Delete(append(append(headerPrefix, encodeBlockNumber(number)...), hash.Bytes()...))
}

// DeleteBody removes all block body data associated with a hash.
func DeleteBody(db ethdb.Database, hash common.Hash, number uint64) {
db.Delete(append(append(bodyPrefix, encodeBlockNumber(number)...), hash.Bytes()...))
}

// DeleteTd removes all block total difficulty data associated with a hash.
func DeleteTd(db ethdb.Database, hash common.Hash, number uint64) {
db.Delete(append(append(append(headerPrefix, encodeBlockNumber(number)...),
hash.Bytes()...), tdSuffix...))
}

// DeleteBlock removes all block data associated with a hash.
func DeleteBlock(db ethdb.Database, hash common.Hash, number uint64) {
DeleteBlockReceipts(db, hash, number)
DeleteHeader(db, hash, number)
DeleteBody(db, hash, number)
DeleteTd(db, hash, number)
}

// DeleteBlockReceipts removes all receipt data associated with a block hash.
func DeleteBlockReceipts(db ethdb.Database, hash common.Hash, number uint64) {
db.Delete(append(append(blockReceiptsPrefix, encodeBlockNumber(number)...), hash.Bytes()...))
}

// DeleteTransaction removes all transaction data associated with a hash.
func DeleteTransaction(db ethdb.Database, hash common.Hash) {
db.Delete(hash.Bytes())
db.Delete(append(hash.Bytes(), txMetaSuffix...))
}
```

```
// DeleteReceipt removes all receipt data associated with a transaction hash.
func DeleteReceipt(db ethdb.Database, hash common.Hash) {
db.Delete(append(receiptsPrefix, hash.Bytes()...))
}

// returns a formatted MIP mapped key by adding prefix, canonical number and level
//
// ex. fn(98, 1000) = (prefix || 1000 || 0)
func mipmapKey(num, level uint64) []byte {
lkey := make([]byte, 8)
binary.BigEndian.PutUint64(lkey, level)
key := new(big.Int).SetUint64(num / level * level)

return append(mipmapPre, append(lkey, key.Bytes()...)...)
}

// WriteMipmapBloom writes each address included in the receipts' logs to the
// MIP bloom bin.
func WriteMipmapBloom(db ethdb.Database, number uint64, receipts types.Receipts) error {
mipmapBloomMu.Lock()
defer mipmapBloomMu.Unlock()

batch := db.NewBatch()
for _, level := range MIPMapLevels {
key := mipmapKey(number, level)
bloomDat, _ := db.Get(key)
bloom := types.BytesToBloom(bloomDat)
for _, receipt := range receipts {
for _, log := range receipt.Logs {
bloom.Add(log.Address.Big())
}
}
batch.Put(key, bloom.Bytes())
}
if err := batch.Write(); err != nil {
return fmt.Errorf("mipmap write fail for: %d: %v", number, err)
}
return nil
}

// GetMipmapBloom returns a bloom filter using the number and level as input
```

```
// parameters. For available levels see MIPMapLevels.
func GetMipmapBloom(db ethdb.Database, number, level uint64) types.Bloom {
bloomDat, _ := db.Get(mipmapKey(number, level))
return types.BytesToBloom(bloomDat)
}

// PreimageTable returns a Database instance with the key prefix for preimage entries.
func PreimageTable(db ethdb.Database) ethdb.Database {
return ethdb.NewTable(db, preimagePrefix)
}

// WritePreimages writes the provided set of preimages to the database. `number` is the
// current block number, and is used for debug messages only.
func WritePreimages(db ethdb.Database, number uint64, preimages map[common.Hash][]byte)
error {
table := PreimageTable(db)
batch := table.NewBatch()
hitCount := 0
for hash, preimage := range preimages {
if _, err := table.Get(hash.Bytes()); err != nil {
batch.Put(hash.Bytes(), preimage)
hitCount++
}
}
preimageCounter.Inc(int64(len(preimages)))
preimageHitCounter.Inc(int64(hitCount))
if hitCount > 0 {
if err := batch.Write(); err != nil {
return fmt.Errorf("preimage write fail for block %d: %v", number, err)
}
}
return nil
}

// GetBlockChainVersion reads the version number from db.
func GetBlockChainVersion(db ethdb.Database) int {
var vsn uint
enc, _ := db.Get([]byte("BlockchainVersion"))
rlp.DecodeBytes(enc, &vsn)
return int(vsn)
}
```

```
// WriteBlockChainVersion writes vsn as the version number to db.
func WriteBlockChainVersion(db ethdb.Database, vsn int) {
enc, _ := rlp.EncodeToBytes(uint(vsn))
db.Put([]byte("BlockchainVersion"), enc)
}

// WriteChainConfig writes the chain config settings to the database.
func WriteChainConfig(db ethdb.Database, hash common.Hash, cfg *params.ChainConfig) error {
// short circuit and ignore if nil config. GetChainConfig
// will return a default.
if cfg == nil {
return nil
}

jsonChainConfig, err := json.Marshal(cfg)
if err != nil {
return err
}

return db.Put(append(configPrefix, hash[:]...), jsonChainConfig)
}

// GetChainConfig will fetch the network settings based on the given hash.
func GetChainConfig(db ethdb.Database, hash common.Hash) (*params.ChainConfig, error) {
jsonChainConfig, _ := db.Get(append(configPrefix, hash[:]...))
if len(jsonChainConfig) == 0 {
return nil, ErrChainConfigNotFound
}

var config params.ChainConfig
if err := json.Unmarshal(jsonChainConfig, &config); err != nil {
return nil, err
}

return &config, nil
}

// FindCommonAncestor returns the last common ancestor of two block headers
func FindCommonAncestor(db ethdb.Database, a, b *types.Header) *types.Header {
for bn := b.Number.Uint64(); a.Number.Uint64() > bn; {
a = GetHeader(db, a.ParentHash, a.Number.Uint64()-1)
if a == nil {
```

```
return nil
}
}
for an := a.Number.Uint64(); an < b.Number.Uint64(); {
b = GetHeader(db, b.ParentHash, b.Number.Uint64()-1)
if b == nil {
return nil
}
}
for a.Hash() != b.Hash() {
a = GetHeader(db, a.ParentHash, a.Number.Uint64()-1)
if a == nil {
return nil
}
b = GetHeader(db, b.ParentHash, b.Number.Uint64()-1)
if b == nil {
return nil
}
}
return a
}
```

72:F:\git\coin\ethereum\go-ethereum\core\database_util_test.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"bytes"
"io/ioutil"
"math/big"
"os"
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/crypto/sha3"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/params"
"github.com/ethereum/go-ethereum/rlp"
)
```

```
// Tests block header storage and retrieval operations.
func TestHeaderStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

// Create a test header to move around the database and make sure it's really new
header := &types.Header{Number: big.NewInt(42), Extra: []byte("test header")}
if entry := GetHeader(db, header.Hash(), header.Number.Uint64()); entry != nil {
t.Fatalf("Non existent header returned: %v", entry)
}
// Write and verify the header in the database
if err := WriteHeader(db, header); err != nil {
t.Fatalf("Failed to write header into database: %v", err)
}
if entry := GetHeader(db, header.Hash(), header.Number.Uint64()); entry == nil {
t.Fatalf("Stored header not found")
} else if entry.Hash() != header.Hash() {
t.Fatalf("Retrieved header mismatch: have %v, want %v", entry, header)
}
if entry := GetHeaderRLP(db, header.Hash(), header.Number.Uint64()); entry == nil {
t.Fatalf("Stored header RLP not found")
} else {
hasher := sha3.NewKeccak256()
hasher.Write(entry)

if hash := common.BytesToHash(hasher.Sum(nil)); hash != header.Hash() {
t.Fatalf("Retrieved RLP header mismatch: have %v, want %v", entry, header)
}
}
// Delete the header and verify the execution
DeleteHeader(db, header.Hash(), header.Number.Uint64())
if entry := GetHeader(db, header.Hash(), header.Number.Uint64()); entry != nil {
t.Fatalf("Deleted header returned: %v", entry)
}
}

// Tests block body storage and retrieval operations.
func TestBodyStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

// Create a test body to move around the database and make sure it's really new
body := &types.Body{Uncles: []*types.Header{{Extra: []byte("test header")}}}
```

```
hasher := sha3.NewKeccak256()
rlp.Encode(hasher, body)
hash := common.BytesToHash(hasher.Sum(nil))

if entry := GetBody(db, hash, 0); entry != nil {
t.Fatalf("Non existent body returned: %v", entry)
}
// Write and verify the body in the database
if err := WriteBody(db, hash, 0, body); err != nil {
t.Fatalf("Failed to write body into database: %v", err)
}
if entry := GetBody(db, hash, 0); entry == nil {
t.Fatalf("Stored body not found")
} else if types.DeriveSha(types.Transactions(entry.Transactions)) !=
types.DeriveSha(types.Transactions(body.Transactions)) || types.CalcUncleHash(entry.Uncles) !=
types.CalcUncleHash(body.Uncles) {
t.Fatalf("Retrieved body mismatch: have %v, want %v", entry, body)
}
if entry := GetBodyRLP(db, hash, 0); entry == nil {
t.Fatalf("Stored body RLP not found")
} else {
hasher := sha3.NewKeccak256()
hasher.Write(entry)

if calc := common.BytesToHash(hasher.Sum(nil)); calc != hash {
t.Fatalf("Retrieved RLP body mismatch: have %v, want %v", entry, body)
}
}
// Delete the body and verify the execution
DeleteBody(db, hash, 0)
if entry := GetBody(db, hash, 0); entry != nil {
t.Fatalf("Deleted body returned: %v", entry)
}
}

// Tests block storage and retrieval operations.
func TestBlockStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

// Create a test block to move around the database and make sure it's really new
block := types.NewBlockWithHeader(&types.Header{
```

```
Extra:       []byte("test block"),
UncleHash:   types.EmptyUncleHash,
TxHash:      types.EmptyRootHash,
ReceiptHash: types.EmptyRootHash,
})
if entry := GetBlock(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Non existent block returned: %v", entry)
}
if entry := GetHeader(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Non existent header returned: %v", entry)
}
if entry := GetBody(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Non existent body returned: %v", entry)
}
// Write and verify the block in the database
if err := WriteBlock(db, block); err != nil {
t.Fatalf("Failed to write block into database: %v", err)
}
if entry := GetBlock(db, block.Hash(), block.NumberU64()); entry == nil {
t.Fatalf("Stored block not found")
} else if entry.Hash() != block.Hash() {
t.Fatalf("Retrieved block mismatch: have %v, want %v", entry, block)
}
if entry := GetHeader(db, block.Hash(), block.NumberU64()); entry == nil {
t.Fatalf("Stored header not found")
} else if entry.Hash() != block.Header().Hash() {
t.Fatalf("Retrieved header mismatch: have %v, want %v", entry, block.Header())
}
if entry := GetBody(db, block.Hash(), block.NumberU64()); entry == nil {
t.Fatalf("Stored body not found")
} else if types.DeriveSha(types.Transactions(entry.Transactions)) !=
types.DeriveSha(block.Transactions()) || types.CalcUncleHash(entry.Uncles) !=
types.CalcUncleHash(block.Uncles()) {
t.Fatalf("Retrieved body mismatch: have %v, want %v", entry, block.Body())
}
// Delete the block and verify the execution
DeleteBlock(db, block.Hash(), block.NumberU64())
if entry := GetBlock(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Deleted block returned: %v", entry)
}
if entry := GetHeader(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Deleted header returned: %v", entry)
```

```
}
if entry := GetBody(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Deleted body returned: %v", entry)
}
}

// Tests that partial block contents don't get reassembled into full blocks.
func TestPartialBlockStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()
block := types.NewBlockWithHeader(&types.Header{
Extra:       []byte("test block"),
UncleHash:   types.EmptyUncleHash,
TxHash:      types.EmptyRootHash,
ReceiptHash: types.EmptyRootHash,
})
// Store a header and check that it's not recognized as a block
if err := WriteHeader(db, block.Header()); err != nil {
t.Fatalf("Failed to write header into database: %v", err)
}
if entry := GetBlock(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Non existent block returned: %v", entry)
}
DeleteHeader(db, block.Hash(), block.NumberU64())

// Store a body and check that it's not recognized as a block
if err := WriteBody(db, block.Hash(), block.NumberU64(), block.Body()); err != nil {
t.Fatalf("Failed to write body into database: %v", err)
}
if entry := GetBlock(db, block.Hash(), block.NumberU64()); entry != nil {
t.Fatalf("Non existent block returned: %v", entry)
}
DeleteBody(db, block.Hash(), block.NumberU64())

// Store a header and a body separately and check reassembly
if err := WriteHeader(db, block.Header()); err != nil {
t.Fatalf("Failed to write header into database: %v", err)
}
if err := WriteBody(db, block.Hash(), block.NumberU64(), block.Body()); err != nil {
t.Fatalf("Failed to write body into database: %v", err)
}
if entry := GetBlock(db, block.Hash(), block.NumberU64()); entry == nil {
t.Fatalf("Stored block not found")
```

```
} else if entry.Hash() != block.Hash() {
t.Fatalf("Retrieved block mismatch: have %v, want %v", entry, block)
}
}

// Tests block total difficulty storage and retrieval operations.
func TestTdStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

// Create a test TD to move around the database and make sure it's really new
hash, td := common.Hash{}, big.NewInt(314)
if entry := GetTd(db, hash, 0); entry != nil {
t.Fatalf("Non existent TD returned: %v", entry)
}
// Write and verify the TD in the database
if err := WriteTd(db, hash, 0, td); err != nil {
t.Fatalf("Failed to write TD into database: %v", err)
}
if entry := GetTd(db, hash, 0); entry == nil {
t.Fatalf("Stored TD not found")
} else if entry.Cmp(td) != 0 {
t.Fatalf("Retrieved TD mismatch: have %v, want %v", entry, td)
}
// Delete the TD and verify the execution
DeleteTd(db, hash, 0)
if entry := GetTd(db, hash, 0); entry != nil {
t.Fatalf("Deleted TD returned: %v", entry)
}
}

// Tests that canonical numbers can be mapped to hashes and retrieved.
func TestCanonicalMappingStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

// Create a test canonical number and assinged hash to move around
hash, number := common.Hash{0: 0xff}, uint64(314)
if entry := GetCanonicalHash(db, number); entry != (common.Hash{}) {
t.Fatalf("Non existent canonical mapping returned: %v", entry)
}
// Write and verify the TD in the database
if err := WriteCanonicalHash(db, hash, number); err != nil {
t.Fatalf("Failed to write canonical mapping into database: %v", err)
```

```
}
if entry := GetCanonicalHash(db, number); entry == (common.Hash{}) {
t.Fatalf("Stored canonical mapping not found")
} else if entry != hash {
t.Fatalf("Retrieved canonical mapping mismatch: have %v, want %v", entry, hash)
}
// Delete the TD and verify the execution
DeleteCanonicalHash(db, number)
if entry := GetCanonicalHash(db, number); entry != (common.Hash{}) {
t.Fatalf("Deleted canonical mapping returned: %v", entry)
}
}

// Tests that head headers and head blocks can be assigned, individually.
func TestHeadStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

blockHead := types.NewBlockWithHeader(&types.Header{Extra: []byte("test block header")})
blockFull := types.NewBlockWithHeader(&types.Header{Extra: []byte("test block full")})
blockFast := types.NewBlockWithHeader(&types.Header{Extra: []byte("test block fast")})

// Check that no head entries are in a pristine database
if entry := GetHeadHeaderHash(db); entry != (common.Hash{}) {
t.Fatalf("Non head header entry returned: %v", entry)
}
if entry := GetHeadBlockHash(db); entry != (common.Hash{}) {
t.Fatalf("Non head block entry returned: %v", entry)
}
if entry := GetHeadFastBlockHash(db); entry != (common.Hash{}) {
t.Fatalf("Non fast head block entry returned: %v", entry)
}
// Assign separate entries for the head header and block
if err := WriteHeadHeaderHash(db, blockHead.Hash()); err != nil {
t.Fatalf("Failed to write head header hash: %v", err)
}
if err := WriteHeadBlockHash(db, blockFull.Hash()); err != nil {
t.Fatalf("Failed to write head block hash: %v", err)
}
if err := WriteHeadFastBlockHash(db, blockFast.Hash()); err != nil {
t.Fatalf("Failed to write fast head block hash: %v", err)
}
// Check that both heads are present, and different (i.e. two heads maintained)
```

```
if entry := GetHeadHeaderHash(db); entry != blockHead.Hash() {
t.Fatalf("Head header hash mismatch: have %v, want %v", entry, blockHead.Hash())
}
if entry := GetHeadBlockHash(db); entry != blockFull.Hash() {
t.Fatalf("Head block hash mismatch: have %v, want %v", entry, blockFull.Hash())
}
if entry := GetHeadFastBlockHash(db); entry != blockFast.Hash() {
t.Fatalf("Fast head block hash mismatch: have %v, want %v", entry, blockFast.Hash())
}
}

// Tests that transactions and associated metadata can be stored and retrieved.
func TestTransactionStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

tx1 := types.NewTransaction(1, common.BytesToAddress([]byte{0x11}), big.NewInt(111),
big.NewInt(1111), big.NewInt(11111), []byte{0x11, 0x11, 0x11})
tx2 := types.NewTransaction(2, common.BytesToAddress([]byte{0x22}), big.NewInt(222),
big.NewInt(2222), big.NewInt(22222), []byte{0x22, 0x22, 0x22})
tx3 := types.NewTransaction(3, common.BytesToAddress([]byte{0x33}), big.NewInt(333),
big.NewInt(3333), big.NewInt(33333), []byte{0x33, 0x33, 0x33})
txs := []*types.Transaction{tx1, tx2, tx3}

block := types.NewBlock(&types.Header{Number: big.NewInt(314)}, txs, nil, nil)

// Check that no transactions entries are in a pristine database
for i, tx := range txs {
if txn, _, _, _ := GetTransaction(db, tx.Hash()); txn != nil {
t.Fatalf("tx #%d [%x]: non existent transaction returned: %v", i, tx.Hash(), txn)
}
}
// Insert all the transactions into the database, and verify contents
if err := WriteTransactions(db, block); err != nil {
t.Fatalf("failed to write transactions: %v", err)
}
for i, tx := range txs {
if txn, hash, number, index := GetTransaction(db, tx.Hash()); txn == nil {
t.Fatalf("tx #%d [%x]: transaction not found", i, tx.Hash())
} else {
if hash != block.Hash() || number != block.NumberU64() || index != uint64(i) {
t.Fatalf("tx #%d [%x]: positional metadata mismatch: have %x/%d/%d, want %x/%v/%v", i,
tx.Hash(), hash, number, index, block.Hash(), block.NumberU64(), i)
```

```
}
if tx.String() != txn.String() {
t.Fatalf("tx #%d [%x]: transaction mismatch: have %v, want %v", i, tx.Hash(), txn, tx)
}
}
}
// Delete the transactions and check purge
for i, tx := range txs {
DeleteTransaction(db, tx.Hash())
if txn, _, _, _ := GetTransaction(db, tx.Hash()); txn != nil {
t.Fatalf("tx #%d [%x]: deleted transaction returned: %v", i, tx.Hash(), txn)
}
}
}

// Tests that receipts can be stored and retrieved.
func TestReceiptStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

receipt1 := &types.Receipt{
PostState:         []byte{0x01},
CumulativeGasUsed: big.NewInt(1),
Logs: []*types.Log{
{Address: common.BytesToAddress([]byte{0x11})},
{Address: common.BytesToAddress([]byte{0x01, 0x11})},
},
TxHash:          common.BytesToHash([]byte{0x11, 0x11}),
ContractAddress: common.BytesToAddress([]byte{0x01, 0x11, 0x11}),
GasUsed:         big.NewInt(111111),
}
receipt2 := &types.Receipt{
PostState:         []byte{0x02},
CumulativeGasUsed: big.NewInt(2),
Logs: []*types.Log{
{Address: common.BytesToAddress([]byte{0x22})},
{Address: common.BytesToAddress([]byte{0x02, 0x22})},
},
TxHash:          common.BytesToHash([]byte{0x22, 0x22}),
ContractAddress: common.BytesToAddress([]byte{0x02, 0x22, 0x22}),
GasUsed:         big.NewInt(222222),
}
receipts := []*types.Receipt{receipt1, receipt2}
```

```
// Check that no receipt entries are in a pristine database
for i, receipt := range receipts {
if r := GetReceipt(db, receipt.TxHash); r != nil {
t.Fatalf("receipt #%d [%x]: non existent receipt returned: %v", i, receipt.TxHash, r)
}
}
// Insert all the receipts into the database, and verify contents
if err := WriteReceipts(db, receipts); err != nil {
t.Fatalf("failed to write receipts: %v", err)
}
for i, receipt := range receipts {
if r := GetReceipt(db, receipt.TxHash); r == nil {
t.Fatalf("receipt #%d [%x]: receipt not found", i, receipt.TxHash)
} else {
rlpHave, _ := rlp.EncodeToBytes(r)
rlpWant, _ := rlp.EncodeToBytes(receipt)

if !bytes.Equal(rlpHave, rlpWant) {
t.Fatalf("receipt #%d [%x]: receipt mismatch: have %v, want %v", i, receipt.TxHash, r, receipt)
}
}
}
// Delete the receipts and check purge
for i, receipt := range receipts {
DeleteReceipt(db, receipt.TxHash)
if r := GetReceipt(db, receipt.TxHash); r != nil {
t.Fatalf("receipt #%d [%x]: deleted receipt returned: %v", i, receipt.TxHash, r)
}
}
}

// Tests that receipts associated with a single block can be stored and retrieved.
func TestBlockReceiptStorage(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

receipt1 := &types.Receipt{
PostState:         []byte{0x01},
CumulativeGasUsed: big.NewInt(1),
Logs: []*types.Log{
{Address: common.BytesToAddress([]byte{0x11})},
{Address: common.BytesToAddress([]byte{0x01, 0x11})},
```

```
},
TxHash:          common.BytesToHash([]byte{0x11, 0x11}),
ContractAddress: common.BytesToAddress([]byte{0x01, 0x11, 0x11}),
GasUsed:         big.NewInt(111111),
}
receipt2 := &types.Receipt{
PostState:         []byte{0x02},
CumulativeGasUsed: big.NewInt(2),
Logs: []*types.Log{
{Address: common.BytesToAddress([]byte{0x22})},
{Address: common.BytesToAddress([]byte{0x02, 0x22})},
},
TxHash:          common.BytesToHash([]byte{0x22, 0x22}),
ContractAddress: common.BytesToAddress([]byte{0x02, 0x22, 0x22}),
GasUsed:         big.NewInt(222222),
}
receipts := []*types.Receipt{receipt1, receipt2}

// Check that no receipt entries are in a pristine database
hash := common.BytesToHash([]byte{0x03, 0x14})
if rs := GetBlockReceipts(db, hash, 0); len(rs) != 0 {
t.Fatalf("non existent receipts returned: %v", rs)
}
// Insert the receipt slice into the database and check presence
if err := WriteBlockReceipts(db, hash, 0, receipts); err != nil {
t.Fatalf("failed to write block receipts: %v", err)
}
if rs := GetBlockReceipts(db, hash, 0); len(rs) == 0 {
t.Fatalf("no receipts returned")
} else {
for i := 0; i < len(receipts); i++ {
rlpHave, _ := rlp.EncodeToBytes(rs[i])
rlpWant, _ := rlp.EncodeToBytes(receipts[i])

if !bytes.Equal(rlpHave, rlpWant) {
t.Fatalf("receipt #%d: receipt mismatch: have %v, want %v", i, rs[i], receipts[i])
}
}
}
// Delete the receipt slice and check purge
DeleteBlockReceipts(db, hash, 0)
if rs := GetBlockReceipts(db, hash, 0); len(rs) != 0 {
```

```
t.Fatalf("deleted receipts returned: %v", rs)
}
}

func TestMipmapBloom(t *testing.T) {
db, _ := ethdb.NewMemDatabase()

receipt1 := new(types.Receipt)
receipt1.Logs = []*types.Log{
{Address: common.BytesToAddress([]byte("test"))},
{Address: common.BytesToAddress([]byte("address"))},
}
receipt2 := new(types.Receipt)
receipt2.Logs = []*types.Log{
{Address: common.BytesToAddress([]byte("test"))},
{Address: common.BytesToAddress([]byte("address1"))},
}

WriteMipmapBloom(db, 1, types.Receipts{receipt1})
WriteMipmapBloom(db, 2, types.Receipts{receipt2})

for _, level := range MIPMapLevels {
bloom := GetMipmapBloom(db, 2, level)
if !bloom.Test(new(big.Int).SetBytes([]byte("address1"))) {
t.Error("expected test to be included on level:", level)
}
}

// reset
db, _ = ethdb.NewMemDatabase()
receipt := new(types.Receipt)
receipt.Logs = []*types.Log{
{Address: common.BytesToAddress([]byte("test"))},
}
WriteMipmapBloom(db, 999, types.Receipts{receipt1})

receipt = new(types.Receipt)
receipt.Logs = []*types.Log{
{Address: common.BytesToAddress([]byte("test 1"))},
}
WriteMipmapBloom(db, 1000, types.Receipts{receipt})
```

```
bloom := GetMipmapBloom(db, 1000, 1000)
if bloom.TestBytes([]byte("test")) {
t.Error("test should not have been included")
}
}

func TestMipmapChain(t *testing.T) {
dir, err := ioutil.TempDir("", "mipmap")
if err != nil {
t.Fatal(err)
}
defer os.RemoveAll(dir)

var (
db, _   = ethdb.NewLDBDatabase(dir, 0, 0)
key1, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")
addr    = crypto.PubkeyToAddress(key1.PublicKey)
addr2   = common.BytesToAddress([]byte("jeff"))

hash1 = common.BytesToHash([]byte("topic1"))
)
defer db.Close()

gspec := &Genesis{
Config: params.TestChainConfig,
Alloc:  GenesisAlloc{addr: {Balance: big.NewInt(1000000)}},
}
genesis := gspec.MustCommit(db)
chain, receipts := GenerateChain(params.TestChainConfig, genesis, db, 1010, func(i int, gen
*BlockGen) {
var receipts types.Receipts
switch i {
case 1:
receipt := types.NewReceipt(nil, new(big.Int))
receipt.Logs = []*types.Log{{Address: addr, Topics: []common.Hash{hash1}}}
gen.AddUncheckedReceipt(receipt)
receipts = types.Receipts{receipt}
case 1000:
receipt := types.NewReceipt(nil, new(big.Int))
receipt.Logs = []*types.Log{{Address: addr2}}
```

```
gen.AddUncheckedReceipt(receipt)
receipts = types.Receipts{receipt}

}

// store the receipts
err := WriteReceipts(db, receipts)
if err != nil {
t.Fatal(err)
}
WriteMipmapBloom(db, uint64(i+1), receipts)
})
for i, block := range chain {
WriteBlock(db, block)
if err := WriteCanonicalHash(db, block.Hash(), block.NumberU64()); err != nil {
t.Fatalf("failed to insert block number: %v", err)
}
if err := WriteHeadBlockHash(db, block.Hash()); err != nil {
t.Fatalf("failed to insert block number: %v", err)
}
if err := WriteBlockReceipts(db, block.Hash(), block.NumberU64(), receipts[i]); err != nil {
t.Fatal("error writing block receipts:", err)
}
}

bloom := GetMipmapBloom(db, 0, 1000)
if bloom.TestBytes(addr2[:]) {
t.Error("address was included in bloom and should not have")
}
}

73:F:\git\coin\ethereum\go-ethereum\core\error.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import "errors"

var (
// ErrKnownBlock is returned when a block to import is already known locally.
ErrKnownBlock = errors.New("block already known")
```

```
// ErrGasLimitReached is returned by the gas pool if the amount of gas required
// by a transaction is higher than what's left in the block.
ErrGasLimitReached = errors.New("gas limit reached")

// ErrBlacklistedHash is returned if a block to import is on the blacklist.
ErrBlacklistedHash = errors.New("blacklisted hash")
)

74:F:\git\coin\ethereum\go-ethereum\core\events.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
)

// TxPreEvent is posted when a transaction enters the transaction pool.
type TxPreEvent struct{ Tx *types.Transaction }

// PendingLogsEvent is posted pre mining and notifies of pending logs.
type PendingLogsEvent struct {
Logs []*types.Log
}

// PendingStateEvent is posted pre mining and notifies of pending state changes.
type PendingStateEvent struct{}

// NewMinedBlockEvent is posted when a block has been imported.
type NewMinedBlockEvent struct{ Block *types.Block }

// RemovedTransactionEvent is posted when a reorg happens
type RemovedTransactionEvent struct{ Txs types.Transactions }

// RemovedLogsEvent is posted when a reorg happens
type RemovedLogsEvent struct{ Logs []*types.Log }

type ChainEvent struct {
Block *types.Block
Hash  common.Hash
Logs  []*types.Log
```

```
}

type ChainSideEvent struct {
Block *types.Block
}

type ChainHeadEvent struct{ Block *types.Block }

75:F:\git\coin\ethereum\go-ethereum\core\evm.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"math/big"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/core/vm"
)

// ChainContext supports retrieving headers and consensus parameters from the
// current blockchain to be used during transaction processing.
type ChainContext interface {
// Engine retrieves the chain's consensus engine.
Engine() consensus.Engine

// GetHeader returns the hash corresponding to their hash.
GetHeader(common.Hash, uint64) *types.Header
}

// NewEVMContext creates a new context for use in the EVM.
func NewEVMContext(msg Message, header *types.Header, chain ChainContext, author
*common.Address) vm.Context {
// If we don't have an explicit author (i.e. not mining), extract from the header
var beneficiary common.Address
if author == nil {
beneficiary, _ = chain.Engine().Author(header) // Ignore error, we're past header validation
} else {
beneficiary = *author
}
```

```
return vm.Context{
CanTransfer: CanTransfer,
Transfer:    Transfer,
GetHash:     GetHashFn(header, chain),
Origin:      msg.From(),
Coinbase:    beneficiary,
BlockNumber: new(big.Int).Set(header.Number),
Time:        new(big.Int).Set(header.Time),
Difficulty:  new(big.Int).Set(header.Difficulty),
GasLimit:    new(big.Int).Set(header.GasLimit),
GasPrice:    new(big.Int).Set(msg.GasPrice()),
}
}

// GetHashFn returns a GetHashFunc which retrieves header hashes by number
func GetHashFn(ref *types.Header, chain ChainContext) func(n uint64) common.Hash {
return func(n uint64) common.Hash {
for header := chain.GetHeader(ref.ParentHash, ref.Number.Uint64()-1); header != nil; header =
chain.GetHeader(header.ParentHash, header.Number.Uint64()-1) {
if header.Number.Uint64() == n {
return header.Hash()
}
}

return common.Hash{}
}
}

// CanTransfer checks wether there are enough funds in the address' account to make a transfer.
// This does not take the necessary gas in to account to make the transfer valid.
func CanTransfer(db vm.StateDB, addr common.Address, amount *big.Int) bool {
return db.GetBalance(addr).Cmp(amount) >= 0
}

// Transfer subtracts amount from sender and adds amount to recipient using the given Db
func Transfer(db vm.StateDB, sender, recipient common.Address, amount *big.Int) {
db.SubBalance(sender, amount)
db.AddBalance(recipient, amount)
}

76:F:\git\coin\ethereum\go-ethereum\core\fees.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.
```

```
package core

import (
"math/big"
)

var BlockReward = big.NewInt(5e+18)

77:F:\git\coin\ethereum\go-ethereum\core\filter_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

78:F:\git\coin\ethereum\go-ethereum\core\gaspool.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import "math/big"

// GasPool tracks the amount of gas available during
// execution of the transactions in a block.
// The zero value is a pool with zero gas available.
type GasPool big.Int

// AddGas makes gas available for execution.
func (gp *GasPool) AddGas(amount *big.Int) *GasPool {
i := (*big.Int)(gp)
i.Add(i, amount)
return gp
}

// SubGas deducts the given amount from the pool if enough gas is
// available and returns an error otherwise.
func (gp *GasPool) SubGas(amount *big.Int) error {
i := (*big.Int)(gp)
if i.Cmp(amount) < 0 {
return ErrGasLimitReached
}
i.Sub(i, amount)
return nil
```

```
}

func (gp *GasPool) String() string {
return (*big.Int)(gp).String()
}

79:F:\git\coin\ethereum\go-ethereum\core\genesis.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"errors"
"fmt"
"math/big"
"strings"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/hexutil"
"github.com/ethereum/go-ethereum/common/math"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/params"
"github.com/ethereum/go-ethereum/rlp"
)

//go:generate gencodec -type Genesis -field-override genesisSpecMarshaling -out gen_genesis.go
//go:generate gencodec -type GenesisAccount -field-override genesisAccountMarshaling -out
gen_genesis_account.go

var errGenesisNoConfig = errors.New("genesis has no chain configuration")

// Genesis specifies the header fields, state of a genesis block. It also defines hard
// fork switch-over blocks through the chain configuration.
type Genesis struct {
Config     *params.ChainConfig `json:"config"`
Nonce      uint64              `json:"nonce"`
Timestamp  uint64              `json:"timestamp"`
ParentHash common.Hash         `json:"parentHash"`
ExtraData  []byte              `json:"extraData"`
```

```
GasLimit   uint64             `json:"gasLimit"   gencodec:"required"`
Difficulty *big.Int           `json:"difficulty" gencodec:"required"`
Mixhash    common.Hash        `json:"mixHash"`
Coinbase   common.Address     `json:"coinbase"`
Alloc      GenesisAlloc       `json:"alloc"      gencodec:"required"`
}

// GenesisAlloc specifies the initial state that is part of the genesis block.
type GenesisAlloc map[common.Address]GenesisAccount

// GenesisAccount is an account in the state of the genesis block.
type GenesisAccount struct {
Code    []byte                  `json:"code,omitempty"`
Storage map[common.Hash]common.Hash `json:"storage,omitempty"`
Balance *big.Int                `json:"balance" gencodec:"required"`
Nonce   uint64                  `json:"nonce,omitempty"`
}

// field type overrides for gencodec
type genesisSpecMarshaling struct {
Nonce      math.HexOrDecimal64
Timestamp  math.HexOrDecimal64
ExtraData  hexutil.Bytes
GasLimit   math.HexOrDecimal64
Difficulty *math.HexOrDecimal256
Alloc      map[common.UnprefixedAddress]GenesisAccount
}
type genesisAccountMarshaling struct {
Code    hexutil.Bytes
Balance *math.HexOrDecimal256
Nonce   math.HexOrDecimal64
}

// GenesisMismatchError is raised when trying to overwrite an existing
// genesis block with an incompatible one.
type GenesisMismatchError struct {
Stored, New common.Hash
}

func (e *GenesisMismatchError) Error() string {
return fmt.Sprintf("database already contains an incompatible genesis block (have %x, new %x)",
e.Stored[:8], e.New[:8])
```

```
}

// SetupGenesisBlock writes or updates the genesis block in db.
// The block that will be used is:
//
//                          genesis == nil       genesis != nil
//                       +-----------------------------------------
//     db has no genesis |  main-net default  |  genesis
//     db has genesis    |  from DB           |  genesis (if compatible)
//
// The stored chain configuration will be updated if it is compatible (i.e. does not
// specify a fork block below the local head block). In case of a conflict, the
// error is a *params.ConfigCompatError and the new, unwritten config is returned.
//
// The returned chain configuration is never nil.
func SetupGenesisBlock(db ethdb.Database, genesis *Genesis) (*params.ChainConfig,
common.Hash, error) {
if genesis != nil && genesis.Config == nil {
return params.AllProtocolChanges, common.Hash{}, errGenesisNoConfig
}

// Just commit the new block if there is no stored genesis block.
stored := GetCanonicalHash(db, 0)
if (stored == common.Hash{}) {
if genesis == nil {
log.Info("Writing default main-net genesis block")
genesis = DefaultGenesisBlock()
} else {
log.Info("Writing custom genesis block")
}
block, err := genesis.Commit(db)
return genesis.Config, block.Hash(), err
}

// Check whether the genesis block is already written.
if genesis != nil {
block, _ := genesis.ToBlock()
hash := block.Hash()
if hash != stored {
return genesis.Config, block.Hash(), &GenesisMismatchError{stored, hash}
}
}
```

```
// Get the existing chain configuration.
newcfg := genesis.configOrDefault(stored)
storedcfg, err := GetChainConfig(db, stored)
if err != nil {
if err == ErrChainConfigNotFound {
// This case happens if a genesis write was interrupted.
log.Warn("Found genesis block without chain config")
err = WriteChainConfig(db, stored, newcfg)
}
return newcfg, stored, err
}
// Special case: don't change the existing config of a non-mainnet chain if no new
// config is supplied. These chains would get AllProtocolChanges (and a compat error)
// if we just continued here.
if genesis == nil && stored != params.MainnetGenesisHash {
return storedcfg, stored, nil
}

// Check config compatibility and write the config. Compatibility errors
// are returned to the caller unless we're already at block zero.
height := GetBlockNumber(db, GetHeadHeaderHash(db))
if height == missingNumber {
return newcfg, stored, fmt.Errorf("missing block number for head header hash")
}
compatErr := storedcfg.CheckCompatible(newcfg, height)
if compatErr != nil && height != 0 && compatErr.RewindTo != 0 {
return newcfg, stored, compatErr
}
return newcfg, stored, WriteChainConfig(db, stored, newcfg)
}

func (g *Genesis) configOrDefault(ghash common.Hash) *params.ChainConfig {
switch {
case g != nil:
return g.Config
case ghash == params.MainnetGenesisHash:
return params.MainnetChainConfig
case ghash == params.TestnetGenesisHash:
return params.TestnetChainConfig
default:
return params.AllProtocolChanges
```

```
}
}

// ToBlock creates the block and state of a genesis specification.
func (g *Genesis) ToBlock() (*types.Block, *state.StateDB) {
db, _ := ethdb.NewMemDatabase()
statedb, _ := state.New(common.Hash{}, state.NewDatabase(db))
for addr, account := range g.Alloc {
statedb.AddBalance(addr, account.Balance)
statedb.SetCode(addr, account.Code)
statedb.SetNonce(addr, account.Nonce)
for key, value := range account.Storage {
statedb.SetState(addr, key, value)
}
}
root := statedb.IntermediateRoot(false)
head := &types.Header{
Nonce:      types.EncodeNonce(g.Nonce),
Time:       new(big.Int).SetUint64(g.Timestamp),
ParentHash: g.ParentHash,
Extra:      g.ExtraData,
GasLimit:   new(big.Int).SetUint64(g.GasLimit),
Difficulty: g.Difficulty,
MixDigest:  g.Mixhash,
Coinbase:   g.Coinbase,
Root:       root,
}
if g.GasLimit == 0 {
head.GasLimit = params.GenesisGasLimit
}
if g.Difficulty == nil {
head.Difficulty = params.GenesisDifficulty
}
return types.NewBlock(head, nil, nil, nil), statedb
}

// Commit writes the block and state of a genesis specification to the database.
// The block is committed as the canonical head block.
func (g *Genesis) Commit(db ethdb.Database) (*types.Block, error) {
block, statedb := g.ToBlock()
if _, err := statedb.CommitTo(db, false); err != nil {
return nil, fmt.Errorf("cannot write state: %v", err)
```

```
}
if err := WriteTd(db, block.Hash(), block.NumberU64(), g.Difficulty); err != nil {
return nil, err
}
if err := WriteBlock(db, block); err != nil {
return nil, err
}
if err := WriteBlockReceipts(db, block.Hash(), block.NumberU64(), nil); err != nil {
return nil, err
}
if err := WriteCanonicalHash(db, block.Hash(), block.NumberU64()); err != nil {
return nil, err
}
if err := WriteHeadBlockHash(db, block.Hash()); err != nil {
return nil, err
}
if err := WriteHeadHeaderHash(db, block.Hash()); err != nil {
return nil, err
}
config := g.Config
if config == nil {
config = params.AllProtocolChanges
}
return block, WriteChainConfig(db, block.Hash(), config)
}

// MustCommit writes the genesis block and state to db, panicking on error.
// The block is committed as the canonical head block.
func (g *Genesis) MustCommit(db ethdb.Database) *types.Block {
block, err := g.Commit(db)
if err != nil {
panic(err)
}
return block
}

// GenesisBlockForTesting creates and writes a block in which addr has the given wei balance.
func GenesisBlockForTesting(db ethdb.Database, addr common.Address, balance *big.Int)
*types.Block {
g := Genesis{Alloc: GenesisAlloc{addr: {Balance: balance}}}
return g.MustCommit(db)
}
```

```
// DefaultGenesisBlock returns the Ethereum main net genesis block.
func DefaultGenesisBlock() *Genesis {
return &Genesis{
Config:     params.MainnetChainConfig,
Nonce:      66,
ExtraData:
hexutil.MustDecode("0x11bbe8db4e347b4e8c937c1c8370e4b5ed33adb3db69cbdb7a38e1e50b1
b82fa"),
GasLimit:   5000,
Difficulty: big.NewInt(17179869184),
Alloc:      decodePrealloc(mainnetAllocData),
}
}

// DefaultTestnetGenesisBlock returns the Ropsten network genesis block.
func DefaultTestnetGenesisBlock() *Genesis {
return &Genesis{
Config:     params.TestnetChainConfig,
Nonce:      66,
ExtraData:
hexutil.MustDecode("0x3535353535353535353535353535353535353535353535353535353535
53535"),
GasLimit:   16777216,
Difficulty: big.NewInt(1048576),
Alloc:      decodePrealloc(testnetAllocData),
}
}

// DefaultRinkebyGenesisBlock returns the Rinkeby network genesis block.
func DefaultRinkebyGenesisBlock() *Genesis {
return &Genesis{
Config:     params.RinkebyChainConfig,
Timestamp:  1492009146,
ExtraData:
hexutil.MustDecode("0x52657370656374206d7920617574686f7269746168207e452e436172746
d616e42eb768f2244c8811c63729a21a3569731535f067ffc57839b00206d1ad20c69a1981b489f77
2031b279182d99e65703f0076e4812653aab85fca0f0000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000"),
GasLimit:   4700000,
Difficulty: big.NewInt(1),
```

```
Alloc:      decodePrealloc(rinkebyAllocData),
}
}

// DevGenesisBlock returns the 'geth --dev' genesis block.
func DevGenesisBlock() *Genesis {
return &Genesis{
Config:     params.AllProtocolChanges,
Nonce:      42,
GasLimit:   4712388,
Difficulty: big.NewInt(131072),
Alloc:      decodePrealloc(devAllocData),
}
}

func decodePrealloc(data string) GenesisAlloc {
var p []struct{ Addr, Balance *big.Int }
if err := rlp.NewStream(strings.NewReader(data), 0).Decode(&p); err != nil {
panic(err)
}
ga := make(GenesisAlloc, len(p))
for _, account := range p {
ga[common.BigToAddress(account.Addr)] = GenesisAccount{Balance: account.Balance}
}
return ga
}

80:F:\git\coin\ethereum\go-ethereum\core\genesis_alloc.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

// Constants containing the genesis allocation of built-in genesis blocks.
// Their content is an RLP-encoded list of (address, balance) tuples.
// Use mkalloc.go to create/update them.

const mainnetAllocData =
"\xfa\x04]X\u0793\r\x83b\x011\x8e\u0189\x9agT\x06\x908'\x80t2\x80\x89\n\u05ce\xbcZ\xc6
\x00\x00\u0793\x17bC\x0e\xa9\u00e2nWI\xaf\xdbp\xda_x\u077b\x8c\x89\n\u05ce\xbcZ\xc6
\x00\x00\u0793\x1d\x14\x80K9\x9cn\xf8\x0edWoev`\x80O\xec\v\x89\u3bb5sr@\xa0\x00\x00\u079
32@5\x87\x94{\x9f\x15b*h\xd1\x04\xd5M3\xdb\xd1\u0349\x043\x87Oc,\xc6\x00\x00\u0793I~\x92\
xcd\xc0\xe0\xb9c\xd7R\xb2)j\u02c7\u0682\x8b$\x89\n\x8fd\x9f\xe7\xc6\x18\x00\x00\u0793K\xfb\x
```

e1Tk\xc6\xc6[\\~\xaaU0K8\xbb\xfe\xc6\u04c9lk\x93[\x8b\xbd@\x00\x00\u0793Z\x9c\x03\xf6\x9d\x17\xd6l\xbb\x8a\xd7!\x00\x8a\x9e\xbb\xb86\xfb\x89lk\x93[\x8b\xbd@\x00\x00\u0793]\x0e\xe8\x15^\xc0\xa6\xffh\bU,\xa5\xf1k\xb5\xbe2:\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u0793v\"\xd8J#K\xb8\xb0x#\x0f\u03c4\xb6z\u9a2c\xae\x89%\xe1\xccQ\x99R\xf8\x00\x00\u0793{\x9f\xc3\x19\x05\xb4\x99K\x04\xc9\xe2\xcf\xdc^'pP?B\x89l]\xb2\xa4\xd8\x15\xdc\x00\x00\u0793\u007fJ#\xca\x00\xcd\x04=%\u0088\x8c\x1a\xa5h\x8f\x81\xa3D\x89)\xf0\xa9[\xfb\xf7)\x00\x00\u0793\x869\u06bb\u3bac\x88{]\xc0\xe4>\x13\xbc\u0487\xd7l\x89\x10\xd0\xe3\xc8}n,\x00\x00\u0793\x89P\x86y\xab\xf8\xc7\x1b\xf6x\x16\x87\x12\x0e>j\x84XM\x89a\x94\x04\x9f0\xf7
\x00\x00\u0793\x8f\xc7\u02ed\xff\xbd\r\u007f\xe4O\x8d\xfd`\xa7\x9dr\x1a\x1c\x9c\x8965\u026d\xc5\u07a0\x00\x00\u0793\x95`\xa3\xdebxh\xf9\x1f\xa8\xbf\xe1\xc1\xb7\xaf\xaf\b\x18k\x89\x1cg\xf5\xf7\xba\xa0\xb0\x00\x00\u0793\x96\x97G\xf7\xa5\xb3\x06E\xfe\x00\xe4l\x01CZ\xce$\xcc7\x89\\(=A\x03\x94\x10\x00\x00\u0793\x9am}\xb3&g\x9bw\xc9\x03\x91\xa7Gm#\x8f;\xa3>\x89\n\xdaUGK\x814\x00\x00\u0793\x9e\xef\n\b\x86\x05n?i!\x18S\xb9\xb7E\u007f7\x82\u4262\xa8x\x06\x9b(\xe0\x00\x00\u0793\x9f\xdb\xf4N\x1fJcb\xb7i\u00daG_\x95\xa9l+\u01c9\x1e\x93\x12\x83\xcc\xc8P\x00\x00\u07d3\xa5y\u007fR\xc9\u054f\x18\x9f6\xb1\xd4]\x1b\xf6\x04\x1f/k\x8a\x01'\u0473F\x1a\xcd\x1a\x00\x00\u0793\xaaS\x81\xb2\x13\x8e\xbe\xff\xc1\x91\xd5\xd8\u00d1u;p\x98\u04895\xab\xb0\x9f\xfe\u07b6\x80\x00\u0793\xaa\xda%\xea\"\x86p\x9a\xbbB-
A\x92?\u04c0\xcd\x04\u01c9#=\xf3)\x9far\x00\x00\u0793\xac\xbf\xb2\xf2ZT\x85\xc79\xefp\xa4N\xee\xeb|e\xa6o\x89\x05k\xc7^-
c\x10\x00\x00\u07d3\xac\xc6\xf0\x82\xa4B\x82\x87d\xd1\x1fX\u0589J\xe4\b\xf0s\x8a\f\xb4\x9bD\xba`-
\x80\x00\x00\u0793\xb2w\xb0\x99\xa8\xe8f\xca\x0e\xc6[\u02c7(O\xd1B\xa5\x82\x89j\xcb=\xf2~\x1f\x88\x00\x00\u0753\xbd\xd4\x01:\xa3\x1c\x04al+\xc9x_'\x88\xf9\x15g\x9b\x88\xb9\xf6]\x00\xf6<\x00\x00\u0793\xc2}c\xfd\xe2K\x92\xee\x8a\x1e~\xd5\xd2m\x8d\xc5\xc8;\x03\x89lk\x93[\x8b\xbd@\x00\x00\u0553\xc4\x0f\xe2\tT#P\x9b\x9f\u0677T21X\xaf#\x10\xf3\x80\u0793\xd7^\xd6\fwO\x8b:ZQs\xfb\x183\xadq\x05\xa2\u0649l\xb7\xe7Hg\xd5\xe6\x00\x00\u07d3\u05cd\x89\xb3_G'\x16\xec\xea\xfe\xbf`\x05'\u04e1\xf9i\x8a\x05\xe0T\x9c\x962\xe1\xd8\x00\x00\u0793\xda\xe2{5\v\xae
\xc5e!$\xaf]\x8b\\\xba\x00\x1e\xc1\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u0793\xdc\x01\xcb\xf4Ix\xa4.\x8d\xe8\xe46\xed\xf9B\x05\u03f6\xec\x89O\x0f\xeb\xbc\u068c\xb4\x00\x00\u07d3\u607c-
\x10\xdbb\u0785\x84\x83$I\"P4\x8e\x90\xbf\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d3\xf4c\xe17\xdc\xf6%\xfb\U000fc8de\u0298\u04b9h\xcf\u007f\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d4\x01\x00\a9K\x8bue\xa1e\x8a\xf8\x8c\xe4cI\x915\u05b7\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x01\r\xf1\xdfK\xed#v\r-
\x1c\x03x\x15\x86\xdd\xf7\x91\x8eT\x89\x03@\xaa\xd2\x1b;p\x00\x00\xe0\x94\x01\x0fJ\x98\u07e1\xd9y\x9b\xf5\u01d6\xfbU\x0e\xfb\xe7\xec\xd8w\x8a\x01\xb2\xf2\x92#b\x92\xc7\x00\x00\u07d4\x01\x15PW\x00/k\r\x18\xac\xb98\x8d;\xc8\x12\x9f\x8fz
\x89H\xa4<T`/p\x00\x00\u07d4\x01\"n\n\xd8\xd6\"w\xb1bb\x1cb\xc9(\xe9n\v\x9a\x8c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x01&\xe1.\xbc\x17\x03_5\xc0\xe9\xd1\x1d\xd1H9<@]z\x89lf\x06E\xaaG\x18\x00\x00\u07d4\x01/9j+^\xb85Y\xba\xc5\x15\xe5!\r\xf2\xc8\xc3b\xba\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x014\xff8\x15_\xab\xae\x94\xfd5\xc4\xff\xe1\u05dd\xe7\xef\x9cY\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\x016\xa5\xafl2\x99\u01b5\xf0\x05\xfd\xad\xdb\x14\x8c\a\v)\x9b\x89\x01\x1a\xa9\xac\x15\xf1(\x00\x00\u07d4\x01H\x8a\xd3\xda`<L\xddl\xb0\xb7\xa1\xe3\r*0\xc8\xfc8\x89\n\u0

5ce\xbcZ\xc6
\x00\x00\u07d4\x01It\xa1\xf4k\xf2\x04\x94J\x851\x11\xe5/\x16\x02a}\xef\x89lk\x93[\x8b\xbd@\x00
\x00\u07d4\x01K\u007fg\xb1O]\x98=\x87\x01OW\f\x8b\x99;\x98r\xb5\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x01Q\xfa]\x17\xa2\xdc\xe2\xd7\xf1\xeb9\xef\u007f\xe2\xad!=]\x89\x89\xd8\xd7&\x
b7\x17z\x80\x00\x00\u07d4\x01Wz\xfdNP\x89\x02G\u0271\rD\xafs\"\x9a\xec\x88O\x89$\xdc\xe5
M4\xa1\xa0\x00\x00\xe0\x94\x01_\t}\x9a\xcd\xdc\u076f\xaf*\x10~\xb9:@\xfc\x94\xb0L\x8a\x04<3\
xc1\x93ud\x80\x00\x00\u07d4\x01i\xc1\xc2\x10\xea\xe8E\xe5h@A.\x1fe\x99>\xa9\x0f\xb4\x89lk\x
93[\x8b\xbd@\x00\x00\u07d4\x01k`\xbbmg\x92\x8c)\xfd\x03\x13\xc6f\u068f\x16\x98\xd9\u0149lk\
x93[\x8b\xbd@\x00\x00\u07d4\x01l\x85\xe1a;\x90\x0f\xa3W\xb8(;\x12\x0ee\xae\xfc\xdd\b\x89+]\x
97\x84\xa9|\xd5\x00\x00\u07d4\x01\x84\x92H\x8b\xa1\xa2\x924\"G\xb3\x18U\xa5Y\x05\xfe\xf2i\x8
9\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4\x01\x8f
\xa2{'\xecD\x1a\xf7#\xfd\x90\x99\xf2\u02f7\x9dbc\x89uy*\x8a\xbd\xef|\x00\x00\u07d4\x01\x91\xeb
T~{\xf6\x97k\x9b\x1bWuFv\x1d\xe6V\"\xe2\x89lkLM\xa6\u077e\x00\x00\u07d4\x01\x9dp\x95y\xffK
\xc0\x9f\xdc\xdd\xe41\xdc\x14G\xd2\xc2`\xbc\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x01\xa2
Z_Z\xf0\x16\x9b0\x86L;\xe4\xd7V<\xcdD\xf0\x9e\x89M\x85<\x8f\x89\b\x98\x00\x00\xe0\x94\x01\x
a7\xd9\xfa}\x0e\xb1\x18\\g\xe5M\xa8<.u\xdbi\u37ca\x01\x9dJ\xdd\xd0\u063c\x96\x00\x00\u07d4\x
01\xa8\x18\x13ZAB\x10\xc3|b\xb6%\xac\xa1\xa5F\x11\xac6\x89\x0e\x189\x8ev\x01\x90\x00\x00\u
07d4\x01\xb1\xca\xe9\x1a;\x95Y\xaf\xb3<\xdcmh\x94B\xfd\xbf\xe07\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x01\xb5\xb5\xbcZ\x11\u007f\xa0\x8b4\xed\x1d\xb9D\x06\bYz\xc5H\x89\n\u05ce\x
bcZ\xc6
\x00\x00\u07d4\x01\xbb\xc1Og\xaf\x069\xaa\xb1D\x1ej\b\xd4\xceqb\t\x0f\x89F\xfc\xf6\x8f\xf8\xbe\
x06\x00\x00\xe0\x94\x01\xd08\x15\xc6\x1fAkq\xa2a\n-
\xab\xa5\x9f\xf6\xa6\xde[\x8a\x02\x05\xdf\xe5\v\x81\xc8.\x00\x00\u07d4\x01\u0559\xee\r_\x8c8\xa
b-9.,e\xb7L<\xe3\x18
\x89\x1b\xa5\xab\xf9\xe7y8\x00\x00\u07d4\x01\xe4\x05!\x12%0\u066c\x91\x11<\x06\xa0\x19\vmc
\x85\v\x89Hz\x9a0E9D\x00\x00\u07d4\x01\xe6A]X{\x06T\x90\xf1\xed\u007f!\xd6\xe0\xf3\x86\xeeg
G\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x01\xe8d\xd3Tt\x1bB>oB\x85\x17$F\x8ct\xf5\xaa\x9c\x8
a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x01\xed_\xba\x8d.\xabg:\xec\x04-
0\xe4\xe8\xa6\x11\xd8\xc5Z\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x01\xfb\x8e\xc1$%\xa0O\x81>F\
xc5L\x05t\x8c\xa6\xb2\x9a\xa9\x89\x0e\x15s\x03\x85F|\x00\x00\u07d4\x01\xff\x1e\xb1\u07adP\xa
7\xf2\xf9c\x8f\xde\xe6\xec\xcf:{*\u0209
\x86\xac5\x10R`\x00\x00\u07d4\x02\x03b\u00ed\xe8x\u0290\u05b2\u0609\xa4\xccU\x10\xee\xd5\
xf3\x898\x88\xe8\xb3\x11\xad\xb3\x80\x00\u07d4\x02\x03\xae\x01\xd4\xc4\x1c\xae\x18e\xe0K\x1f
[S\xcd\xfa\xec\xae1\x896\x89\xcd\u03b2\x8c\xd7\x00\x00\u07d4\x02\b\x93a\xa3\xfetQ\xfb\x1f\x87\
xf0\x1a-
\x86fS\xdc\v\a\x89\x02*\xc7H2\xb5\x04\x00\x00\u07d4\x02\x1fi\x04=\xe8\x8cI\x17\xca\x10\xf1\x84
(\x97\xee\xc0X\x9c|\x89kD\u03f8\x14\x87\xf4\x00\x00\u07d4\x02)\x0f\xb5\xf9\xa5\x17\xf8(E\xac\x
de\xca\x0f\xc8F\x03\x9b\xe23\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x029\xb4\xf2\x1f\x8e\x05\xcd\
x01Q++\xe7\xa0\xe1\x8am\x97F\a\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x02Gr\x12\xff\xddu\xe
5\x15VQ\xb7e\x06\xb1dfq\xa1\xeb\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\x02J\t\x8a\xe7\x02\x
be\xf5@l\x9c\"\xb7\x8b\xd4\xeb,\u01e2\x93\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x02K\xdd,
{\xfdP\x0e\xe7@O\u007f\xb3\xe9\xfb1\xdd

\xfb\u0449\t\xc2\x00vQ\xb2P\x00\x00\u07d4\x02Sg\x96\x03\x04\xbe\xee4Y\x11\x18\xe9\xac-
\x13X\xd8\x02\x1a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x02V\x14\x9f[Pc\xbe\xa1N\x15f\x1f\xfbX\x
f9\xb4Y\xa9W\x89&)\xf6n\fS\x00\x00\x00\u07d4\x02`=z;\xb2\x97\xc6|\x87~]4\xfb\u0579\x13\xd4\x
c6:\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x02a\xad:\x17*\xbf\x13\x15\xf0\xff\xec2p\x98j\x84\t
\xcb%\x89\v\b!;\u03cf\xfe\x00\x00\u07d4\x02d2\xaf7\xdcQ\x13\xf1\xf4mH\nM\u0c80R#~\x89\x13l\
xb7\x86\xe4\v\xfc\x00\x00\u07d4\x02f\xab\x1ck\x02\x16#\v\x93\x95D=_\xa7^hEh\u018965\u026d\
xc5\u07a0\x00\x00\u07d4\x02u\x1d\u018c\xb5\xbdsp'\xab\xf7\u0777s\x90\xcdw\xc1k\x89\x01\x15
\x8eF\t\x13\xd0\x00\x00\xe0\x94\x02w\x8e9\x0f\xa1u\x10\xa3B\x8a\xf2\x87\fBsT}8l\x8a\x03l<f\x17
\f\rr\x00\x00\xe0\x94\x02\xad\xe5\xdb\"\xf8\xb7X\xee\x14Cbld\xec/2\xaa\n\x15\x8a\x04<3\xc1\x93
ud\x80\x00\x00\u07d4\x02\xaf$Y\xa9=\v?M\x06&6#l\u0532\x9e;\xce\u03c9g\x8a\x93
b\xe4\x18\x00\x00\u07d4\x02\xb1\xafr3\x9b*\"V8\x9f\xd6F\a\xde$\xf0\xde`\n\x89lk\x93[\x8b\xbd@\
x00\x00\xe0\x94\x02\xb6C\xd6\xfa\xbdCz\x85\x1a\u033ey\xab\xb7\xfd\xe1&\xdc\u03ca\x01\x86P\
x12|\xc3\u0700\x00\x00\xe0\x94\x02\xb6\xd6\\\xb0\v{6\xe1\xfb^\xd3c,L\xb2\n\x89A0\x8a\x04<3\xc
1\x93ud\x80\x00\x00\u07d4\x02\xb7\xb1\u05b3L\xe0S\xa4\x0e\xb6\\\u0524\xf7\xdd\xdd\x0e\x9f0\
x89%\"H\u07b6\xe6\x94\x00\x00\xe0\x94\x02\xc9\xf7\x94\n{\x8bzA\v\xf8=\xc9\xc2#3\xd4']\u04ca\
x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\x02\u0523\th\xa3\x9e+4\x98\u00e6\xa4\xedE\xc1\xc6dh\"\x89lk\x93[\x8b\xbd@\x0
0\x00\u07d4\x02\xdf\xcb\x17\xa1\xb8tA\x03ct\xb7b\xa5\xd3A\x8b\x1c\xb4\u0509H\xb0+\xa9\u047
aF\x00\x00\u07d4\x02\xe4\xcb\"\xbeF%\x8a@\xe1mC8\xd8\x02\xff\xfd\x00\xc1Q\x89\x14\x96\x96
\xea\u03ba\x81\x00\x00\u07d4\x02\xe8\x16\xaf\xc1\xb5\xc0\xf3\x98R\x13\x19Y\xd9F\xeb;\a\xb5\x
ad\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x02\xf7\xf6r\t\xb1j\x17U\fiLrX8\x19\xc8\vT\xad\x89\x05
U\x93\x06\xa7\x8ap\x00\x00\xe0\x94\x03\ts\x80{/Bi\x14\xad\x00\x18\x12p\xac\xd2{\x8f\xf6\x1f\x8a
\x01!\xeah\xc1\x14\xe5\x10\x00\x00\u07d4\x03\ty#\xba\x15^\x16\xd8/:\xd3\xf6\xb8\x15T\b\x84\xb9
,\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\x03\x0f\xb3@\x1fr\xbd4\x18\xb7\xd1\xdau\xbf\x8cQ\x9d\
xd7\a\u0709\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x03\x1e%\xdbQk\x0f\t\x9f\xae\xbf\xd9O\x89\f\xf9
f`\x83k\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x03(Q\f\t\xdb\u0345\x19J\x98\xd6|3\xacI\xf2\xf9M`\x
8a\x02TO\xaaw\x80\x90\xe0\x00\x00\u07d4\x03)\x18\x8f\b\x06W\xab:*\xfaR$g\x17\x82y\x83
\x85\x89\v\xbfQ\r\xdf\xcb&\x00\x00\u07d4\x031x&\xd1\xf7\n\xa4\xbd\u07e0\x9b\xe0\xc4\x10UR\xd
25\x8b\x89\x02\x1auJm\xc5(\x00\x00\u07d4\x033p\x12\xae\x1d\u007f\xf3\xee\u007fi|@>w\x80\x1
8\x8b\xf0\xef\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x037|\x0eUkd\x01\x03(\x9aa\x89\u1baecI4g\x8a\x04<3\xc1\x93ud\x80\x00\x00\
u07d4\x03IcM\u00a9\xe8\f?w!\xee+PF\xae\xaa\xed\xfb\xb5\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u
07d4\x03U\xbc\xac\xbd!D\x1e\x95\xad\xee\xdc0\xc1r\x18\u0224\b\u0389\x15\xaf\x1dx\xb5\x8c@\
x00\x00\u07d4\x03n\xef\xf5\xba\x90\xa6\x87\x9a\x14\xdf\xf4\xc5\x04;\x18\xca\x04`\u0249\x05k\xc
7^-
c\x10\x00\x00\xe0\x94\x03qKA\u04a6\xf7Q\x00\x8e\xf8\xddM+)\xae\u02b8\xf3n\x8a\x01EB\xba\x
12\xa37\xc0\x00\x00\xe0\x94\x03r\xe8RX.\t44J\x0f\xed!x0M\xf2]F(\x8a\x04<3\xc1\x93ud\x80\x00\
x00\u07d4\x03r\xeeU\b\xbf\x81c\xed(N^\xef\x94\xceMsg\xe5\"\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x03}\xd0V\xe7\xfd\xbdd\x1d\xb5\xb6\xbe\xa2\xa8x\n\x83\xfa\u1009\a\x96\xe
3\xea?\x8a\xb0\x00\x00\xe0\x94\x03\x83#\xb1\x84\xcf\xf7\xa8*\xe2\u1f67y?\xe41\x9c\xa0\xbf\x8a\
x04<3\xc1\x93ud\x80\x00\x00\u07d4\x03\x87y\xca-
\xbef>c\xdb?\xe7V\x83\xea\x0e\xc6.#\x83\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\xe0\x94\x03\x8eE\x

ea\xdd=\x88\xb8\u007f\xe4\u06b0fh\x05\"\xf0\xdf\xc8\xf9\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x03\x92T\x9ar\u007f\x81eT)\u02d2\x8bR\x9f%\xdfM\x13\x85\x89\x01lC\xa0\xee\xa0t\x00\x00\u07d4\x03\x94\xb9\x0f\xad\xb8`O\x86\xf4?\xc1\xe3]1$\xb3*Y\x89\x89)j\xa1@'\x8ep\x00\x00\u0794\x03\x9ezN\xbc(N,\xcdB\xb1\xbd\xd6\v\xd6Q\x1c\x0fw\x06\x88\xf0\x15\xf2W6B\x00\x00\u07d4\x03\x9e\xf1\xceR\xfeyc\xf1f\u0562u\u0131\x06\x9f\xe3\xa82\x89\x15\xaf9\u4ab2t\x00\x00\u07d4\x03\xa2l\xfcL\x181op\u055e\x9e\x1ay\xee>\x8b\x96/L\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x03\xaab(\x81#m\xd0\xf4\x94\f$\xc3$\xff\x8b{~!\x86\x89\xadx\xeb\u016cb\x00\x00\x00\u07d4\x03\xafz\xd9\xd5\"<\xf7\xc8\xc1? \xdfg\xeb\xe5\xff\u017bA\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x03\xb0\xf1|\xd4F\x9d\xdc\u03f7\xdai~\x82\xa9\x1a_\x9ewt\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x03\xb4\x1bQ\xf4\x1d\xf2\r\xd2y\xba\xe1\x8c\x12w_w\xadw\x1c\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x03\xbe[F)\xae\xfb\xbc\xab\x9d\xe2m9Wl\xb7\xf6\x91\xd7d\x89\n\xdf0\xbap\u0217\x00\x00\u07d4\x03\xc6G\xa9\xf9)\xb0x\x1f\xe9\xae\x01\u02a3\xe1\x83\xe8vw~\x89\x18*\xb7\xc2\f\xe5$\x00\x00\u07d4\x03\xc9\x1d\x92\x946\x03\xe7R
>\x054\x0eV`\x13\xb9\x00E\x89+|\xc2\xe9\xc3\"\\\x00\x00\u07d4\x03\xcbLOE\x16\xc4\xffy\xa1\xb6$O\xbfW.\x1c\u007f\xeay\x89\x94\x89#z\u06daP\x00\x00\u07d4\x03\u02d8\u05ec\xd8\x17\u079d\x88m\"\xfa\xb3\xf1\xb5}\x92\xa6\b\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4\x03\u031d-
!\xf8k\x84\xac\x8c\xea\xf9q\u06e7\x8a\x90\xe6%p\x89WG=\x05\u06ba\xe8\x00\x00\u07d4\x03\xd1rO\xd0\x0eT\xaa\xbc\xd2\xde*\x91\xe8F+\x10l\xdd:\x89\x8f\x1d\\\x1c\xae7@\x00\x00\u07d4\x03\xde\xdf\xcd\v<.\x17\xc7\x05\xda$\x87\x90\uf626\xbdWQ\x89Hz\x9a0E9D\x00\x00\u07d4\x03\u8c04SuW\xe7\t\xea\xe2\xe1\u1966\xbc\xe1\xef\x83\x14\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x03\xeam&\u0400\xe5z\xee9&\xb1\x8e\x8e\xd7:N[(&\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x03\xeb<\xb8`\xf6\x02\x8d\xa5T\xd3D\xa2\xbbZP\n\xe8\xb8o\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x03\xeb\xc6?\xdaf`\xa4e\x04^#_\xben\\\xf1\x95s_\x89\a\xb0l\xe8\u007f\xddh\x00\x00\xe0\x94\x03\xefj\xd2\x0f\xf7\xbdO\x00+\xacX\xd4uD\u03c7\x9a\xe7(\x8a\x01u\xc7X\u0439n\\\x00\x00\u07d4\x03\xf7\xb9
\b\x81:\xe0\xa6v\xeb!(\x14\xaf\xab5\"\x10i\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x04\x11p\xf5\x81\u0780\xe5\x8b*\x04\\\x8f|\x14\x93\xb0\x01\xb7\u02c90<t\xa1\xa36\x94\x00\x00\u07d4\x04\x13\xd0\xcfx\xc0\x01\x89\x8a7\x8b\x91\x8c\xd6\xe4\x98\xeaw<M\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\u07d4\x04$\x1bA\xec\xbd\v\xfd\xf1)^\x9dO\xa5\x9e\xa0\x9ela\x86\x89e_v\x94P\xbcx\x00\x00\u07d4\x047\a\a\x1e*\xe2\x1e\xed\x97x\x91\xdcy\xcd]\x8e\xe1\xc2\u0689lk\x93[\x8b\xbd@\x00\x00\u07d4\x04N\x851D\xe36D\x95\u799f\xa1\xd4j\xbe\xa3\xac\td\x89\x02\xab\"T\xb1\u071a\x80\x00\u07d4\x04U\xdc\xec\x8a\u007f\xc4F\x1b\xfd\u007f7Eo\xce?L<\xaa\u01c9\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x04^\xd7\xf6\xd9\xee\x9f%.\a2h\xdb\x02,c&\xad\xfc[\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x04cw\xf8d\xb0\x14?(!t\xa8\x92\xa7=>\xc8\xeca2\x89\nZ\xa8P\t\xe3\x9c\x00\x00\u07d4\x04i\xe8\xc4@E\v\x0eQ&&\xfe\x81~gT\xa8\x15(0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x04m'K\x1a\xf6\x15\xfbPZvJ\xd8\u0767p\xb1\xdb/=\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x04}Z&\u05ed\x8f\x8ep`\x0fp\xa3\x98\u076a\x1c-
\xb2o\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x04~\x87\xc8\xf7\xd1\xfc\xe3\xb0\x13S\xa8Xb\xa9H\xac\x04\x9f>\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4\x04\u007f\x9b\xf1R\x9d\xaf\x87\xd4\a\x17^o\x17\x1b^Y\xe9\xff>\x89#<\x8f\xe4'\x03\xe8\x00\x00\xe0\x94\x04\x85'2\xb4\xc6R\xf6\xc2\u53b3e\x87\xe6\nb\xda\x14\u06ca\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\x04\x8a\x89p\xeaAE\xc6M

U\x17\xb8\xde[F\xd0YZ\xad\x06\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x04\x9c]K\xc6\xf2]NEli{
R\xa0x\x11\xcc\u045f\xb1\x89\x10D\x00\xa2G\x0eh\x00\x00\u07d4\x04\xa1\xca\xda\x1c\xc7Q\b/\
xf8\u0692\x8e<\xfa\x00\b
\xa9\xe9\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\x04\xa8\n\xfa\xd5>\xf1\xf8Ae\xcf\xd8R\xb0\xfd
\xf1\xb1\xc2K\xa8\x89\x03$\xe9d\xb3\xec\xa8\x00\x00\u07d4\x04\xaa\xfc\x8a\xe5\xceoI\x03\u021
d\u007f\xac\x9c\xb1\x95\x12\"Gw\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x04\xbaK\xb8q@\
x02,!Jo\xacB\xdbZ\x16\u0755@E\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x04\xba\x8a?\x03\xf0\x
8b\x89P\x95\x99M\xdaa\x9e\u06ac\xee>z\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x04\xc2\xc6K\xb5
L>\xcc\xd0U\x85\xe1\x0e\xc6\xf9\x9a\f\xdb\x01\xa3\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x04\xceE\xf6\x00\xdb\x18\xa9\u0405\x1b)\xd99>\xbd\xaa\xfe=\u0149\x01\x1
5\x8eF\t\x13\xd0\x00\x00\u07d4\x04\u05b8\xd4\u0686t\a\xbb\x99wI\u07bb\xcd\xc0\xb3XS\x8a\x8
965\u026d\xc5\u07a0\x00\x00\u07d4\x04\xd78\x96\xcfe\x93\xa6\x91\x97*\x13\xa6\xe4\x87\x1f\xf2
\xc4+\x13\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x04\xd8*\xf9\xe0\x1a\x93m\x97\xf8\xf8Y@\xb9p\xf
9\xd4\u06d96\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x04\xe5\xf5\xbc|\x92?\xd1\xe3\x175\xe7.\xf9h\xfdg\x11\fn\x89WU\x1d\xbc\x8ebL\x
00\x00\u07d4\x04\xec\xa5\x01c\n\xbc\xe3R\x18\xb1t\x95k\x89\x1b\xa2^\xfb#\x8966\x9e\xd7t}&\x0
0\x00\u07d4\x05\x05\xa0\x8e\"\xa1\t\x01Z\"\xf6\x850STf*U1\u0549\x8c\xf2?\x90\x9c\x0f\xa0\x00\x
00\u07d4\x05\x14\x95L</\xb6W\xf9\xa0oQ\x0e\xa2'H\xf0'\xcd\u04c9\x15\xaf\x1dx\xb5\x8c@\x00\x
00\xe0\x94\x05\x163\b\r\a\xa5W\xad\xde1\x92a\xb0t\x99\u007f\x14i-
\x8a\x01:k+VHq\xa0\x00\x00\u07d4\x05\x17D\x8d\xad\xa7a\xcc[\xa4\x03>\xe8\x81\xc807\x03d\x0
0\x89lO\xd1\xee$nx\x00\x00\u07d4\x05\x1dBBv\xb2\x129fQ\x86\x13=e;\xb8\xb1\x86/\x89\x8965\u
026d\xc5\u07a0\x00\x00\u07d4\x05!\xbc:\x9f\x87\x11\xfe\xcb\x10\xf5\a\x97\xd7\x10\x83\xe3A\ub7
49\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x05#mL\x90\xd0e\xf9\u34c3X\xaa\xff\xd7w\xb8j\xecI\x8
9\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x05*X\xe05\xf1\xfe\x9c\xdd\x16\x9b\xcf
\x97\x03E\xd1+\x9cQ\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4\x05.\xab\x1fa\xb6\xd4U\x17(?A\xd1D
\x18$\x87\x87I\u0409\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x053n\x9ar'(\xd9c\xe7\xa1\xcf'Y\xfd\
x02tS\x0f\u02891\xa2D?\x88\x8ay\x80\x00\u07d4\x054q\u035aA\x92[9\x04\xa5\xa8\xff\xca6Y\xe0
4\xbe#\x89\n\xd2\x01\xa6yO\xf8\x00\x00\u07d4\x056\x1d\x8e\xb6\x94\x1dN\x90\xfb~\x14\x18\xa9
Z2\xd5%w2\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x05B:T\xc8\xd0\xf9p~pAs\xd9#\xb9F\xed\
xc8\xe7\x00\x89\x06\xea\x03\u00bf\x8b\xa5\x80\x00\u07d4\x05D\f[\a;R\x9bH)
\x9d\xff\x88\t\x0e\a\xc4\xf6\xf5\x89E\u04977\xe2/
\x00\x00\u07d4\x05Z\xb6X\xc6\xf0\xedO\x87^\xd6t.K\xc7)-
\x1a\xbb\xf0\x89\x04\x86\u02d7\x99\x19\x1e\x00\x00\u07d4\x05[\xd0,\xaf\x19\xd6
+\xbc\u0703m\x18{\xd1\xc0\x1c\xf2a\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x05^\xacO\x1a\xd3\xf5\x8f\v\xd0$\u058e\xa6\r\xbe\x01\u01af\xb3\x89\x05k\x
c7^-
c\x10\x00\x00\u07d4\x05fQU\xccI\xcb\xf6\xaa\xbd\u056e\x92\xcb\xfa\xad\x82\xb8\xc0\xc1\x89\x1
5\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x05f\x86\a\x8f\xb6\xbc\xf9\xba\n\x8a\x8d\xc6:\x90o_\xea\xc0
\xea\x89\x1b\x18\x1eK\xf24<\x00\x00\u07d4\x05iks\x91k\xd3\x03>\x05R\x1e2\x11\xdf\xec\x02n\x
98\xe4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x05k\x15F\x89O\x9a\x85\xe2\x03\xfb3m\xb5i\xb1l%\x
e0O\x89\t.\xdb\t\xff\b\u0600\x00\u07d4\x05yI\xe1\xca\x05pF\x9eL\xe3\u0190\xaea:k\x01\xc5Y\x89
\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\x05}\u049f-

\x19\xaa=\xa4#'\xeaP\xbc\xe8o\xf5\xc9\x11\u0649\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x05\u007f\u007f\x81\xcdz@o\xc4Y\x94@\x8bPI\x91,Vdc\x89\\(=A\x03\x94\x10\x00\x00\u07d4\x05\x91]N\"Zf\x81b\xae\xe7\xd6\xc2_\xcf\xc6\xed\x18\xdb\x03\x89\x03\x98\xc3ry%\x9e\x00\x00\u07d4\x05\x96\xa2}\xc3\xee\x11_\xce/\x94\xb4\x81\xbc
z\x9e&\x15%\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x05\xa80rC\x02\xbc\x0fn\xbd\xaa\x1e\xbe\xee\xb4nl\xe0\v9\x89\x05V\xf6L\x1f\xe7\xfa\x00\x00\u07d4\x05\xae\u007f\u053b\u0300\xca\x11\xa9\n\x1e\u01e3\x01\xf7\xcc\u0303\u06c91T\xc9r\x9d\x05x\x00\x00\u07d4\x05\xbbd\xa9\x16\xbef\xf4`\xf5\xe3\xb6C2\x11\r
\x9e\x19\xae\x89\u3bb5sr@\xa0\x00\x00\xe0\x94\x05\xbfO\xcf\xe7r\xe4[\x82dC\x85.l5\x13P\xcer\xa2\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94\x05\xc6@\x04\xa9\xa8&\xe9N^N\xe2g\xfa*v2\xddNo\x8a\x03m\xc4.\xbf\xf9\v\u007f\x80\x00\xe0\x94\x05\xc76\xd3e\xaa7\xb5\xc0\xbe\x9c\x12\u022d\\\xd9\x03\xc3,\xf9\x8a\x01E^{\x80\n\x86\x88\x00\x00\xe0\x94\x05\xcbl;\x00r\xd3\x11ga\xb52\xb2\x18D;S\xe8\xf6\u014a\x1e\x02\xc3\xd7\xfc\xa9\xb6(\x00\x00\u07d4\x05\xd0\xf4\xd7(\xeb\xe8.\x84\xbfYu\x15\xadA\xb6\v\xf2\x8b9\x89\u3bb5sr@\xa0\x00\x00\u07d4\x05\u058d\xada\u04fb\u07f3\xf7y&\\IGJ\xff?\xcd0\x89\x02\"\xc5]\xc1Q\x9d\x80\x00\u07d4\x05\xe6q\xdeU\xaf\xec\x96K\aM\xe5t\xd5\x15\x8d]!\xb0\xa3\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x05\xe9{\tI,\u058fc\xb1+\x89.\xd1\xd1\x1d\x15,\x0e\u02897\b\xba\xed=h\x90\x00\x00\u07d4\x05\xf3c\x1fVd\xbd\xad]\x012\xc88\x8d6\xd7\u0612\t\x18\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\x06\t\xd8:I\xe1\xff\u0276\x90\xf3\xe9\xa8\x1e\x98>\x8b\xdcM\x9d\x8a\x0e\u04b5%\x84\x1a\xdf\xc0\x00\x00\u07d4\x06\x1e\xa4\x87|\u0409D\xebd\u0096n\x9d\xb8\xde\xdc\xfe\xc0k\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x06%\xd0`V\x96\x8b\x00\"\x06\xff\x91\x98\x01@$+\xfa\xa4\x99\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x06(\xbf\xbeU5x/\xb5\x88@k\xc9f`\xa4\x9b\x01\x1a\xf5\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d4\x061\u044b\xbb\xbd0\xd9\xe1s+\xf3n\xda\xe2\u0389\x01\xab\x80\x89\xa3\xf9\x88U\xec9\x90\x00\x00\u07d4\x061\xdc@\xd7NP\x95\xe3r\x9e\xdd\xf4\x95D\xec\xd49og\x89\b\xacr0H\x9e\x80\x00\x00\xe0\x94\x067Y\xdd\x1cN6.\xb1\x93\x98\x95\x1f\xf9\xf8\xfa\xd1\xd3\x10h\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x06_\xf5u\xfd\x9c\x16\xd3\xcbo\u058f\xfc\x8fH?\xc3.\xc85\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x06a\x8e\x9dWb\xdfb\x02\x86\x01\xa8\x1dD\x87\u05a0\xec\xb8\x0e\x89Hz\x9a0E9D\x00\x00\xe0\x94\x06fG\xcf\xc8]#\xd3v\x05W=
\x8c\xa1T\xb2D\xd7l\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x06xeJ\xc6v\x1d\xb9\x04\xa2\xf7\xe8Y^\xc1\xea\xacsC\b\x89/\x98\xb2\x9c(\x18\xf8\x00\x00\u07d4\x06\x86\n\x93RYU\xffbI@\xfa\xdc\xff\xb8\xe1I\xfdY\x9c\x89lh\xcc\u041b\x02,\x00\x00\xe0\x94\x06\x8c\xe8\xbdn\x90*E\u02c3\xb5\x15A\xb4\x0f9\xc4F\x97\x12\x8a\x01\x1c\x0f\x9b\xadJF\xe0\x00\x00\u07d4\x06\x8e)\xb3\xf1\x91\xc8\x12\xa699\x18\xf7\x1a\xb93\xaehG\xf2\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\x06\x8eeWf\xb9D\xfb&6\x19e\x87@\xb8P\xc9J\xfa1\x89\x01\xe8\u007f\x85\x80\x9d\xc0\x00\x00\u0794\x06\x96N-\x17\xe9\x18\x9f\x88\xa8
96\xb4\n\xc9nS<\x06\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x06\x99L\xd8:\xa2d\n\x97\xb2`\vA3\x9d\x1e\r>\xdeI\x89\r\x8drkqw\xa8\x00\x00\u07d4\x06\x9e\u042bz\xa7}\xe5q\xf1a\x06\x05\x1d\x92\xaf\xe1\x95\xf2\u0409\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x06\xac&\xad\x92\u02c5\x9b\u0550]\xdc\xe4&j\xa0\xecP\xa9\u0149*\x03I\x19\u07ff\xbc\x00\x00\u07d4\x06\xb0\xc1\xe3\u007fZ^\u013b\xf5\b@T\x8f\x9d:\xc0(\x88\x97\x89\xd8\u0602\u148e}\x00\x00\u07d4\x06\xb0\xff\x83@s\xcc\xe1\xcb\xc9\xeaU~\xa8{`Yc\u8d09\x10CV\x1a\x

88)0\x00\x00\xe0\x94\x06\xb1\x06d\x9a\xa8\xc4!\xdd\xcd\x1b\x8c2\xcd\x04\x18\xcf0\xda\x1f\x8a\
bxg\x83&\xea\xc9\x00\x00\x00\u07d4\x06\xb5\xed\xe6\xfd\xf1\xd6\xe9\xa3G!7\x9a\xea\xa1|q=\xd
8*\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x06\xcb\xfa\b\xcd\xd4\xfb\xa77\xba\xc4\a\xbe\x82$\xf4\xe
e\xf3X(\x89
+\xe5\xe88.\x8b\x80\x00\u07d4\x06\xd6\xcb0\x84\x81\xc36\xa6\xe1\xa2%\xa9\x12\xf6\xe65Y@\x
a1\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\x06\xdc\u007f\x18\xce\xe7\xed\xab[yS7\xb1\xdfj\x9e\
x8b\u062eY\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x06\xf6\x8d\xe3\xd79\xdbA\x12\x1e\xac\x
f7y\xaa\xda=\xe8v!\a\x89\x01\x84\x93\xfb\xa6N\xf0\x00\x00\u07d4\x06\xf7\u070d\x1b\x94b\xce\xf
6\xfe\xb13h\xa7\xe3\x97K\t\u007f\x9f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\a\x01\xf9\xf1G\xecHhV
\xf5\xe1\xb7\x1d\xe9\xf1\x17\xe9\x9e!\x05\x89\te\xdaq\u007f\u0578\x00\x00\u07d4\a\r]6L\xb7\xbb
\xf8\"\xfc,\xa9\x1a5\xbd\xd4A\xb2\x15\u0549lk\x93[\x8b\xbd@\x00\x00\xe0\x94\a\x1d\xd9\r\x14\xd
4\x1fO\xf7\xc4\x13\xc2B8\xd35\x9c\xd6\x1a\a\x8a\a\xb5?y\xe8\x88\xda\xc0\x00\x00\u07d4\a&\xc4
.\x00\xf4T\x04\x83n\xb1\xe2\x80\xd0s\xe7\x05\x96\x87\xf5\x89X\x00>?\xb9G\xa3\x80\x00\xe0\x9
4\a'\xbe\n*\x00!
H\xb5R\x0f\xbe\xfb\x95>\xbc\x9dT\xa0\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\a)\xa8\
xa4\xa5\xba#\xf5y\xd0\x02[\x1a\xd0\xf8\xa0\xd3\\\xdf\u048a\x02\r\u058a\xaf2\x89\x10\x00\x00\u0
7d4\a)\xb4\xb4|\t\xeb\x16\x15\x84d\u022a\u007f\xd9i\vC\x889\x89lh\xcc\u041b\x02,\x00\x00\u079
4\a4\xa0\xa8\x1c\x95b\xf4\xd9\xe9\xe1\n\x85\x03\xda\x15\xdbF\xd7n\x88\xfc\x93c\x92\x80\x1c\x0
0\x00\xe0\x94\a<g\xe0\x9b\\q<R!\u0220\xc7\xf3\xf7Df\xc3G\xb0\x8a\x04\x1b\xad\x15^e\x12
\x00\x00\u07d4\a?\x1e\xd1\xc9\xc3\xe9\xc5*\x9b\x02I\xa5\xc1\u02a0W\x1f\xdf\x05\x89\x03\xd0\xf
f\v\x01;\x80\x00\x00\u07d4\aHq1E\xef\x83\xc3\xf0\xefM1\xd8#xo~\x9c\u0189\x89\xf3\xf2\v\x8d\xf
ai\xd0\x00\x00\u07d4\a]\x15\xe2\xd3=\x8bO\xa7\u06e8\xb9\xe6\a\xf0J&\x1e4\v\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\aea\xa8VE]~\xf8nc\xf8|s\u06f6(\xa5_E\x890\xca\x02O\x98{\x90\x00\x00
\u07d4\an\xe9\x9d5Hb:\x03\xb5\xf9\x98Y\xd2\u05c5\xa1w\x8dH\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\ap\xb4=\xba\xe4\xb1\xf3Z\x92{O\xa8\x12M8f\xca\xf9{\x897\x19>\xa7\xef[G\x00\x0
0\xe0\x94\ap\xc6\x1b\xe7\x87r#\f\xb5\xa3\xbb$)\xa7&\x14\xa0\xb36\x8a\x01n\u0899\xb7\x13A\x8
0\x00\u07d4\ar><0\xe8\xb71\xeeEj)\x1e\xe0\u7630
Jw\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\as\xee\xac\xc0P\xf7G
\xb4\xa1\xbdW\x89[\x1c\xce\xebI]\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\a\x80\r/\x80
h\xe4H\u01daOi\xb1\xf1^\xf6\x82\xaa\xe5\xf6\x8a\x04\x1b\xad\x15^e\x12
\x00\x00\u07d4\a\xa8\xda\xde\xc1BW\x1a}S\xa4)pQxm\a,\xbaU\x89\x01;m\xa1\x13\x9b\u0680\x0
0\u07d4\a\xaf\x93\x8c\x127\xa2|\x900\tM\xcf$\aP$n=,\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\
x94\a\xb1\xa3\x06\xcbC\x12\xdffH,,\xaer\xd1\xe0a@\x0f\u034a\x04<3\xc1\x93ud\x80\x00\x00\u07
d4\a\xb7\xa5p3\xf8\xf1\x130\xe4f^\x18]#N\x83\xec\x14\v\x89\xea~\xe9*\f\x9a\v\x80\x00\u07d4\a\x
bc,\xc8\xee\xdc\x01\x97\a\x00\xef\xc9\xc4\xfb6s^\x98\xcdq\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u
07d4\a\xd4\x12\x17\xba\u0725\xe0\xe6\x03'\xd8E\xa3FO\x0f'\xf8J\x89\xd8\xd7&\xb7\x17z\x80\x0
0\x00\u07d4\a\xd43N\u00c5\xe8\xaaT\xee\xda\xea\xdb0\x02/\f\u07e4\xab\x89\x8e\x91\xd5
\xf2\xeby\x00\x00\u07d4\a\xda\xe6\"c\r\x1168\x193\u04adk\"\xb89\xd8!\x02\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\a\xdc+\xf8;\u01af\x19\xa8B\xff\xeaf\x1a\xf5\xb4\x1bg\xfd\xa1\x89QP\xae\x84\xa8\
xcd\xf0\x00\x00\u07d4\a\u070c\x8b\x92z\xdb\xed\xfa\x8f]c\x9bCR5\x1f/6\u0489\x11\n\xed;U0\xdb
\x00\x00\u07d4\a\xdd\xd0B,\x86\xefe\xbf\f\u007f\xc3E(b\xb1\"\x8b\b\xb8\x89o\xf5\u04aa\x8f\x9f\xc
f\x00\x00\u07d4\a\xe1\x16,\xea\xe3\xcf!\xa3\xf6-

\x10Y\x900.0\u007fN;\x89R\xf1\x03\xed\xb6k\xa8\x00\x00\u07d4\a\xe2\xb4\xcd\xee\xd9\u0407\xb1.Um\x9ew\f\x13\xc0\x99a_\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\a\xfe\xefT\xc16\x85\b)\xba\xdcKI\xc3\xf2\xa7<\x89\xfb\x9e\x89\x06hZ\xc1\xbf\xe3,\x00\x00\u07d4\b\x05FP\x8a=&\x82\u0239\x88O\x13c{\x88G\xb4M\xb3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\b\t\bv\xba\xad\xfe\xe6\\=6;\xa5S\x12t\x8c\xfa\x87=\x89*\x997\x1c\xff\xe1\x00\x00\u07d4\b\x16o\x021?\xea\u12f0D\xe7\x87|\x80\x8bU\xb5\xbfX\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\b)\xd0\xf7\xbb|Dl\xfb\xb0\u07ad\xb29M\x9d\xb7$\x9a\x87\x89\x02,\xa3X|\xf4\xeb\x00\x00\u07d4\b0m\xe5\x19\x81\u7b21\x85hY\xb7\xc7xijki\xf9\x89\xadx\xeb\u016cb\x00\x00\x00\xe0\x94\b7S\x9b_jR*H,\xdc\u04e9\xbbpC\xaf9\xbd\u048a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\b8\xa7v\x8d\x9c*\u028b\xa2y\xad\xfe\xe4\xb1\xf4\x91\xe3&\xf1\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\bA\x16R\xc8qq6\t\xaf\x00b\xa8\xa1(\x1b\xf1\xbb\xcf\u0649K\xe4\xe7&{j\xe0\x00\x00\xe0\x94\bM\x102Tu\x9b4<\xb2\xb9\xc2\xd8\xff\x9e\x1a\xc5\xf1E\x96\x8a\x01\x9b\xff/\xf5yh\xc0\x00\x00\u07d4\bPO\x05d?\xabY\x19\xf5\xee\xa5Y%\u05e3\xed}\x80z\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\b[J\xb7]\x83b\xd9\x14C\\\xed\xee\x1d\xaa+\x1e\xe1\xa2;\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\b[\xa6_\xeb\xe2>\xef\xc2\xc8\x02fj\xb1&#\x82\xcf\u0114\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\bt\x98\xc0FFh\xf3\x11P\xf4\xd3\u013c\u0765\"\x1b\xa1\x02\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\bw\uebabx\xd5\xc0\x0e\x83\xc3+-
\x98\xfay\xadQH/\x89\x17\xd2-
q\xdab&\x00\x00\u0794\b\x93j7\u07c5\xb3\xa1X\xca\xfd\x9d\xe0!\xf5\x817h\x13G\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\b\xa9\xa4N\x1fA\xde=\xbb\xa7\xa3c\xa3\xabA,\x12L\xd1^\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\b\xb7\xbd\u03d4MUp\x83\x8b\xe7\x04`$:\x86\x94HXX\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\b\xb8E6\xb7L\x8c\x01T=\xa8\x8b\x84\u05cb\xb9WG\xd8\"\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\b\xc2\xf26\xacJ\xdc\xd3\xfd\xa9\xfb\xc6\xe4S\"S\xf9\xda;\xec\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\b\xc8\x02\xf8wX4\x9f\xa0>k\xc2\xe2\xfd\a\x91\x19~\ua689lk\x93[\x8b\xbd@\x00\x00\u07d4\b\xc9\U0007fd89\xfd\xf8\x04\xd7i\xf8!#6\x02\x15\xaf\xf9;\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\b\xca\u0215&A\xd8\xfcRn\xc1\xabO-
\xf8&\xa5\xe7q\x0f\x89\x10CV\x1a\x88)0\x00\x00\xe0\x94\b\xcc\xdaP\xe4\xb2j\x0f\xfc\x0e\xf9.\x92\x051\a\x06\xbe\xc2\u01ca\x01Iul8W\xc6\x00\x00\x00\u07d4\b\u0406M\xc3/\x9a\xcb6\xbfN\xa4G\xe8\xddg&\x90j\x15\x89lnY\xe6|xT\x00\x00\u07d4\b\xd4&\u007f\xeb\x15\u0697\x00\xf7\xcc\xc3\xc8J\x89\x18\xbf\x17\xcf\u0789a\t=|,m8\x00\x00\xe0\x94\b\xd41\x1c\x9c\x1b\xba\xf8\u007f\xab\xe1\xa1\xd0\x14c\x82\x8d]\x98\u038a\x13\x0e\xe8\xe7\x17\x90D@\x00\x00\u07d4\b\xd5N\x83\xadHj\x93L\xfa\xea\u20e3>\xfd\"|\x0e\x99\x898S\x05\x83$^\xdc\x00\x00\u07d4\b\xd9~\xad\xfc\xb7\xb0d\xe1\xcc\xd9\u0217\x9f\xbe\xe5\xe7z\x97\x19\x89\x0el]\xa8\xd6z\xc1\x80\x00\u07d4\b\xda:z\x0fE!a\u03fc\xec1\x1b\xb6\x8e\xbf\xde\xe1~\x88\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\b\xe3\x8e\xe0\xceH\xc9\xcad\\\x10\x19\xf7;SUX\x1cV\xe6\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4\b\xef?\xa4\xc4<\xcd\xc5{\"\xa4\xb9\xb23\x1a\x82\xe58\x18\xf2\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\t\td\x8c\x18\xa3\xce[\xaez\x04~\xc2\xf8h\xd2L\u0768\x1d\x89\xcf\x15&@\xc5\xc80\x00\x00\u07d4\t\f\xd6{`\xe8\x1dT\xe7\xb5\xf6\a\x8f>\x02\x1b\xa6[\x9a\x1e\x8965\u026d\xc5\u07a0\x00\x00\u07d4\t\f\xeb\xef),>\xb0\x81\xa0_\u062a\xf7\u04db\xf0{\x89\u0509\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\t\x0f\xa96{\xdaW\xd0\xd3%:\n\x8f\xf7l\xe0\xb8\xe1\x9as\x8965\u026d\xc5\u07a0\x00\x00\u07d4\t\x14n\xa3\x88Qv\xf0w\x82\xe1\xfe0\xdc\xe3\xce$\u011e\x1f\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d

4\t!`_\x99\x16N;\xcc(\xf3\x1c\xae\xcex\x971\x82V\x1d\x89+\ai*\x90e\xa8\x00\x00\xe0\x94\t&\x1f\x9a\xcbE\x1c7\x88\x84O\f\x14Q\xa3[\xadP\x98\xe3\x8a\x01\u056d'P)
`\x00\x00\xe0\x94\t'\"\x04\x92\x19K.\u069f\u013b\xe3\x8f%\u0581\xdf\xd3l\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u0794\t*\xcbbK\b\xc0U\x10\x18\x9b\xbb\xe2\x1ee$\xd6D\u032d\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\t.\x81UX@-
g\xf9\rk\xfem\xa0\xb2\xff\xfa\x91EZ\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\tP0\xe4\xb8&\x92\xdc\xf8\xb8\u0411$\x94\xb9\xb3x\xec\x93(\x89H\xa4<T`/p\x00\x00\u07d4\tRp\xccB\x14\x1d\u0658\xad(b\xdb\xd1\xfe\x9bD\xe7\xe6P\x89A\rXj
\xa4\xc0\x00\x00\u07d4\tTW\xf8\xef\x8e+\xdc6!\x96\xb9\xa9\x12]\xa0\x9cg\u3ac9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\tT\xa8\xcb]2\x1f\xc35\x1au#\xa6\x17\xd0\xf5\x8d\xa6v\xa7\x89\x87\u067cz\xa4\x98\xe8\x00\x00\xe0\x94\t[\x0e\xa2\xb2\x18\xd8.\n\xea|(\x89#\x8a9\u027f\x90w\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\t[\x94\x9d\xe33:7}P\x19\u0613uJ^FV\xff\x97\x89\x12nr\xa6\x9aP\xd0\x00\x00\xe0\x94\t^\x01t\x82\x9f4\xc3x\x1b\xe1\xa5\xe3\x8d\x15A\xeaC\x9b\u007f\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\t_ZQ\xd0oc@\xd8\vm)\xea.\x88\x11\x8a\xd70\xfe\x89lnY\xe6|xT\x00\x00\u07d4\th\xeeZ7\x8f\x8c\xad\xb3\xba\xfd\xbe\xd1\u045a\xaa\u03d3g\x11\x8965\u026d\xc5\u07a0\x00\x00\u07d4\tw\xbf\xba\x03\x8aD\xfbI\xb09p\xd8\xd8\xcf,\xb6\x1f\x8b%\x89\x16\u012b\xbe\xbe\xa0\x10\x00\x00\u07d4\t}\xa1,\xfc\x1f|\x1a$d\xde\xf0\x8c)\xbe\xd5\xe2\xf8Q\xe9\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\t~\u0362%g\xc2\xd9\x1c\xb0?\x8cR\x15\xc2.\x9d\u0369I\x89\x01\x16Q\xac>zu\x80\x00\u07d4\t\x89\xc2\x00D\v\x87\x89\x91\xb6\x9d`\x95\xdf\xe6\x9e3\xa2.p\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\t\x90\xe8\x1c\u05c5Y\x9e\xa26\xbd\x19f\xcfRc\x02\xc3[\x9c\x8965\u026d\xc5\u07a0\x00\x00\u07d4\t\x98\xd8'1\x15\xb5j\xf4<P^\bz\xff\x06v\xed6Y\x89\xd8\xd6\xed\xdf-.\x18\x00\x00\u07d4\t\xa0%1o\x96\u007f\xa8\xb9\xa1\xd6\a\x00\x06?Zh\x00\x1c\xaa\x89\x02\x12!\xa9\x9b\x93\xec\x00\x00\u0794\t\xa9(\xd5(\xec\x1b>%\xff\xc8>!\x8c\x1e\n\xfe\x89(\u01c8\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\t\xaeI\xe3\u007f\x12\x1d\xf5\xdc\x15\x8c\xfd\xe8\x06\xf1s\xa0k\f\u007f\x89\xd80\x9e&\xab\xa1\xd0\x00\x00\u07d4\t\xaf\xa7;\xc0G\xefF\xb9w\xfd\x97c\xf8r\x86\xa6\xbeh\u0189\x1b/\xb5\xe8\xf0jf\x00\x00\u07d4\t\xb4f\x86\x96\xf8j\b\x0f\x8b\xeb\xb9\x1d\xb8\xe6\xf8p\x15\x91Z\x89#\x8f\xf7\xb3O`\x01\x00\x00\xe0\x94\t\xb5\x9b\x86\x98\xa7\xfb\xd3\xd2\xf8\xc7:\x00\x89\x88\xde>@k+\x8a\bxg\x83&\xea\xc9\x00\x00\x00\xe0\x94\t\xb7\xa9\x88\xd1?\xf8\x91\x86so\x03\xfd\xf4au\xb5=\x16\xe0\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\t\xc1w\xf1\xaeD$\x11\u076c\xf1\x87\xd4m\xb9V\x14\x83`\xe7\x8a\x01\xe5.3l\xde\"\x18\x00\x00\xe0\x94\t\u020f\x91~Mj\xd4s\xfa\x12\u93a3\xc4G*^\xd6\u068a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\t\u0438\xcd\a|i\xd9\xf3-
\x9c\xcaC\xb3\xc2\b\xa2\x1e\u050b\x89\b!\xd2!\xb5)\x1f\x80\x00\xe0\x94\t\xd6\xce\xfdu\xb0\u0133\xf8\xf1\u0587\xa5\"\xc9a#\xf1\xf59\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\t\xe47\xd4H\x86\x12(\xa22\xb6.\xe8\xd3ye\xa9\x04\ud70a\x04\x98\xcf@\x1d\xf8\x84.\x80\x00\u07d4\t\xee\x12\xb1\xb4+\x05\xaf\x9c\xf2\a\xd5\xfc\xac%[.\xc4\x11\xf2\x89\x031\xcd\xddG\xe0\xfe\x80\x00\u07d4\t\xf3\xf6\x01\xf6\x05D\x11@Xl\xe0eo\xa2J\xa5\xb1\u066e\x89Sswo\xe8\xc4T\x00\x00\u07d4\t\xf9W[\xe5}\x00G\x93\u01e4\ub137\x15\x87\xf9|\xbbj\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\n\x06P\x86\x1fx^\xd8\xe4\xbf\x10\x05\xc4P\xbb\xd0n\xb4\x8f\xb6\x89\xa6A;y\x14N~\x00\x00\u07d4\n\x06\xfa\xd7\xdc\u05e4\x92\xcb\xc0S\xee\xab\xdei4\xb3\x9d\x867\x89\x01\x1

5\x8eF\t\x13\xd0\x00\x00\u07d4\n\a}\xb1?\xfe\xb0\x94\x84\xc2\x17p\x9dX\x86\xb8\xbf\x9cZ\x8b\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\n\x0e\u0366cow\x16\xef\x19saF\x87\xfd\x89\xa8
\xa7\x06\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\n)\xa8\xa4\xd5\xfd\x95\x00u\xff\xb3Mw\xaf\xeb-\x82;\u0589\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\n*\u0795\xb2\xe8\xc6m\x8a\xe6\xf0\xbad\xcaW\u05c3\xbemD\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\n+O\xc5\xd8\x1a\xceg\xdcK\xba\x03\xf7\xb4UA=F\xfe=\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\n-
\xcbzg\x17\x01\u06f8\xf4\x95r\x80\x88&Xs5l\x8e\x89\b?\x16\xce\b\xa0l\x00\x00\u07d4\n=\xe1U\xd5\xec\xd8\xe8\x1c\x1f\xf9\xbb\xf07\x83\x01\xf8\xd4\xc6#\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\nG\xad\x90Y\xa2I\xfc\x93k&b5=\xa6\x90_u\u00b9\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\nH)ov1p\x8c\x95\u04b7Iu\xbcJ\xb8\x8a\xc19*\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x94\nJ\x01\x19\x95\u0181\xbc\x99\x9f\xddyuN\x9a2J\xe3\xb3y\x8a\b\xc1\x9a\xb0n\xb8\x9a\xf6\x00\x00\u07d4\nX\xfd\xddq\x89\x8d\xe7s\xa7O\xda\xe4^{\xd8N\xf46F\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\n[y\xd8\xf2;d\x83\xdb\u2f6ab\xb1\x06L\xc7cf\xae\x89j\u0202\x10\tR\u01c0\x00\u07d4\ne.*\x8bw\xbd\x97\xa7\x90\xd0\xe9\x13a\u0248\x90\u06f0N\x8965\u026d\xc5\u07a0\x00\x00\u07d4\nn\xber;n\xd1\xf9\xa8ji\xdd\xdah\xdcGF\\+\x1b\x89@=-
\xb5\x99\xd5\xe4\x00\x00\u07d4\nw\xe7\xf7+C{WO\x00\x12\x8b!\xf2\xac&Q3R\x8c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\n\x91\u007f;\\\xb0\xb8\x83\x04\u007f\u0676Y=\xbc\xd5W\xf4S\xb9\x8965\u026d\xc5\u07a0\x00\x00\u07d4\n\x93\x1bD\x9e\xa8\xf1,\xdb\xd5\xe2\xc8\xccv\xba\xd2\xc2|\x069\x89\x01?\x9e\x8cy\xfe\x05\x80\x00\u0794\n\x98\x04\x13x\x03\xbahh\xd9:U\xf9\x98_\xcdT\x04Q\u4239\x8b\xc8)\xa6\xf9\x00\x00\u07d4\n\x9a\xb2c\x8b\x1c\xfdeM%\u06b0\x18\xa0\xae\xbd\u07c5\xfdU\x89\x01.\x8c\xb5\xfeLJ\x80\x00\u07d4\n\xb3f\xe6\xe7\u056b\xbc\xe6\xb4JC\x8di\xa1\u02bb\x90\xd13\x89\x11X\xe4`\x91=\x00\x00\x00\u07d4\n\xb4(\x1e\xbb1\x85\x90\xab\xb8\x9a\x81\xdf\a\xfa:\xf9\x04%\x8a\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u0794\n\xb5\x9d9\a\x02\xc9\xc0Y\xdb\x14\x8e\xb4\xf3\xfc\xfa}\x04\xc7\xe7\x88\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94\n\xbf\xb3\x9b\x11HmyW(f\x19[\xa2Ic\vg\x84\u06ca\x19\xba\x877\xf9i(\xf0\x00\x00\u07d4\n\u029aV&\x91;\b\xcf\u0266m@P\x8d\xceR\xb6\x0f\x87\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\n\xd3\xe4M<\x00\x1f\xa2\x90\xb3\x93ap0TA\b\xacn\xb9\x89j\xbd\xa0\xbc0\xb2\u07c0\x00\u07d4\n\xec.Bn\xd6\xcc\f\xf3\xc2I\xc1\x89~\xacG\xa7\xfa\xa9\xbd\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\n\xf6_\x14xNU\xa6\xf9Vg\xfds%*\x1c\x94\a-
\x89\nv;\x8e\x02\xd4O\x80\x00\u07d4\n\xf6\xc8\xd59\xc9mP%\x9e\x1b\xa6q\x9e\x9c\x80`\xf3\x88\u008965\u026d\xc5\u07a0\x00\x00\u07d4\v\x069\x0f$7\xb2\x0e\u0123\xd3C\x1b2y\xc6X>^\u05c9\n\x84Jt$\xd9\xc8\x00\x00\u07d4\v\v8b\x11\xee\u00e04\x92\xb1\xb0_D\x0e\xcaT%n\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\v\x0e\x05[(\xcb\xd0=\xc5\xffD\xaad\xf3\xdc\xe0O^c\xfb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\v\x11\x9d\xf9\x9ck\x8d\xe5\x8a\x1e,?)zgD\xbfU\"w\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\v\x14\x89\x19\x99\xa6\\\x9e\xf73\b\xef\xe3\x10\f\xa1\xb2\x0e\x81\x92\x89+^:\xf1k\x18\x80\x00\x00\u07d4\v!\x13PE4d*\x1d\xaf\x10.\xee\x10\xb9\xeb\xdev\xe2a\x89\x94,\xdd|\x95\xf2\xbd\x80\x00\xe0\x94\v(\x8aZ\x8bu\xf3\xdcA\x91\xeb\x04W\xe1\xc8=\xbd
M%\x8a\x01\a\x14\xe7{\xb4:\xb4\x00\x00\u07d4\v6\x9e\x00.\x1bLy\x13\xfc\xf0\x0f-
^\x19\u0141eG\x8f\x89\x03\u007fe\x16(\x8c4\x00\x00\u07d4\vC\xbd#\x91\x02U\x81\u0615l\xe4*\a%y\u02ff\xcb\x14\x89\x01\x04\xe7\x04d\xb1X\x00\x00\u07d4\vP|\xf5SV\x8d\xaa\xf6U\x04\xaeN\xaa\x17\xa8\xea<\xdb\xf5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\v]f\xb1<\x87\xb3\x92\xe9M\x91\xd

5\xf7l\rE\nU(C\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\v^
\x11\xeb\xc2Z\x00\u007f!6)`I\x8a\xfb\x8a\xf2\x80\xfb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\vd\x9d\
xa3\xb9j\x10,\xdcm\xb6R\xa0\xc0}e\xb1\xe4C\xe6\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\vi
\xa6K6;\x8d]\x90\x80$\x94\xcfVKT|C\r\x89A\rXj
\xa4\xc0\x00\x00\u07d4\vp\x11\x01\xa4\x10\x9f\x9c\xb3`\xdcW\xb7tBg=^Y\x83\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4\vq\xf5T\x12$i\uf5ce/\x1f\xef\xd7\u02f4\x10\x98'r\x89\xd2U\xd1\x12\xe1\x03\xa0\
x00\x00\xe0\x94\v{\xb3B\xf0\x1b\u0248\x8ej\x9a\xf4\xa8\x87\xcb\xf4\xc2\xdd,\xaf\x8a\x03c\\\x9a\x
dc]\xea\x00\x00\x00\u07d4\v}3\x93q\xe5\xbeg'\xe6\xe31\xb5\x82\x1f\xa2K\u06ddZ\x89.\u007f\x81
\x86\x82b\x01\x00\x00\u07d4\v\u007f\xc9\xdd\xf7\x05v\xf63\x06i\xea\xaaq\xb6\xa81\xe9\x95(\x89\
a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4\v\x80\xfcp(,\xbd\xd5\xfd\xe3[\xf7\x89\x84\xdb;\xdb\x12\x01
\x88\x8968\x02\x1c\xec\u06b0\x00\x00\u07d4\v\x92M\xf0\a\xe9\xc0\x87\x84\x17\xcf\xe6;\x97n\xa
1\xa3\x82\xa8\x97\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\v\x93\xfc\xa4\xa4\xf0\x9c\xac
\xdb`\xe0e\xed\xcc\xcc\x11\u0976\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\v\x9d\xf8\x0f\xbe#
\t\xda\xcf\n\xa8\xca\u0153v\xe2Gb\x03\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\v\xa6\xe4j\xf2Z\x13\x
f5qi%Z4\xa4\xda\xc7\xce\x12\xbe\x04\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\v\xa8p[\xf5\\\
xf2\x19\xc0\x95k^?\xc0\x1cDt\xa6\xcd\xc1\x89\x05%\xe0Y]Mk\x80\x00\u07d4\v\xafn\u0379\x1a\xc
b6\x06\xa85|\v\xc4\xf4\\\xfd-
~o\x8965\u026d\xc5\u07a0\x00\x00\u07d4\v\xb0_r$\xbbX\x04\x85eV\xc0~\xea\xdb\ud1fa\x8f|\x89
\x15\xbeat\xe1\x91.\x00\x00\u07d4\v\xb0\xc1&\x82\xa2\xf1\\\x9bWA\xb28\\\xbeA\xf04\x06\x8e\x89
QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4\v\xb2\\\xa7\u0448\xe7\x1eMi={\x17\a\x17\xd6\xf8\xf0\xa7\
n\x89\x12C\x02\xa8/\xad\xd7\x00\x00\u07d4\v\xb2e\x0e\xa0\x1a\xcau[\xc0\xc0\x17\xb6K\x1a\xb5\
xa6m\x82\xe3\x89Hz\x9a0E9D\x00\x00\u07d4\v\xb5Lr\xfdf\x10\xbf\xa463\x97\xe0
8K\x02+\fl\x89Hz\x9a0E9D\x00\x00\u07d4\v\xb7\x16\n\xba)7b\xf8sO>\x03&\xff\u0264\xca\xc1\x90
\x8965\u026d\xc5\u07a0\x00\x00\u07d4\v\xc9\\\xb3-
\xbbWL\x83/\xa8\x17J\x815m8\xbc\x92\xac\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\v\xd6}\xbd\xe0z\
x85n\xbd\x89;^\xdcO:[\xe4
&\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\v\xdb\xc5L\u023d\xbb\xb4\x02\xa0\x89\x11\xe2#*T`\u
0386k\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\v\xddX\xb9n|\x91m\xd2\xfb05o*\xeb\xfa\xaf\x1d\x8
60\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\v\u1f39\x03C\xfa\xe501s\xf4a\xbd\x91JH9\x05l\x8a\x01
EB\xba\x12\xa37\xc0\x00\x00\u07d4\v\xe1\xfd\xf6&\xeea\x89\x10-
p\xd1;1\x01,\x95\xcd\x1c\u0589lk\x93[\x8b\xbd@\x00\x00\u07d4\v\xe2\xb9J\xd9P\xa2\xa6&@\xc3
[\xfc\xcdlg\xda\xe4P\xf6\x89i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94\v\u681eC\a\xfeH\xd4\x12\xb8\u0
461\xa8(M\xceHba\x8a\x04\x0f\xbf\xf8\\\x0180\x00\x00\u07d4\v\xef\xb5G\a\xf6\x1b,\x9f\xb0G\x15\
xab\x02n\x1b\xb7
B\xbd\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\v\xf0dB\x8f\x83bg\"\xa7\xb5\xb2j\x9a\xb2\x04!\x
a7r>\x89\a?u\u0460\x85\xba\x00\x00\u07d4\v\xfb\xb6\x92]\xc7^R\xcf&\x84\"K\xbe\x05P\xfe\xa6\x
85\u04c9j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\f\b\x80\x06\xc6K0\xc4\u076f\xbc6\xcb_\x05F\x9e\xb6(
4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\f
s\xbaD\xd3\u077d\xb69\xc0N\x19\x109\xa7\x17\x16#\u007f\x89M\x85<\x8f\x89\b\x98\x00\x00\xe0
\x94\f\",|A\u0270H\xef\xcc\xe0\xa22CCb\xe1-
g;\x8a\x02\x1e\x83Yivw8\x00\x00\xe0\x94\f(\b\xb9Q\ud787-
{2y\x0f\xccY\x94\xaeA\xff\u070a\x15\x99n[<\u05b3\xc0\x00\x00\u07d4\f(\x84~O\t\xdf\xce_\x9b%\x

af|NS\x0fY\u0200\xfe\x8965\u026d\xc5\u07a0\x00\x00\u07d4\f-
\\\x92\x058\xe9S\u02af$\xf0s\u007fUL\u0192wB\x8965\u026d\xc5\u07a0\x00\x00\u07d4\f0\xca\xc
c?r&\x9f\x8bO\x04\xcf\a=+\x05\xa8=\x9a\u0449lyt\x12?d\xa4\x00\x00\u07d4\f29\xe2\xe8A$-
\xb9\x89\xa6\x15\x18\xc2\"G\xe8\xc5R\b\x89\x0eJ\xf6G\x174d\x00\x00\xe0\x94\fH\r\xe9\xf7F\x10\
x02\x90\x8bI\xf6\x0f\xc6\x1e+b\xd3\x14\v\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\fH\xa
eb\xd1S\x97\x88\xeb\xa0\x13\xd7^\xa6\vd\xee\xbaN\x80\x89w\xfb\xdcC\xe00\x99\x80\x00\u07d4\
fU\x89\xa7\xa8\x9b\x9a\xd1[\x02u\x190AYH\xa8u\xfb\xef\x89\x06\u0519\xeclc8\x00\x00\u07d4\fg\
x03=\xd8\xee\u007f\f\x8a\xe54\xd4*Q\xf7\xd9\xd4\xf7\x97\x8f\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\fhE\xbfA\xd5\xee'<>\u6d70\u059fo\xd5\xea\xbb\xf7\x89\xa2\xa1\xb9h.X\t\x00\x00\
xe0\x94\f\u007f\x86\x9f\x8e\x90\xd5?\xdc\x03\u8c81\x9b\x01k\x9d\x18\xeb&\x8a\x04<3\xc1\x93u
d\x80\x00\x00\u07d4\f\x86\x92\xee\xff*S\xd6\xd1h\x8e\xd5j\x9d\u06fdh\u06bb\xa1\x89lk\x93[\x8b\
xbd@\x00\x00\u07d4\f\x8ff\xc6\x01{\xce[
4r\x04\xb6\x02\xb7C\xba\u05cd`\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\f\x8f\xd7w^T\xa6\xd9\u02
63\xbf\x89\x0ev\x1fewi?\xf0\x8a\x02\x15\xf85\xbcv\x9d\xa8\x00\x00\u07d4\f\x92Z\xd5\xeb5,\x8e\x
f7m\f\"-
\x11[\a\x91\xb9b\xa1\x89\xacc]\u007f\xa3N0\x00\x00\u07d4\f\x96~0a\xb8zu>\x84P~\xb6\t\x86x,\x
8f0\x13\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\f\xa1*\xb0\xb9fl\xf0\xce\xc6g\x1a\x15)/&SGj\xb2\x8a,x'\xc4-
\"\xd0|\x00\x00\u07d4\f\xa6p\xeb,\x8b\x96\u02e3y!\u007fY)\u00b8\x92\xf3\x9e\xf6\x89lk\x93[\x8b\
xbd@\x00\x00\u07d4\f\xae\x10\x8em\xb9\x9b\x9ecxv\xb0d\xc60>\u068ae\u0209\xa2\xa1]\tQ\x9b\
xe0\x00\x00\u07d4\f\xbd\x92\x1d\xbe\x12\x15c\xb9\x8ahq\xfe\xcb\x14\xf1\xcc~\x88\u05c9\n\u05c
e\xbcZ\xc6 \x00\x00\u07d4\f\xbf\x87p\xf0\xd1\b.\\
\u016e\xad4\xe5\xfc\xa9\xaez\xe2\x8965\u026d\xc5\u07a0\x00\x00\u07d4\f\xc6\u007f\x82s\xe1\x
ba\xe0\x86\u007f\xd4.\x8b\x81\x93\xd7&y\xdb\xf8\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\f\
u05a1A\x91\x8d\x12k\x10m\x9f.\xbfi\xe1\x02\xdeM2w\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x9
4\f\xda\x12\xbfr\xd4a\xbb\xc4y\xeb\x92\xe6I\x1d\x05~kZ\u044a\x02\x1e\x19\xe0\u027a\xb2@\x00
\x00\u07d4\f\u0716\v\x99\x8c\x14\x19\x98\x16\r\xc1y\xb3l\x15\u0484p\xed\x89\x1b\x1bk\u05efd\x
c7\x00\x00\xe0\x94\f\xfb\x17#5\xb1l\x87\xd5\x19\xcd\x14uS\r W\u007f^\x0e\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94\r\x1f*Wq>\xbcn\x94\xde)\x84n\x88D\xd3vfWc\x8a\x01\x0f\f
\xf0d\xddY
\x00\x00\u07d4\r2e\xd3\u7f79=^\x8e\x8b\x1c\xa4\u007f!\ny>\u030e\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\r5@\x8f\"ef\x11o\xb8\xac\u06a9\xe2\xc9\u055bvh?\x892\xf5\x1e\u06ea\xa30\x00\
x00\u07d4\rU\x1e\xc1\xa2\x13<\x98\x1d_\u01a8\xc8\x17?\x9e|OG\xaf\x89lk\x93[\x8b\xbd@\x00\x
00\u07d4\r]\x98V\\d|\xa5\xf1w\xa2\xad\xb9\xd3\x02/\xac(\u007f!\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\re\x80\x14\xa1\x99\x06\x1c\xf6\xb3\x943\x14\x03\x03\xc2\x0f\xfdNZ\x8a\x01\xb
c\x85\xdc*\x89\xbb
\x00\x00\u07d4\rg\x87\x06\xd07\x18\u007f>\"\xe6\xf6\x9b\x99\xa5\x92\xd1\x1e\xbcY\x89U\xa6\xe
7\x9c\xcd\x1d0\x00\x00\u07d4\ri\x10\f9\\\xe6\xc5\xea\xad\xf9]\x05\xd8r\x83~\xde\xdd!\x89\x15\xaf
\x1dx\xb5\x8c@\x00\x00\xe0\x94\rt~\u559b\xf7\x9dW8\x1do\xe3\xa2@l\xd0\xd8\xce'\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\r\x80#\x92\x9d\x91r4\xae@Q+\x1a\xab\xb5\xe8\xa4Q'q\x89\
b\x05\xe9\x9f\xdc\xc5\xd0\x00\x00\xe0\x94\r\x8a\xab\x8ft\xea\x86,\xdfvh\x05\x00\x9d?>B\xd8\xd0
\v\x8a\x01;\x80\xb9\x9cQ\x85p\x00\x00\u07d4\r\x8c@\xa7\x9e\x18\x99O\xf9\x9e\xc2Q\xee\x10\u

0408\u00d1.\x80\x89\x066d\xfc\u04bb\xc4\x00\x00\u07d4\r\x8e\xd7\xd0\xd1V83\x0e\xd7\xe4\xea\u032b\x8aE\x8dus~\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\r\x92X/\u06e0^\xab\xc3\xe5\x158\xc5m\xb8\x817\x85\xb3(\x89\nZ\xa8P\t\xe3\x9c\x00\x00\u07d4\r\x94C\xa7\x94h\xa5\xbb\xf7\xc1<n\"]\x1d\xe9\x1a\xee\a\u07c9\x03\xcbq\xf5\x1f\xc5X\x00\x00\xe0\x94\r\x9a\x82_\xf2\xbc\u04d7\u02ed[q\x1d\x9d\u0315\xf1\xcc\x11-
\x8a\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\x00\u07d4\r\x9d?\x9b\u0124\xc6\xef\xbdYg\x9bi\x82k\xc1\xf6=\x99\x16\x89
\x86\xac5\x10R`\x00\x00\u07d4\r\xa52\xc9\x10\xe3\xac\r\xfb\x14\xdba\xcds\x9a\x935?\xd0_\x89Hx\xbe\x1f\xfa\xf9]\x00\x00\u07d4\r\xa7@\x12b8N.\x8bK&\xdd\x15G\x99\xb5QE\uf809\x10CV\x1a\x88)0\x00\x00\u07d4\r\xae>\xe5\xb9\x15\xb3d\x87\xf9\x16\x1f\x19\x84m\x10\x1431\x8a\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\r\xbdA|7+\x8b\r\x01\xbc\xd9Dpk\xd3.`\xae(\u0449\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\r\xc1\x00\xb1\a\x01\x1c\u007f\xc0\xa13\x96\x12\xa1l\xce\xc3(R\b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\r\u03dd\x8c\x98\x04E\x9fd|\x14\x13\x8e\xd5\x0f\xadV;AT\x89\t`\xdbwh\x1e\x94\x00\x00\u07d4\r\xcf\xe87\xea\x1c\xf2\x8ce\xfc\xce\u00fe\xf1\xf8NY\xd1P\xc0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\r\xd4\xe6t\xbb\xad\xb1\xb0\u0702D\x98q=\xce;QV\xda)\x89\t79SM(h\x00\x00\u07d4\r\xfb\u0501pP\xd9\x1d\x9db\\\x02\x05<\xf6\x1a>\xe2\x85r\x89\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\x0e\x02N\u007f\x02\x9cj\xaf:\x8b\x91\x0f^\b\bs\xb8W\x95\xaa\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x0e\tdl\x99\xafC\x8e\x99\xfa'L\xb2\xf9\xc8V\xcbe\xf76\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\x0e\f\x9d\x00^\xa0\x16\u0095\xcdy\\\xc9!>\x87\xfe\xbc3\xeb\x89\n\xbb\xcdN\xf3wX\x00\x00\u07d4\x0e\rf3\xdb\x1e\f\u007f#Jm\xf1c\xa1\x0e\n\xb3\x9c
\x0f\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x0e\x11\xd7z\x89w\xfa\xc3\r&\x84E\xe51\x14\x9b1T\x1a$\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x0e\x12=}\xa6\xd1\xe6\xfa\xc2\u072d\xd2p)$\v\xb3\x90R\xfe\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x0e\x18\x01\xe7\vbb\x86\x1b\x114\u033c9\x1fV\x8a\xfc\x92\xf7\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x0e
\x94\xac\x16T\xa4k\xa1\xc4\u04e4\v\xb8\xc1}\xa7\U000d6209\x13h?\u007f<\x15\xd8\x00\x00\u07d4\x0e!\xaf\x1b\x8d\xbf'\xfc\xf6?7\xe0G\xb8z\x82\\\xbe|'\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x0e.PJ-
\x11\"\xb5\xa9\xfe\xee\\\xb1E\x1b\xf4\u00ac\xe8{\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x0e/\x8e(\xa6\x81\xf7|X;\xd0\xec\xde\x16cK\xdd~\x00\u0349\x05'8\xf6Y\xbc\xa2\x00\x00\u07d4\x0e2\x02\x19\x83\x8e\x85\x9b/\x9f\x18\xb7.=@s\xcaP\xb3}\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x0e3\xfc\xbb\xc0\x03Q\v\xe3W\x85\xb5*\x9c]!k\xc0\x05\xf4\x89e\xea=\xb7UF`\x00\x00\u07d4\x0e6\x96\xcf\x1fB\x17\xb1c\u047c\x12\xa5\xeas\x0f\x1c2\xa1J\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x0e9\x0fD\x05=\xdf\xce\xf0\xd6\b\xb3^M\x9c,\xbe\x98q\xbb\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x0e:(\xc1\u07ef\xb0P[\xdc\xe1\x9f\xe0%\xf5\x06\xa6\xd0\x1c\xeb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x0e=\xd7\xd4\xe4)\xfe90\xa6A@5\xf5+\xdcY\x9dxM\x89\x02,\xa3X|\xf4\xeb\x00\x00\u07d4\x0eGey\x03Rek\xc6Vh,$\xfc^\xf3\xe7j#\u01c9\x02\x86\xd7\xfc\f\xb4\xf5\x00\x00\u07d4\x0eI\x88\x00Dqw\xb8\u022f\xc3\xfd\xfa\u007fi\xf4\x05\x1b\xb6)\x89t\x05\xb6\x9b\x8d\xe5a\x00\x00\u07d4\x0ek\xaa\xa3\u07b9\x89\xf2\x89b\x00vf\x86\x18\xe9\xac3(e\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x0el\xd6d\xad\x9c\x1e\xd6K\xf9\x87I\xf4\x06D\xb6&\xe3y,\x8a\f\xb4\x9bD\xba`-

\x80\x00\x00\xe0\x94\x0em\xfdU;.\x87=*\xec\x15\xbd_\xbb?\x84r\xd8\u04d4\x8a\x02\x8a\x85t%F
o\x80\x00\x00\u07d4\x0en\xc3\x137bq\xdf\xf5T#\xabT\"\xcc:\x8b\x06\xb2+\x89\xd8\xd7&\xb7\x17z
\x80\x00\x00\xe0\x94\x0en\u0399\x11\x1c\xad\x19a\xc7H\xed=\xf5\x1e\xddi\u04a3\xb1\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\x0e\x83\xb8PH\x1a\xb4MI\xe0\xa2)\xa2\xe4d\x90,iS\x9b\x8
9\x05k\xc7^-
c\x10\x00\x00\u07d4\x0e\x89\xed\xdd?\xa0\xd7\x1d\x8a\xb0\xff\x8d\xa5X\x06\x86\xe3\xd4\xf7O\x
89lk\x93[\x8b\xbd@\x00\x00\u07d4\x0e\x90\x96\xd3C\xc0`\xdbX\x1a\x12\x01\x12\xb2x`~\xc6\xe5
+\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\x0e\x9cQ\x18d\xa1w\xf4\x9b\xe7\x82\x02w?`H\x9f
\xe0NR\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\x0e\xa2\xa2\x101+>\x86~\xe0\xd1\xcch,\xe
1\xd6f\xf1\x8e\u054a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x0e\xb1\x89\xef,-
Wb\xa9c\u05b7\xbd\xf9i\x8e\xa8\u7d0a\x89Hz\x9a0E9D\x00\x00\xe0\x94\x0e\xb5\xb6b\xa1\xc7\x
18`\x8f\xd5/\f%\xf97\x880\x17\x85\x19\x8a\x01J7(\x1aa.t\x00\x00\xe0\x94\x0e\xc4f\x96\xff\xac\x1f
X\x00_\xa8C\x98$\xf0\x8e\xed\x1d\xf8\x9b\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x0
e\xc5\n\xa8#\xf4e\xb9FK\v\xc0\u0125w$\xa5U\xf5\u058a\f\x83\xd1Bj\u01f1\xf0\x00\x00\u07d4\x0e
\xc50\x8b1(.!\x8f\xc9\xe7Y\xd4\xfe\xc5\xdb7\b\xce\u01096C\xaady\x86\x04\x00\x00\u07d4\x0e\xc
c\xf6\x17\x84O\xd6\x1f\xbab\xcb\x0eD[z\u018b\xcc\x1f\xbe\x89\x14\xfeO\xe65e\xc6\x00\x00\u07d
4\x0e\u04fb:N\xb5T\xcf\u0297\x94}WU\a\xcd\xfdm!\u0609\x1d\xb3
_\xcc#\u0540\x00\u07d4\x0e\xd7l,;]P\xff\x8f\xb5\v>\xea\xcdh\x15\x90\xbe\x1c-\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x0e\u0680\xf4\xed\aJ\xeaiz\xed\xdf(;c\xdb\xca=\xc4\u0689lk\x93[\x8b\xbd@\
x00\x00\u07d4\x0e\xddKX\x0f\xf1\x0f\xe0lJ\x03\x11b9\xef\x96b+\xae5\x89\n\xad\xec\x98?\xcf\xf4\
x00\x00\u07d4\x0e\xe3\x91\xf0<v[\x11\u0590&\xfd\x1a\xb3S\x95\xdc8\x02\xa0\x89\n\u05ce\xbcZ\
xc6
\x00\x00\u07d4\x0e\xe4\x14\x94\x04\x87\xfd$\xe3\x907\x82\x85\xc5\u05f93M\x8be\x89\x91Hx\xa
8\xc0^\xe0\x00\x00\u07d4\x0e\xf5J\xc7&M\"T\xab\xbb_\x8bA\xad\u0787QW\xdb|\x89\x02+\x1c\x8
c\x12'\xa0\x00\x00\u07d4\x0e\xf8[l\u040au\x19\x86\x92\x91N\u0774\xb2,\xf5\xfaDP\x89l\xae0b\x1
dG
\x00\x00\u07d4\x0e\xfd\x17\x89\xeb\x12D\xa3\xde\xde\x0f]\xe5\x82\u0616<\xb1\xf3\x9f\x89QP\xa
e\x84\xa8\xcd\xf0\x00\x00\xe0\x94\x0f\x04,\x9c/\xb1\x87f\xf86\xbbY\xf75\xf2}\xc3)\xfe<\x8a\x02\x
1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x0f\x04\x9a\x8b\xdf\xd7a\u078e\xc0,\xee()\xc4\x00[#\xc0
k\x89\r\xa93\xd8\xd8\xc6p\x00\x00\xe0\x94\x0f\x05\xf1
\u021e\x9f\xbc\x93\u052b\f^+J\r\xf0\x92\xb4$\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4\x0f\x12{\x
bf\x8e1\x1c\xae\xa2\xbaP*3\xfe\xce\xd3\xf70\xbaB\x89\n1\x06+\xee\xedp\x00\x00\u07d4\x0f\x1c$
\x9c\xd9b\xb0\x0f\xd1\x14\xa94\x9fjl\xc7x\xd7lM\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x0f
n\x1a\x1d\xa7 ~\xa5\x18\xb1\x12A\x8b\xaa\x8b\x06&\x03(\x89
\x86\xac5\x10R`\x00\x00\u07d4\x0f$\x10Z\xbb\u06a0?\xa60\x9e\xf6\xc1\x88\xe5\x1fqJnY\x89\n\u
05ce\xbcZ\xc6
\x00\x00\u07d4\x0f&H\n\x15\ta\xb8\xe3\aPq:\x94\xeeo.G\xfc\x00\x8965\u026d\xc5\u07a0\x00\x00
\xe0\x94\x0f-
\x8d\xaf\x04\xb5AJ\x02a\xf5I\xffdw\xb8\x0f/\x1d\a\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4\x0f
/\xb8\x84\u022a\xffoT:\xc6\"\x8b\u040eO`\xb0\xa5\xfd\x89\xaa}\xa4\x85\x13k\x84\x00\x00\u07d4\
x0f2\xd9\xcbM\x0f\u06a0\x15\x06V\xbb`\x8d\xccC\xed}\x93\x01\x89(\u07cb\xf4@\xdby\x00\x00\u
07d4\x0f6e\u050e\x9f\x14\x19\u0358O\xc7\xfa\x92x\x87\x10\xc8\xf2\xe4\x89lk\x93[\x8b\xbd@\x0

0\x00\u07d4\x0f:\x10#\xca\xc0M\xbfD\xf5\xa5\xfaj\x9c\xf8P\x8c\xd4\xfd\u07c9b\xa9\x92\xe5:\n\xf0
\x00\x00\xe0\x94\x0f@s\xc1\xb9\x9d\xf6\n\x15I\u0597\x89\xc71\x8d\x94\x03\xa8\x14\x8a\x04<3\x
c1\x93ud\x80\x00\x00\xe0\x94\x0fF\xc8\x1d\xb7\x80\xc1gJ\xc7=1O\x06S\x9e\xe5n\xbc\x83\x8a\x
02\x15\xf85\xbcv\x9d\xa8\x00\x00\u07d4\x0fO\x94\xb9\x19\x1b\xb7\xbbUj\xaa\xd7\xc7M\xdb(\x84
\x17\xa5\v\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\x0f`\x00\xde\x15xa\x93
\xab\xa5\xe3\x92pk\x13\x1f\xb1\xdeo\x89\x1b\x1a\xb3\x19\xf5\xecu\x00\x00\u07d4\x0fn\x84\n?*$
d}\x8eC\xe0\x9dE\xc7\xc35\xdfBH\x89\x87\x86x2n\xac\x90\x00\x00\u07d4\x0fu\x15\xff\x0e\x80\x8
fi^\f
H_\xf9n\xd2\xf7\xb7\x93\x10\x8968\"\x16`\xa5\xaa\x80\x00\u07d4\x0fx\x9e09|S\xbf%o\xc3d\xe6\x
ef9\xf8SPA\x14\x89\xc5S%\xcat\x15\xe0\x00\x00\u07d4\x0f{a\u015b\x01c\"\xe8\"l\xaf\xae\xe9\xd9
\xe7mP\xa1\xb3\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x0f{\xeaN\xf3\xf7:\xe0#=\xf1\xe1\x00q
\x8c\xbe)1\v\xb0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x0f{\xf67?w\x1aF\x01v,M\xae_\xbb\xf4\xfe\u
075c\u0249lk\x93[\x8b\xbd@\x00\x00\u07d4\x0f\x83*\x93\u07dd\u007ft\xcd\x0f\xb8Tkq\x98\xbfSw
\xd9%\x89\a\xc0\x86\x0eZ\x80\xdc\x00\x00\u07d4\x0f\x83F\x1b\xa2$\xbb\x1e\x8f\u075d\xaeSQr\
xb75\xac\xb4\xe0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x0f\x85\xe4+\x1d\xf3!\xa4\xb3\xe85\xb5\f\x00\xb0as\x96\x846\x895e\x9e\xf9?\x0f\
xc4\x00\x00\xe0\x94\x0f\x88\xaa\xc94l\xb0\xe74\u007f\xbap\x90Tu\xba\x8b>^\u038a\x02\x1e\x19\
xe0\u027a\xb2@\x00\x00\u07d4\x0f\x92\x9c\xf8\x95\xdb\x01z\xf7\x9f>\xad\"\x16\xb1\xbdi\xc3}\u0
1c9lk\x93[\x8b\xbd@\x00\x00\u07d4\x0f\xa0\x10\xce\fs\x1d;b\x8e6\xb9\x1fW\x13\x00\u477e\xab\x
8963\x03\"\xd5#\x8c\x00\x00\u07d4\x0f\xa5\xd8\u0173\xf2\x94\xef\u0515\xabi\xd7h\xf8\x18rP\x85
H\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x0f\xa6\u01f0\x97=\v\xae)@T\x0e$}6'\xe3|\xa3G\x8965\u0
26d\xc5\u07a0\x00\x00\u07d4\x0f\xad\x05P|\u070f$\xb2\xbeL\xb7\xfa]\x92}\u06d1\x1b\x88\x89\xa
2\xdf\x13\xf4A\xf0\t\x80\x00\u07d4\x0f\xb5\xd2\xc6s\xbf\xb1\xdd\xca\x14\x1b\x98\x94\xfdm?\x05\x
dag \x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x0f\u0260\xe3AE\xfb\xfd\xd2\xc9\u04a4\x99\xb6\x17\u05e0)i\xb9\x89\t\xc2\x
00vQ\xb2P\x00\x00\xe0\x94\x0f\xcf\xc4\x06P\b\xcf\xd3#0_b\x86\xb5zM\xd7\xee\xe2;\x8a\x04<3\x
c1\x93ud\x80\x00\x00\xe0\x94\x0f\xdde@#\x95\u07db\u045f\xeeE\a\xefSE\xf7E\x10L\x8a\x01\x0f
\f\xf0d\xddY \x00\x00\u07d4\x0f\xecN\xe0\xd7\xca\x18\x02\x90\xb6\xbd
\xf9\x99#B\xf6\x0f\xf6\x8d\x89\x12
\u007f\x0e\xdc\xe9q\x80\x00\u07d4\x0f\ue06c3\x1e\xfd\x8f\x81\x16\x1cW8+\xb4P{\xb9\xeb\xec\x
89\x15\xaf\x88\r\x8c\u06c3\x00\x00\u07d4\x0f\xfe\xa0mq\x13\xfbj\xec(i\xf4\xa9\u07f0\x90\a\xfa\xc
e\xf4\x89\f8F\x81\xb1\xe1t\x00\x00\u07d4\x10\tq\x98\xb4\xe7\xee\x91\xff\x82\xcc/;\xd9_\xeds\xc5
@\xc0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x10\vM\tw\xfc\xba\xd4\u07bd^d\xa0Iz\xea\xe5\x16\x8f
\xab\x89\x11\f\x90s\xb5$Z\x00\x00\xe0\x94\x10\x1a\nd\xf9\xaf\xccD\x8a\x8a\x13\rM\xfc\xbe\xe8\x
957\xd8T\x8a\x037\xfe_\xea\xf2\u0440\x00\x00\u07d4\x10,G}i\xaa\u06e9\xa0\xb0\xf6+tY\xe1\u00
7f\xbb\x1c\x15a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x101\xe0\xec\xb5I\x85\xae!\xaf\x17\x93\x95\
r\xc8\x11\x88\x8f\xde|\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x104d\x14\xbe\xc6\xd3\xdc\xc4
NP\xe5MT\u00b8\xc3sN>\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x108\x98X\xb8\x00\xe8\xc0\
xec2\xf5\x1e\xd6\x1a5YF\xcc@\x9b\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x10Y\xcb\xc6>6\xc4>\x88\xf3\x00\b\xac\xa7\xce\x05\x8e\ua816\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94\x10n\xd5\xc7\x19\xb5&\x14w\x89\x04%\xaeuQ\xdcY\xbd%
\\\x8a\x02\x89jX\xc9[\xe5\x88\x00\x00\u07d4\x10q\x1c=\xda21x\x85\xf0\xa2\xfd\x8a\xe9.\x82\x06\

x9b\r\v\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x10sy\xd4\xc4gFO#[\xc1\x8eU\x93\x8a\xad>h\
x8a\u05c9\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94\x10v!-
Ou\x8c\x8e\xc7\x12\x1c\x1c}t%I&E\x92\x84\x8a\ai[Y\xb5\xc1{L\x00\x00\u07d4\x10x\xd7\xf6\x1b\x
0eV\xc7N\xe6c[.\x18\x19\xef\x1e=\x87\x85\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x10z\x03\xcf\b
B\xdb\u07b0a\x8f\xb5\x87\xcai\x18\x9e\xc9/\xf5\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x10\x80\xc
1\xd85\x8a\x15\xbc\x84\xda\xc8%<h\x831\x90
\xdf,\x89\x90\xf54`\x8ar\x88\x00\x00\xe0\x94\x10\x8a+|3oxGy\u0635M\x02\xa8\xd3\x1d\x9a\x13\x
9c\n\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x10\x8b\xa7\u0089\\P\xe0r\xdco\x96I2\xd5
\f(-
04\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x10\x8f\xe8\xee*\x13\xdaH{\"\u01abmX.\xa7\x10
d\u064c\x89\x15\xacV\xed\xc4\xd1,\x00\x00\xe0\x94\x10\x91\x17k\u16d9d\xa8\xf7.\x0e\xcek\xf8\x
e3\u03edn\x9c\x8a\x02\x1f/o\x0f\xc3\xc6\x10\x00\x00\u07d4\x10\x98\xc7t\xc2\f\xa1\u06ac]\xdbb\x
03e1m5?\x10\x9c\x89\x05k\xc7^-c\x10\x00\x00\u0794\x10\x98\xcc
\uf13a\xd5\x14f9\xc4\xcd\x1c\xa6\u00d9l\xb9\x9b\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x10\xa
1\xc4-
\xc1\xbati\x86\xb9\x85\xa5\"\xa7<\x93\xea\xe6Lc\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x10\xa9
4WIo\x11\b\u0358\xe1@\xa1\xec\u06ee^m\xe1q\x89\x15\xa9\x90b\xd4\x16\x18\x00\x00\u07d4\x1
0\xb5\xb3M\x12H\xfc\xf0\x17\xf8\xc8\xff\xc4\b\u0389\x9c\xee\xf9/\x89\x0e~\xeb\xa3A\vt\x00\x00\x
e0\x94\x10\xcfV\td\xff\x83\xc1\xc9gLx<\x0fs\xfc\u0619C\xfc\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u0
7d4\x10\xd3$\x16r,\xa4\xe6Hc\x05H\xea\xd9\x1e\xddy\xc0j\xff\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x10\xd9E3N\xcd\xe4{\ub723\x81l\x17=\xfb\xbd\vS3\x89K\xe4\xe7&{j\xe0\x00
\x00\u07d4\x10\xdfh\x15\x06\xe3I0\xacz\\g\xa5L>\x89\u0392\xb9\x81\x89t\xc1\xfa\xb8\xad\xb4T\x
00\x00\u07d4\x10\xe1\xe37x\x85\xc4-
}\xf2\x18R.\xe7vh\x87\xc0^j\x89\x10C\xc4<\xde\x1d9\x80\x00\u07d4\x10\u342d+\xa3=\x82\xb3s\x
88\u041cED\u01b0\"]\xe5\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\x10\xf4\xbf\xf0\u02a5\x02|\nj-
\xcf\xc9R\x82M\xe2\x94\t\t\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x11\x00\x1b\x89\xed\x87>:\xae\xc
1\x15V4\xb4h\x16C\x98c#\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x11\x027\u03d1\x17\xe7g\x92/
\u0121\xb7\x8dyd\u0682\xdf
\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x11\x11\xe5\xdb\xf4^o\x90mb\x86o\x17\b\x10\x17\x88\x
dd\xd5q\x89F{\xe6S>\xc2\xe4\x00\x00\xe0\x94\x11\x17+'\x8d\xddD\xee\xa2\xfd\xf4\xcb\x1d\x16\x
96#\x91\xc4S\u064a\xc6/=\x9b\xfdH\x95\xf0\x00\x00\u07d4\x11&4\xb4\xec0\xffxn\x02AY\xf7\x96\
xa5y9\xea\x14N\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\x110l}WX\x867x\x0f\xc9\xfd\xe8\xe9\x8e\xcb
\x00\x8f\x01d\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94\x116\x12\xbc;\xa0\xeeH\x98\xb4\x9d\xd2\x02
3\x90_/E\x8fb\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4\x11A_\xaba\xe0\xdf\u0539\x06v\x14\
x1aUz\x86\x9b\xa0\xbd\xe9\x89o\x05\xb5\x9d;
\x00\x00\x00\u07d4\x11L\xbb\xbfo\xb5*\xc4\x14\xbe~\xc6\x1f{\xb7\x14\x95\xce\x1d\xfa\x89\xa2\x
a1]\tQ\x9b\xe0\x00\x00\u07d4\x11L\xfe\xfeP\x17\r\xd9z\xe0\x8f\nDTIx\u0159T\x8d\x89.\u0207\xe
7\xa1J\x1c\x00\x00\u07d4\x11a\b\xc1
\x84a.\xed\xa7\xa9=\xdc\xf8\xd2`.'\x9e\\\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x11d\u02aa\x8c\u01
57z\xfe\x1f\xad\x8a}`(\xce-
W)\x9b\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x11gZ%UF\a\xa3\xb6\xc9*\x9e\xe8\xf3ou\xed\
xd3\xe36\x89\b\xa9\xab\xa5W\xe3l\x00\x00\u07d4\x11j\t\xdff\xcb\x15\x0e\x97W\x8e)\u007f\xb0n\x

13\x04\f\x89<\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x11o\xef^`\x16B\xc9\x18\u02c9\x16\x0f\xc2);\x
a7\x1d\xa96\x89+|\xc2\xe9\xc3\"\\\x00\x00\u07d4\x11xP\x1f\xf9J\xdd\x1cX\x81\xfe\x88a6\xf6\xdf\x
db\xe6\x1a\x94\x89\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u07d4\x11y\xc6\r\xbd\x06\x8b\x15\v\aM\xa4\
xbe#\x03;
\u0185X\x89$\xdc\xe5M4\xa1\xa0\x00\x00\u07d4\x11}\x9a\xa3\xc4\xd1;\xee\x12\xc7P\x0f\t\xf5\xd
d\x1cf\xc4e\x04\x89\v*\xd3\x04\x90\xb2x\x00\x00\xe0\x94\x11}\xb867\u007f\xe1TU\xe0,.\xbd\xa4\
v\x1c\xebU\x1b\x19\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\x11\x8c\x18\xb2\xdc\xe1p\xe8\
xf4Eu;\xa5\xd7Q<\xb7cm-
\x8a\x01\xdd\f\x88_\x9a\r\x80\x00\x00\u07d4\x11\x8f\xbdu;\x97\x929Z\xefzMx\xd2c\xcd\u02ab\xd
4\xf7\x8963\x03\"\xd5#\x8c\x00\x00\xe0\x94\x11\x92\x83x\xd2}U\xc5 \xce\xed\xf2L\xeb\x1e\x82-
\x89\r\xf0\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x11\x9a\xa6M[}\x18\x1d\xae\x9d<\xb4I\x95\\
\x89\xc1\xf9c\xfa\x89%\xf2s\x93=\xb5p\x00\x00\xe0\x94\x11\xc05\x8a\xa6G\x9d\xe2\x18f\xfe!\a\x
19$\xb6^p\xf8\xb9\x8a\a\xb5?y\xe8\x88\xda\xc0\x00\x00\xe0\x94\x11\xd2$z\"\x1ep\xc2\xd6m\x17\
xee\x13\x8d8\xc5_\xfb\x86@\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x11\u05c4JG\x1
e\xf8\x9a\x8d\x87uUX<\xee\xbd\x149\xea&\x8a\x02#i\u6e80\u0188\x00\x00\u07d4\x11\xdda\x85\
u0668\xd7=\xdf\u06a7\x1e\x9bwtC\x1cM\xfe\u008965\u026d\xc5\u07a0\x00\x00\u07d4\x11\xe7\x
99~\u0750E\x03\xd7}\xa6\x03\x8a\xb0\xa4\xc84\xbb\xd5c\x89\x15\b\x94\xe8I\xb3\x90\x00\x00\u0
7d4\x11\xec\x00\xf8I\xb61\x9c\xf5\x1a\xa8\u074ff\xb3U)\xc0\xbew\x89lk\x93[\x8b\xbd@\x00\x00\u
07d4\x11\ufe22\x04Q\x16\x1bdJ\x8c\u03bb\xc1\xd3C\xa3\xbb\xcbR\x89\xadx\xeb\u016cb\x00\x00
\x00\xe0\x94\x11\xfe\xfb]\xc1\xa4Y\x8a\xa7\x12d\fQwu\u07e1\xd9\x1f\x8c\x8a\x02\x1e\x19\xe0\u0
27a\xb2@\x00\x00\u07d4\x12\x0f\x9d\xe6\xe0\xaf~\xc0*\a\xc6\t\u0284G\xf1W\xe64L\x89\x0e~\xe
b\xa3A\vt\x00\x00\u07d4\x12\x10\xf8\v\u06c2l\x17Tb\xab\a\x16\xe6\x9eF\xc2J\xd0v\x89\x05k\xc7
^-
c\x10\x00\x00\u07d4\x12\x13N\u007fk\x01{\xf4\x8e\x85Z9\x9c\xa5\x8e.\x89/\xa5\u020965\u026d\
xc5\u07a0\x00\x00\u07d4\x12\x170t\x98\x01S\xae\xaaK\r\xcb\xc7\x13.\xad\xce\xc2\x1bd\x89\r\x0
2\xabHl\xed\xc0\x00\x00\u07d4\x12\x1f\x85[p\x14\x9a\xc84s\xb9po\xb4MG\x82\x8b\x98;\x89K\xe
4\xe7&{j\xe0\x00\x00\u07d4\x12'\xe1\nM\xbf\x9c\xac\xa3\x1b\x17\x80#\x9fUv\x15\xfc5\xc1\x89\n\
u05ce\xbcZ\xc6 \x00\x00\u07d4\x12-\xcf\xd8\x1a\u0779}\x1a\x0eI%\u0135I\x80n\x9f;\xeb\x89R
5\xccn\x01!\x00\x00\u07d4\x12/V\x12%I\xd1h\xa5\xc5\xe2g\xf5&b\xe5\xc5\xcc\xe5\u0209\n\ad\a\x
d3\xf7D\x00\x00\xe0\x94\x121o\xc7\xf1x\xea\xc2.\xb2\xb2Z\xed\xea\xdf=u\xd0\x01w\x8a\x04<3\x
be\x05\xf6\xbf\xb9\x80\x00\xe0\x94\x127Y\xf33\xe1>0i\xe2\x03KO\x059\x89\x18\x11\x9d6\x8a\x0
4<3\xc1\x93ud\x80\x00\x00\u07d4\x12\\\xc5\xe4\xd5k+\xcc.\xe1\xc7\t\xfb\x9eh\xfb\x17t@\xbd\x89
lk\x93[\x8b\xbd@\x00\x00\u07d4\x12c#\x88\xb2v^\xe4E+P\x16\x1d\x1f\xff\xd9\x1a\xb8\x1fJ\x89(\x
1d\x90\x1fO\xdd\x10\x00\x00\u07d4\x12h\x97\xa3\x11\xa1J\xd4;x\xe0\x92\x01\x00\xc4Bk\xfdk\u0
7494\xc7&\x89?-
\x94\x80\x00\u07d4\x12m\x91\xf7\xad\x86\u07bb\x05W\xc6\x12\xca'n\xb7\xf9m\x00\xa1\x89\x05k
\xc7^-
c\x10\x00\x00\u07d4\x12}?\xc5\x00;\xf6<\r\x83\xe99W\x83e\x15\xfd'\x90E\x89\x06\x10\xc9\".nu\x
00\x00\xe0\x94\x12}\xb1\xca\xdf\x1bw\x1c\xbdtu\xe1\xb2ri\x0fU\x8c\x85e\x8a\x02\xf6\xf1\a\x80\xd
2,\xc0\x00\x00\u07d4\x12\x84\xf0\xce\xe9\xd2\xff)\x89\xb6Ut\xd0o\xfd\x9a\xb0\xf7\xb8\x05\x89\x1
5\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x12\x8b\x90\x8f\xe7C\xa44
=\xe2\x94\xc4A\xc7\xe2\n\x86\xeag\x89&\xab\x14\xe0\xc0\xe1<\x00\x00\xe0\x94\x12\x93\u01cc}j

D;\x9dt\xb0\xba^\xe7\xbbG\xfdA\x85\x88\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\x12\x96\xac\xde
\xd1\xe0c\xaf9\xfe\x8b\xa0\xb4\xb6=\xf7\x89\xf7\x05\x17\x89\x05k\xf9\x1b\x1ae\xeb\x00\x00\u07d
4\x12\xaa}\x86\xdd\xfb\xad0\x16\x92\xfe\xac\x8a\b\xf8A\xcb!\\7\x89\amA\xc6$\x94\x84\x00\x00\xe
0\x94\x12\xaf\xbc\xba\x14'\xa6\xa3\x9e{\xa4\x84\x9fz\xb1\xc45\x8a\xc3\x1b\x8a\x04<3\xc1\x93ud\
x80\x00\x00\xe0\x94\x12\xb5\xe2\x89E\xbb)i\xf9\xc6Lc\xcc\x05\xb6\xf1\xf8\xd6\xf4\u054a\x01\xa2
\x9e\x86\x91;t\x05\x00\x00\u0794\x12\u03cb\x0eFR\x13!\x1a[S\u07f0\xdd'\x1a(,\x12\u0248\xd2\xf
1?w\x89\xf0\x00\x00\u07d4\x12\xd2\a\x90\xb7\xd3\xdb\u060c\x81\xa2y\xb8\x12\x03\x9e\x8a`;\u0
409V\xf9\x85\u04c6D\xb8\x00\x00\xe0\x94\x12\xd6\re\xb7\xd9\xfcH\x84\v\xe5\xf8\x91\xc7E\xcev\
xeeP\x1e\x8a\x04\x85\xe58\x8d\fv\x84\x00\x00\u0794\x12\xd9\x1a\x92\xd7O\xc8a\xa7)dm\xb1\x9
2\xa1%\xb7\x9fSt\x88\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94\x12\u992d*\xd5t\x84\xddp\x05e\xbd\
xdbFB;\u067d1\x8a\x04<0\xfb\b\x84\xa9l\x00\x00\u07d4\x12\xf3,\n\x1f-
\xaa\xb6v\xfei\xab\xd9\xe0\x185-
L\xcdE\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\x12\xf4`\xaedl\xd2x\x0f\xd3\\P\xa6\xafK\x
9a\xcc\xfa\x85\u018965\u026d\xc5\u07a0\x00\x00\u07d4\x12\xff\xc1\x12\x86\x05\xcb\f\x13p\x9ar\
x90Po&\x90\x97q\x93\x89\xb5\x0f\u03ef\xeb\xec\xb0\x00\x00\u07d4\x13\x03$F\xe7\xd6\x10\xaa\
x00\xec\x8cV\u0275t\xd3l\xa1\xc0\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x13\x1cy,\x19}\x18\xb
d\x04]p$\x93|\x1f\x84\xb6\x0fD8\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x13\x1d\xf8\xd30\xeb
|\xc7\x14}\nUWo\x05\u078d&\xa8\xb7\x89\n1\x06+\xee\xedp\x00\x00\u07d4\x13\x1f\xae\xd1%a\x
bbz\xee\x04\xe5\x18Z\xf8\x02\xb1\xc3C\x8d\x9b\x89\v\xdf<K\xb02\x8c\x00\x00\u07d4\x13!\xb6\x0
5\x02oO\xfb)j>\x0e\u0733\x90\xc9\xc8V\b\xb7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x13!\xcc\xf2\x
979\xb9t\xe5\xa5\x16\xf1\x8f:\x846q\xe3\x96B\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x13'\xd
7Y\xd5n\n\xb8z\xf3~\xcfc\xfe\x01\xf3\x10\xbe\x10\n\x89#\xbc<\xdbh\xa1\x80\x00\x00\u07d4\x13)\
xdd\x19\xcdK\xaa\x9f\xc6C\x10\xef\xec\xea\xb2!\x17%\x1f\x12\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x137\x1f\x92\xa5n\xa88\x1eC\x05\x9a\x95\x12\x8b\xdcMC\u0166\x8965\u026d\xc
5\u07a0\x00\x00\u07d4\x13<I\x0f\xa5\xbf\u007f7(\x88\xe6\a\xd9X\xfa\xb7\xf9U\xba\xe1\x89U\xa6\
xe7\x9c\xcd\x1d0\x00\x00\u07d4\x13>O\x15\xe1\xe3\x9cSCY0\xaa\xed\xf3\xe0\xfeV\xfd\xe8C\x8
9\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x13Ac\xbe\x9f\xbb\xe1\xc5in\xe2U\xe9\v\x13%C\x95\xc3
\x18\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x13\\\xec\xd9U\xe5y\x83pv\x920\x15\x93\x03\u0671\x83\x9ff\x8a\x01\x0f\f\xf0d\
xddY
\x00\x00\u07d4\x13]\x17\x19\xbf\x03\xe3\xf8f1$y\xfe3\x81\x18\xcd8~p\x89lk\x93[\x8b\xbd@\x00\x
00\xe0\x94\x13^\xb8\xc0\xe9\xe1\x01\xde\xed\xec\x11\xf2\xec\xdbf\xae\x1a\xae\x88g\x8a\x04<3\x
c1\x93ud\x80\x00\x00\u07d4\x13`\xe8}\xf2Li\xeemQ\xc7nsv\u007f\xfe\x19\xa2\x13\x1c\x89\x04\xf
c\xc1\xa8\x90'\xf0\x00\x00\u07d4\x13l\x83K\xf1\x112m s\x95)[.X>\xa7\xf35r\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x13mKf+\xbd\x10\x80\xcf\xe4D[\x0f\xa2\x13\x86D5\xb7\xf1\x89\xd8\xd7&\xb
7\x17z\x80\x00\x00\u07d4\x13oI\a\u02b4\x1e'\bK\x98E\x06\x9f\xf2\xfd\f\x9a\xdey\x89\xd8\xd7&\x
b7\x17z\x80\x00\x00\u07d4\x13t\xfa\xcd{?\x8dhd\x9d`\xd4U\x0e\xe6\x9f\xf0HA3\x89\x0e\x9e\xd6\
xe1\x11r\xda\x00\x00\u07d4\x13|\xf3A\xe8Ql\x81X\x14\xeb\xcds\xe6V\x9a\xf1L\xf7\xbc\x8965\u02
6d\xc5\u07a0\x00\x00\xe0\x94\x13\x84\x8bF\xeau\xbe\xb7\xea\xa8_Y\xd8f\xd7\u007f\xd2L\xf2\x1
a\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\x13\x9d51\u0252*\xd5bi\xf60\x9a\xa7\x89\xfb$\x85\xf
9\x8c\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x13\x9eG\x97d\xb4\x99\xd6f
\x8cJ\x8a\x04z\x97\x041c\u0749 w!*\xffm\xf0\x00\x00\u07d4\x13\xa5\xee\xcb80]\xf9Iq\xef-

\x9e\x17\x9a\xe6\u03ba\xb37\x89\x11\u3ac3\x95\xc6\xe8\x00\x00\u07d4\x13\xac\xad\xa8\x98\n\xf
f\xc7PI!\xbe\x84\xebID\xc8\xfb\xb2\xbd\x89V\u04aa:\\\t\xa0\x00\x00\u07d4\x13\xb9\xb1\a\x15qL\t\
xcf\xd6\x10\u03dc\x98F\x05\x1c\xb1\xd5\x13\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x13\xce3-
\xffe\xa6\xab\x938\x97X\x8a\xa2>\x00\t\x80\xfa\x82\x89\x0e\x02\x056\xf0(\xf0\x00\x00\u07d4\x13\
xd6z~%\xf2\xb1,\u06c5XP\t\xf8\xac\u011b\x96s\x01\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\x13\xde\
xe0>7\x99\x95-
\a8\x84=K\xe8\xfc\n\x80?\xb2\x0e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x13\xe0/\xb4H\xd6\xc8J\x
e1}\xb3\x10\xad(m\x05a`\u0695\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x13\xe3!r\x8c\x9cWb\x80X\x
e9?\xc8f\xa02\xdd\v\u0690\x89&\xbc\xca#\xfe.\xa2\x00\x00\u07d4\x13\xec\x81\"\x84\x02n@\x9b\
xc0f\xdf\xeb\xf9\u0564\xa2\xbf\x80\x1e\x89WG=\x05\u06ba\xe8\x00\x00\xe0\x94\x14\x01)\xea\xa
7f\xb5\xa2\x9f[:\xf2WND\t\xf8\xf6\xd3\xf1\x8a\x01Z\xf1\u05cbX\xc4\x00\x00\x00\u07d4\x14\x05\x1
8\xa3\x19K\xad\x13P\xb8\x94\x9ee\x05e\u07bem\xb3\x15\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1
4\x06\x85M\x14\x9e\b\x1a\xc0\x9c\xb4\xcaV\r\xa4c\xf3\x120Y\x89Hz\x9a0E9D\x00\x00\u07d4\x1
4\f\xa2\x8f\xf3;\x9ff\xd7\xf1\xfc\x00x\xf8\xc1\xee\xf6\x9a\x1b\xc0\x89V\xbcu\xe2\xd61\x00\x00\x00
\u07d4\x14\x0f\xbaX\xdb\xc0H\x03\xd8L!0\xf0\x19x\xf9\xe0\xc71)\x89\x15\xaf\x1dx\xb5\x8c@\x00
\x00\xe0\x94\x14\x1a^9\xee/h\n`\x0f\xbfo\xa2\x97\u0790\xf3\"\\\u074a\x02\x1e\x19\xe0\u027a\xb2
@\x00\x00\u07d4\x14%N\xa1&\xb5-
\x01B\xda\n~\x18\x8c\xe2U\xd8\xc4qx\x89*\x03I\x19\u07ff\xbc\x00\x00\u07d4\x14+\x87\xc5\x04?\
xfbZ\x91\xdf\x18\xc2\xe1\t\xce\xd6\xfeJq\u06c9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x14<c\x97R\xca\xee\xcfj\x99}9p\x9f\xc8\xf1\x98x\xc7\xe8\x89j\xcb=\xf2~\x1f\x88\x
00\x00\u07d4\x14=Sk\x8b\x1c\xb8OV\xa3\x9e\v\xc8\x1f\xd5D+\xca\xcc\xe1\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x14?_\x16X\xd9\xe5x\xf4\xf3\xd9_\x80\xc0\xb1\xbd93\xcb\u0689P\xc5\xe7a\
xa4D\b\x00\x00\u07d4\x14A\x0f\xb3\x10q\x1b\xe0t\xa8\b\x83\xc65\xd0\xefj\xfb%9\x89lk\x93[\x8b\
xbd@\x00\x00\u07d4\x14K\x19\xf1\xf6l\xbe1\x83G\u4344\xb1@9FlY\t\x89lk\x93[\x8b\xbd@\x00\x
00\u07d4\x14RP\xb0nO\xa7\xcb'IB.\xb8\x17\xbd\u068bT\xde_\x89\v\xdf<K\xb02\x8c\x00\x00\u07
d4\x14^\x06\x00\xe2\xa9'\xb2\u074d7\x93V\xb4Z.}Q\u04ee\x89\x8a\x02\xab@\v\xb2\u02c0\x00\u
07d4\x14^\x1d\xe0\x14y\x11\xcc\u0600\x87_\xbb\xeaa\xf6\xa1B\xd1\x1d\x89\xd8\xd7&\xb7\x17z\
x80\x00\x00\xe0\x94\x14c\xa8sU[\xc09~W\\$q\xcfw\xfa\x9d\xb1F\xe0\x8a\x02\x1e\x19\xe0\u027a\
xb2@\x00\x00\u07d4\x14y\xa9\xect\x80\xb7K]\xb8\xfcI\x9b\xe3R\xda\u007f\x84\ue70965\u026d\x
c5\u07a0\x00\x00\u07d4\x14z\xf4j\xe9\xcc\u044b\xb3\\\xa0\x1b5;Q\x99\x0eI\xdc\xe1\x89\xd8\xd7
&\xb7\x17z\x80\x00\x00\u07d4\x14\u007fB\x10\xabX\x04\x94\n\v}\xb8\xc1L(9kb\xa6\xbf\x89lk\x93
[\x8b\xbd@\x00\x00\u07d4\x14\x83\a\x04\u96aa\xd5\xc5^\x1fP+'\xb2,\x12\xc9\x193\x89!\x9c:{\x19
f0\x00\x00\xe0\x94\x14\x9bm\xbd\xe62\xc1\x9fZ\xf4|\xb4\x93\x11K\xeb\u0670<\x1f\x8a\x02\x8a\x
85t%Fo\x80\x00\x00\xe0\x94\x14\x9b\xa1\x0f\r\xa2r]\xc7\x04s>\x87\xf5\xa5$\u0288Q^\x8a\x01\xa
b,\xf7\xc9\xf8~ \x00\x00\u07d4\x14\xa75
f6D\x04\xdbP\xf0\xd0\u05cduJ\"\x19\x8e\xf4\x89e\xea=\xb7UF`\x00\x00\u07d4\x14\xab\x16K;RL\x
82\u05ab\xfb\xc0\u0783\x11&\xae\x8d\x13u\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x14\xb1`>\xc6+
\x02
3\xee\xc4\xd6\xd6eZ\xc2J\x01Z\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94\x14\xc6;\xa2\u
0731\xddM\xf3=\u06b1\x1cO\x00\a\xfa\x96\xa6-
\x8a\x03HA\xb6\x05z\xfa\xb0\x00\x00\xe0\x94\x14\xcd\u077c\x8b\t\xe6gZ\x9e\x9e\x05\t\x1c\xb9\"
8\u00de\x1e\x8a\x01\x14x\xb7\xc3\n\xbc0\x00\x00\u07d4\x14\xd0\n\xad9\xa0\xa7\u045c\xa0SP\xf

7\xb07'\xf0\x8d\xd8.\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x14\xee\xc0\x9b\xf0>5+\xd6\xff\x1b\x1e\x87k\xe6d\xce\xff\xd0\u03c9\x01\x16\xdc:\x89\x94\xb3\x00\x00\u07d4\x14\xf2!\x15\x95\x18x;\u0127\x06go\xc4\xf3\xc5\xee@X)\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x14\xfc\xd19\x1e}s/Avl\xda\u0344\xfa\x1d\xeb\x9f\xfd\u0489lk\x93[\x8b\xbd@\x00
\x00\u07d4\x15\x0e=\xbc\xbc\xfc\x84\xcc\xf8\x9bsBwc\xa5e\xc2>`\u0409\x02+\x1c\x8c\x12'\xa0\x00\x00\xe0\x94\x15\x18b{\x885\x1f\xed\xe7\x96\xd3\xf3\b3d\xfb\u0508{\f\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u0794\x15\"J\xd1\xc0\xfa\xceF\xf9\xf5V\xe4wJ0%\xad\x06\xbdR\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\x15/+\xd2)\xdd\xf3\xcb\x0f\xda\xf4U\xc1\x83
\x9c\x0e\x1e9\xa2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x15/N\x86\x0e\xf3\xee\x80jP'w\xa1\xb8\xdb\xc9\x1a\x90vh\x89
\x86\xac5\x10R`\x00\x00\u07d4\x15<\b\xaa\x8b\x96\xa6\x11\xefc\xc0%>*C4\x82\x9eW\x9d\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\x15<\xf2\x84,\xb9\u0787l'o\xa6Gg\u0468\xec\xf5s\xbb\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x15>\xf5\x8a\x1e.z>\xb6\xb4Y\xa8\n\xb2\xa5G\xc9A\x82\xa2\x8a\x14T+\xa1*3|\x00\x00\x00\u07d4\x15DY\xfa/!1\x8e44D\x97\x89\xd8&\xcd\xc1W\f\xe5\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x15G\xb9\xbfz\xd6bt\xf3A8'#\x1b\xa4\x05\ue308\xc1\x8a\x03\xa9\u057a\xa4\xab\xf1\xd0\x00\x00\u07d4\x15H\xb7p\xa5\x11\x8e\u0787\u06e2\xf6\x903\u007fam\u60eb\x89\x1c\x99V\x85\u0fc7\x00\x00\u07d4\x15R\x83P\xe0\xd9g\n.\xa2\u007f{J3\xb9\xc0\xf9b\x1d!\x89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4\x15[7y\xbbmV4./\u0681{[-
\x81\xc7\xf4\x13'\x89\x02\xb8\xaa:\al\x9c\x00\x00\u07d4\x15e\xaf\x83~\xf3\xb0\xbdN+#V\x8dP#\xcd4\xb1d\x98\x89\x15Q\xe9rJ\u013a\x00\x00\u07d4\x15f\x91\x80\xde\u2558\x86\x9b\b\xa7!\xc7\xd2LL\x0e\xe6?\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x15r\xcd\xfa\xb7*\x01\u0396\x8ex\xf5\xb5D\x8d\xa2\x98S\xfb\u074a\x01\x12bII\x06\x0f\xa6\x00\x00\xe0\x94\x15uY\xad\xc5Wd\xccm\xf7\x93#\t%4\xe3\xd6dZf\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x15x\xbd\xbc7\x1bM$8E3\x05V\xff\xf2\xd5\xefM\xffg\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x15~\xb3\xd3\x11;\u04f5\x97qM:\x95N\xdd\x01\x89\x82\xa5\u02c9lk\x93[\x8b\xbd@\x00\x00\u07d4\x15\x84\xa2\xc0f\xb7\xa4U\xdb\u05ae(\a\xa73N\x83\xc3_\xa5\x89\a\f\x1c\xc7;\x00\xc8\x00\x00\u07d4\x15\x87F\x86\xb6s=\x10\xd7\x03\xc9\xf9\xbe\xc6\xc5.\xb8b\x8dg\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x15\x8a\ra\x92S\xbfD2\xb5\xcd\x02\u01f8b\xf7\u00b7V6\x89\a[\xac|[\x12\x18\x80\x00\u07d4\x15\x98\x12y\x82\xf2\xf8\xad;k\x8f\xc3\xcf'\xbfax\x01\xba+\x89\t`\xdbwh\x1e\x94\x00\x00\xe0\x94\x15\x9a\xdc\xe2z\xa1\vG#d)\xa3JZ\xc4,\xad[d\x16\x8a\x06\xbf\x90\xa9n\xdb\xfaq\x80\x00\u07d4\x15\xa0\xae\xc3\u007f\xf9\xff=T\t\xf2\xa4\xf0\xc1!*\xac\xcb\x02\x96\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x15\xaaS\r\xc3iX\xb4\xed\xb3\x8e\xeem\xd9\xe3\xc7}L\x91E\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x15\xac\xb6\x15h\xecJ\xf7\xea(\x198a\x81\xb1\x16\xa6\xc5\xeep\x8a\x06\x90\x83l\n\xf5\xf5`\x00\x00\u07d4\x15\xb9o0\xc2;\x86d\xe7I\x06Q\x06k\x00\xc49\x1f\xbf\x84\x89\x16B\xe9\xdfHv)\x00\x00\u07d4\x15\xc7\xed\xb8\x11\x8e\xe2{4\"\x85\xebY&\xb4z\x85[\u01e5\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x15\u0654hPz\xa0A?\xb6\r\xca*\xdc\u007fV\x9c\xb3kT\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x15\u06f4\x8c\x980\x97d\xf9\x9c\xed6\x92\xdc\xca5\xee0k\xac\x8a\x1f\u00c4+\xd1\xf0q\xc0\x00\x00\xe0\x94\x15\u072f\xcc+\xac\xe7\xb5[T\xc0\x1a\x1cQF&\xbfa\xeb\u060a\x01\xfd\x934\x94\xaa_\xe0\x00\x00\u07d4\x15\u3d44\x05kb\xc9s\xcf^\xb0\x96\xf1s>T\xc1\\\x91\x892\xc7Z\x02#\xdd\xf3\x00\x00\u07d4\x15\xeb\xd1\xc7\xca\u04af\xf1\x92u\xc6W\xc4\xd8\b\xd0\x10\xef\xa0\xf5\x89\n\xdf0\xbap\u0217\x00\x00\u07d4\x15\xee\x0f\xc6>\xbf\x1b\x1f\u011d{\xb3\x8f\x88c\x82:.\x17\u0489g\x8a\x93

b\xe4\x18\x00\x00\u07d4\x15\xf1\xb3R\x11\rh\x90\x1d\x8fg\xaa\xc4jl\xfa\xfe\x03\x14w\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x15\xf2\xb7\xb1d2\xeeP\xa5\xf5[A#/c4\xedX\xbd\xc0\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x16\x01\x9aM\xaf\xabC\xf4\u067fAc\xfa\xe0\x84}\x84\x8a\xfc\xa2\x89\x01[\xc7\x019\xf7J\x00\x00\u07d4\x16\x02&\xef\xe7\xb5:\x8a\xf4b\xd1\x17\xa0\x10\x80\x89\xbd\xec\xc2\u0449\n\xdf0\xbap\u0217\x00\x00\u07d4\x16\f\xebo\x98\x0e\x041_S\xc4\xfc\x98\x8b+\xf6\x9e(M}\x89\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\xe0\x94\x16\x1c\xafZ\x97*\u0383y\xa6\u0420J\xe6\xe1c\xfe!\xdf+\x8a\x15-\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\x16\x1d&\xefgY\xba[\x9f
\xfd\xcdf\xf1a2\xc3RA^\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x16!\x10\xf2\x9e\xac_}\x02\xb5C\xd8\xdc\u057bY\xa5\xe3;s\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x16+\xa5\x03'b\x14\xb5\t\xf9u\x86\xbd\x84!\x10\xd1\x03\xd5\x17\x8a\x01\xe7\xff\u0609\\\"h\x00\x00\u07d4\x16-
v\xc2\xe6QJ:\xfbo\xe3\xd3\u02d3\xa3\\Z\xe7\x83\xf1\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x16;\xadJ\x12+E}d\xe8\x15\nA>\xaeM\a\x02>k\x89\x01\x04\xe7\x04d\xb1X\x00\x00\u07d4\x16<\u023e\"vF\xcb\tq\x91Y\xf2\x8e\u041c]\xc0\xdc\xe0\x89Hz\x9a0E9D\x00\x00\u07d4\x16=\xcas\xd7\xd6\xea?>`b2*\x874\x18\f\vx\uf25ft
\x03\xcb}\xfc\x00\x00\u07d4\x16Mz\xac>\xec\xba\uc86dQ\x91\xb7S\xf1s\xfe\x12\xec3\x89(VR\xb8\xa4hi\x00\x00\u07d4\x16Rl\x9e\u07d4>\xfaOm\x0f\v\xae\x81\xe1\x8b1\xc5@y\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\x16S\x05\xb7\x872.%\xdcj\xd0\xce\xfelo3Fx\xd5i\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x16e\xab\x179\xd7\x11\x19\xeea2\xab\xbd\x92j'\x9f\xe6yH\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\x16k\xf6\u06b2-
\x84\x1bHl8\xe7\xbaj\xb3:\x14\x87\ud30a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x16v\x99\xf4\x8ax\xc6\x15Q%\x15s\x99X\x993\x12WO\a\x89\x02\x1d;\xd5^\x80<\x00\x00\u07d4\x16x\xc5\xf2\xa5\"92%\x19ca\x89OS\xccu/\xe2\xf3\x89h\xf3e\xae\xa1\xe4@\x00\x00\u07d4\x16|\xe7\xdee\xe8G\bYZRT\x97\xa3\xeb^ZfPs\x89\x1f1Gsfo\xc4\x00\x00\u07d4\x16~>:\xe2\x003HE\x93\x92\xf7\xdf\xceD\xaf|!\xadY\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\x16\x80\xce\xc5\x02\x1e\xe90P\xf8\xae\x12rQ\x83\x9et\xc1\xf1\xfd\x8a\x02\xc6\x14a\xe5\xd7C\u0580\x00\u07d4\x16\x81j\xac\x0e\xde\r-
<\xd4B\xday\xe0c\x88\x0f\x0f\x1dg\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x16\x8bP\x19\xb8\x18i\x16D\x83_\xe6\x9b\xf2)\xe1q\x12\xd5,\x8a\x05\xed\xe2\x0f\x01\xa4Y\x80\x00\x00\u07d4\x16\x8b\xde\xc8\x18\xea\xfcm)\x92\xe5\xefT\xaa\x0e\x16\x01\xe3\xc5a\x8967Pz0\xab\xeb\x00\x00\u07d4\x16\x8d0\xe5?\xa6\x81\t+R\xe9\xba\xe1Z\r\xcbA\xa8\u027b\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x16\x9b\xbe\xfcA\xcf\xd7\xd7\u02f8\xdf\xc60
\xe9\xfb\x06\u0515F\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x16\xa5\x8e\x98]\xcc\xd7\a\xa5\x94\u0453\xe7\u0327\x8b]\x02xI\x89I\xb9\u029aiC@\x00\x00\u07d4\x16\xa9\xe9\xb7:\u92c6M\x17(y\x8b\x87f\xdb\xc6\xea\x8d\x12\x893\xe7\xb4K\r\xb5\x04\x00\x00\u07d4\x16\xaaR\xcb\vUG#\xe7\x06\x0f!\xf3'\xb0\xa6\x83\x15\xfe\xa3\x89\r\x8drkqw\xa8\x00\x00\u07d4\x16\xab\xb8\xb0!\xa7\x10\xbd\u01ce\xa54\x94\xb2\x06\x14\xffN\xaf\xe8\x89\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u07d4\x16\xaf\xa7\x87\xfc\x9f\x94\xbd\xffiv\xb1\xa4/C\n\x8b\xf6\xfb\x0f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x16\xba\xe5\xd2N\xff\x91w\x8c\u064bM:\x1c\xc3\x16/D\xaaw\x89\x15\xbeat\xe1\x91.\x00\x00\u07d4\x16\xbc@!Z\xbb\u066e](\v\x95\xb8\x01\vE\x14\xff\x12\x92\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x16\xbeu\u9299Z9R\"\xd0\v\u05df\xf4\xb6\xe68\u144a\a\x9f\x90\\o\xd3N\x80\x00\x00\u07d4\x16\xc1\xbf[}\xc9\xc8<\x17\x9e\xfa\xcb\xcf.\xb1t\xe3V\x1c\xb3\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x16\u01f3\x1e\x8c7b\x82\xac\"qr\x8c1\xc9^5\xd9R\u00c9lk\x93[\x8b\xbd@\x00\

x00\u07d4\x16\xf3\x13\u03ca\xd0\x00\x91J\n\x17m\u01a44+y\xec%8\x89lk\x93[\x8b\xbd@\x00\x0
0\u07d4\x16\xff\xac\x84\x03)@\xf0\x12\x1a\tf\x8b\x85\x8a~y\xff\xa3\xbb\x89\xd2J\xdan\x10\x87\x
11\x00\x00\xe0\x94\x17\x03\xb4\xb2\x92\xb8\xa9\xde\xdd\xed\xe8\x1b\xb2]\x89\x17\x9fdF\xb6\x8
a\x04+e\xa4U\xe8\xb1h\x00\x00\u07d4\x17\x04\x93\x11\x10\x1d\x81~\xfb\x1de\x91\x0ff6b\xa6\x9
9\u024c\x89lh\xcc\u041b\x02,\x00\x00\u07d4\x17\x04\xce\xfc\xfb\x131\xeczx8\x8b)9>\x85\xc1\xaf
y\x16\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x17\n\x88\xa8\x99\u007f\x92\xd287\x0f\x1a\xff\x
de\xe64pP\xb0\x13\x89\xa2\xacw5\x14\x880\x00\x00\u07d4\x17\x10\x8d\xab,P\xf9\x9d\xe1\x10\u
1cf3\xb4\u0342\xf5\xdf(\xe7\x895
;g\xbc\xca\xd0\x00\x00\xe0\x94\x17\x12[Y\xacQ\xce\xe0)\xe4\xbdx\xd7\xf5\x94}\x1e\xa4\x9b\xb2\
x8a\x04\xa8\x9fT\xef\x01!\xc0\x00\x00\u07d4\x17\x1a\u0660K\xed\u0238a\xe8\xedK\xdd\xf5qx\x1
3\xb1\xbbH\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x17\x1c\xa0*\x8bmb\xbfL\xa4~\x90i\x14\a\
x98a\x97,\xb2\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x17\"\xc4\xcb\xe7\n\x94\xb6U\x9dBP\x84\xca\xee\xd4\xd6\xe6n!\x89\xd8\xd7&\xb
7\x17z\x80\x00\x00\u07d4\x17X\vvotSR\\\xa4\u01a8\x8b\x01\xb5\x05p\xea\b\x8c\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x17X\x9al\x00jT\xca\xd7\x01\x03\x12:\xae\n\x82\x13_\u07b4\x89\xd8\xd7&\x
b7\x17z\x80\x00\x00\xe0\x94\x17Z\x18::#_\xfb\xb0;\xa85gRg\"\x94\x17\xa0\x91\x8a\x03c\\\x9a\xd
c]\xea\x00\x00\x00\u07d4\x17_\xee\xea*\xa4\xe0\xef\xda\x12\xe1X\x8d/H2\x90\xed\xe8\x1a\x89\n
\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x17e6\x1c.\xc2\xf86\x16\u0383c\xaa\xe2\x10%\xf2Vo@\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\x17gR_Z\"\xed\x80\xe9\xd4\xd7q\x0f\x03b\u049e\xfa3\x89\x15\xaf\x1dx\xb5\x8c
@\x00\x00\u07d4\x17v%`\xe8*\x93\xb3\xf5\"\xe0\xe5$\xad\xb8a,:tp\x8965\u026d\xc5\u07a0\x00\x
00\u07d4\x17}\xaex\xbc\x01\x13\xd8\u04dcD\x02\xf2\xa6A\xae*\x10Z\xb8\x89b\x92BV
\xb4H\x00\x00\xe0\x94\x17\x84\x94\x8b\xf9\x98H\u021eDV8PM\u0598'\x1bY$\x8a\x01GLA\r\x87\
xba\xee\x00\x00\u07d4\x17\x88\u069bW\xfd\x05\xed\xc4\xff\x99\xe7\xfe\xf3\x01Q\x9c\x8a\n\x1e\x
89lk\x93[\x8b\xbd@\x00\x00\u07d4\x17\x8e\xafk\x85T\xc4]\xfd\xe1kx\xce\f\x15\u007f.\xe3\x13Q\x
89\x11X\xe4`\x91=\x00\x00\x00\u07d4\x17\x96\x1dc;\xcf
\xa7\xb0)\xa7\xd9K}\xf4\xda.\xc5B\u007f\x89\fo\xf0p\U000532c0\x00\u07d4\x17\x96\xbc\xc9{\x8a\
xbcq\u007fKJ|k\x106\xea!\x82c\x9f\x89\x13A\xf9\x1c\xd8\xe3Q\x00\x00\u07d4\x17\x99=1*\xa1\x1
0iW\x86\x8fjU\xa5\xe8\xf1/w\xc8C\x89\x18e\xe8\x14\xf4\x14.\x80\x00\u07d4\x17\x9a\x82^\x0f\x1f
n\x98S\tf\x84e\xcf\xfe\xd46\xf6\xae\xa9\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x17\xb2\xd6\x
cfe\xc6\xf4\xa3G\xdd\xc6W&U5M\x8aA+)\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x17\xb8\a\xaf\xa3\
xdd\xd6G\xe7#T.{R\xfe\xe3\x95'\xf3\x06\x89\x15\xaf@\xff\xa7\xfc\x01\x00\x00\u07d4\x17\xc0G\x8
6W\xe1\xd3\xd1z\xaa3\x1d\xd4)\xce\u03d1\xf8\xae]\x8964\xfb\x9f\x14\x89\xa7\x00\x00\u07d4\x17
\xc0\xfe\xf6\x98l\xfb.@A\xf9\x97\x9d\x99@\xb6\x9d\xff=\xe2\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\
u07d4\x17\u0511\x8d\xfa\xc1]w\xc4\u007f\x9e\xd4\x00\xa8P\x19\rd\xf1Q\x89lk\x93[\x8b\xbd@\x0
0\x00\u07d4\x17\xd5!\xa8\xd9w\x90#\xf7\x16M#<;d
\xff\xd2#\xed\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x17\xd91\xd4\xc5b\x94\u073ew\xc8e[\xe
4i_\x00mJ<\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x17\xdfIQ\x8ds\xb1)\xf0\xda6\xb1\u0274\f\xb6d
\xfd\u01ca\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x17\xe4\xa0\xe5+\xac>\xe4N\xfe\tT\xe7S
\u0538]dN\x05\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x17\xe5\x84\xe8\x10\xe5gp,a\xd5]CK4\u037
5\xee0\xf6\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\x17\xe8.px\xdcO\xd9\xe8y\xfb\x8aPf\u007fS\xa5\xc5E\x91\x89\n\u05ce\xbcZ\xc6

\x00\x00\u07d4\x17\xe8o;[0\xc0\xbaY\xf2\xb2\xe8XB[\xa8\x9f\n\x10\xb0\x89lk\x93[\x8b\xbd@\x00\
x00\xe0\x94\x17\xee\x9fT\xd4\xdd\xc8Mg\x0e\xff\x11\xe5Je\x9f\xd7/DU\x8a\x03c\\\x9a\xdc]\xea\x
00\x00\x00\xe0\x94\x17\xefJ\xcc\x1b\xf1G\xe3&t\x9d\x10\xe6w\xdc\xff\xd7o\x9e\x06\x8a\bwQ\xf4\
xe0\xe1\xb50\x00\x00\u07d4\x17\xf1F2\xa7\xe2\x82\v\xe6\xe8\xf6\u07c25X(=\xad\xab-
\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x17\xf5#\xf1\x17\xbc\x9f\xe9x\xaaH\x1e\xb4\xf5V\x17\x117\
x1b\u0209li\xf7>)\x13N\x00\x00\u07d4\x17\xfd\x9bU\x1a\x98\xcba\xc2\xe0\u007f\xbfA\xd3\xe8\u0
2650\u02e5\x89\x01v\x8c0\x81\x93\x04\x80\x00\u07d4\x18\x04x\xa6U\u05cd\x0f;\fO
+aH[\xc4\x00/\u0549lk\x93[\x8b\xbd@\x00\x00\u07d4\x18\x13l\x9d\xf1g\xaa\x17\xb6\xf1\x8e\"\xa7
\x02\u020fK\u0082E\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x18\x15'\x9d\xff\x99R\xda;\xe8\
xf7rI\xdb\xe2\"C7{\xe7\x8a\x01\x01|\xb7n{&d\x00\x00\u07d4\x18\x1f\xbb\xa8R\xa7\xf5\x01x\xb1\x
c7\xf0>\xd9\xe5\x8dT\x16))\x89$\x1a\x9bOaz(\x00\x00\xe0\x94\x18'\x03\x9f\tW\x02\x94\b\x8f\xdd\
xf0G\x16\\3\u65a4\x92\x8a\x02\x05\xb4\u07e1\xeetx\x00\x00\u07d4\x18-
\xb8R\x93\xf6\x06\u8248\xc3pL\xb3\xf0\xc0\xbb\xbf\xcaZ\x89\a?u\u0460\x85\xba\x00\x00\u07d4\x
18H\x00<%\xbf\u052a\x90\xe7\xfc\xb5\u05f1k\xcd\f\xff\xc0\u060965\u026d\xc5\u07a0\x00\x00\xe
0\x94\x18JO\v\xebq\xff\xd5X\xa6\xb6\xe8\xf2(\xb7\x87\x96\xc4\xcf>\x8a\x02\x8a\x85t%Fo\x80\x0
0\x00\xe0\x94\x18M\x86\xf3Fj\xe6h;\x19r\x99\x82\xe7\xa7\u1903G\xb2\x8a\x02\x1e\x19\xe0\u027
a\xb2@\x00\x00\u07d4\x18Q\xa0c\xcc\xdb0T\x90w\xf1\xd19\xe7-
\xe7\x97\x11\x97\u0549lk\x93[\x8b\xbd@\x00\x00\u07d4\x18UF\xe8v\x8dPhs\x81\x8a\xc9u\x1c\x
1f\x12\x11j;\xef\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x18X\xcf\x11\xae\xa7\x9fS\x98\xad+\xb2\"g\xb5\xa3\xc9R\xeat\x8a\x02\x15\xf85
\xbcv\x9d\xa8\x00\x00\xe0\x94\x18Z\u007f\u012c\xe3h\xd23\xe6
\xb2\xa4Y5f\x12\x92\xbd\xf2\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x18d\xa3\u01f4\x81UD\x8c
T\u020cp\x8f\x16g\tsm1\x89\a?u\u0460\x85\xba\x00\x00\u07d4\x18j\xfd\xc0\x85\xf2\xa3\xdc\xe4a
^\xdf\xfb\xad\xf7\x1a\x11x\x0fP\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x18k\x95\xf8\xe5\xef\xfd\xdc\xc9O\x1a1[\xf0)];\x1e\xa5\x88\x89lj\xccg\u05f1\xd4\x
00\x00\u07d4\x18}\x9f\f\a\xf8\xebt\xfa\xaa\xd1^\xbc{\x80Dt\x17\xf7\x82\x89\x01\x15\x8eF\t\x13\xd
0\x00\x00\u07d4\x18\x95\xa0\xebJCrr/\xcb\u016f\xe6\x93o(\x9c\x88\xa4\x19\x891T\xc9r\x9d\x05x
\x00\x00\u07d4\x18\x99\xf6\x9fe;\x05\xa5\xa6\xe8\x1fH\a\x11\u041b\xbf\x97X\x8c\x89i\xfb\x13=\x
f7P\xac\x00\x00\u07d4\x18\xa6\xd2\xfcR\xbes\b@#\xc9\x18\x02\xf0[\xc2JK\xe0\x9f\x89lk\x93[\x8
b\xbd@\x00\x00\u07d4\x18\xb0@|\xda\xd4\xceR`\x06#\xbd^\x1fj\x81\xaba\xf0&\x89\x11Q\xcc\xf0\
xc6T\u0180\x00\u07d4\x18\xb8\xbc\xf9\x83!\xdaa\xfbN>\xac\xc1\xecT\x17'-
\xc2~\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4\x18\xc6r:gS)\x9c\xb9\x14G}\x04\xa3\xbd!\x8d\xf8\xc7u\x
8965\u026d\xc5\u07a0\x00\x00\u07d4\x18\xe1\x13\xd8\x17|i\x1aa\xbexXR\xfa[\xb4z\uef6f\x89Hz\
x9a0E9D\x00\x00\xe0\x94\x18\xe4\xceGH;S\x04\n\u06eb5\x17,\x01\xefdPn\f\x8a\x01\xe7\xe4\x17
\x1b\xf4\u04e0\x00\x00\xe0\x94\x18\xe52C\x98\x1a\xab\xc8v}\xa1\fsD\x9f\x13\x91V\x0e\xaa\x8a\
x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x18\xfa\x86%\xc9\u0704<x\u01eb%\x9f\xf8|\x95\x99\xe0\
u007f\x10\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x18\xfb\t\x18\x8f'\xf1\x03\x8ee@1\x92Ob\x8a!\
x06p=\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x18\xfc\xcfb\xd2\xc39TS\xb7X{\x9e&\xf5\xcf\xf9\xebt\x
82\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x19\x13\x13RR8\xa2\x1cvtW\xa9\x13t\xf0\"\x00\xc5TH
\x89\x06O_\xdfIOx\x00\x00\u07d4\x19\x14\xf1\xeb\x95\xd1'~\x93\xb6\xe6\x1bf\x8b}w\xf1:\x11\xa1
\x894\x95tD\xb8@\xe8\x00\x00\u07d4\x19#\xcf\u018b\x13\xea~ U\x806E\xc1\xe3
\x15k\u060d\x89Hz\x9a0E9D\x00\x00\u07d4\x193j#m\xeduXrA\x1f.\x04\x91\xd8>>\x00\x15\x9e\x

892\xf5\x1e\u06ea\xa30\x00\x00\xe0\x94\x193\xe34\xc4\x0f:\u02ed\f\v\x85\x11X
i$\xbe\xca:\x8a\x01\x99^\xaf\x01\xb8\x96\x18\x80\x00\xe0\x94\x197\xc5\xc5\x15\x05uS\u033dF\u0546dU\xcef)\x02\x84\x8a\xd3\xc2\x1b\xce\xcc\xed\xa1\x00\x00\x00\u07d4\x19:\xc6Q\x83e\x18\x00\xe25\x80\xf8\xf0\xea\u04fbY~\xb8\xa4\x89\x02\xb6*\xbc\xfb\x91\n\x00\x00\u07d4\x19=7\xed4}\x1c/N55\r\x9aDK\xc5|\xa4\xdbC\x89\x03@\xaa\xd2\x1b;p\x00\x00\xe0\x94\x19@\u0713d\xa8R\x16_GAN'\xf5\x00$E\xa4\xf1C\x8a\x02L-
\xffj<|H\x00\x00\u07d4\x19E\xfe7\u007f\xe6\u0537\x1e>y\x1fo\x17\xdb$<\x9b\x8b\x0f\x89vy\u7fb9\x886\x00\x00\u07d4\x19Jk\xb3\x02\xb8\xab\xa7\xa5\xb5y\u07d3\xe0\xdf\x15t\x96v%\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x19L\ubd12\x98\x82\xbf;K\xf9\x86L+\x1b\x0fb\u0083\xf9\x89\x1e\xf8aS\x1ft\xaa\x00\x00\u07d4\x19O\xf4J\xef\xc1{\xd2\x0e\xfdz
LG\xd1b\f\x86\xdb]\x89\xa2\x99\th\u007fj\xa4\x00\x00\xe0\x94\x19O\xfex\xbb\xf5\xd2\r\u044a\x1f\x01\xdaU.\x00\xb7\xb1\x1d\xb1\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4\x19S1>*\xd7F#\x9c\xb2'\x0fH\xaf4\u063b\x9cDe\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x19W\x1a+\x8f\x81\u01bc\xf6j\xb3\xa1\x00\x83)V\x17\x15\x00\x03\x89\x1a\xb2\xcf|\x9f\x87\xe2\x00\x00\xe0\x94\x19h}\xaa9\xc3h\x13\x9bn{\xe6\r\xc1u:\x9f\f\xbe\xa3\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x19l\x02!\nE\n\xb0\xb3cpe_qz\xa8{\xd1\xc0\x04\x89\x0e\x10\xac\xe1W\xdb\xc0\x00\x00\u07d4\x19n\x85\xdf~s+J\x8f\x0e\xd06#\xf4\u06dd\xb0\xb8\xfa1\x89\x01%\xb9/\\\xef$\x80\x00\u07d4\x19s+\xf9s\x05]\xbd\x91\xa4S:\u06a2\x14\x9a\x91\u04c3\x80\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x19vr\xfd9\xd6\xf2F\xcef\xa7\x90\xd1:\xa9\"\xd7\x0e\xa1\t\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x19y\x8c\xbd\xa7\x15\ua69b\x9dj\xab\x94,U\x12\x1e\x98\xbf\x91\x89A\rXj
\xa4\xc0\x00\x00\u07d4\x19\x8b\xfc\xf1\xb0z\xe3\b\xfa,\x02\x06\x9a\xc9\xda\xfeq5\xfbG\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\x19\x8e\xf1\xec2Z\x96\xcc5Lrf\xa08\xbe\x8b\\U\x8fg\x8a\x80\xd1\xe47>\u007f!\xda\x00\x00\xe0\x94\x19\x91\x8a\xa0\x9e}IN\x98\xff\xa5\xdbP5\b\x92\xf7\x15j\u018a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x19\xb3k\f\x87\xeafN\xd8\x03\x18\xdcw\xb6\x88\xdd\xe8}\x95\xa5\x89i\x9fI\x98\x020=\x00\x00\u07d4\x19\u07d4E\xa8\x1c\x1b=\x80J\xea\xebonNB6f?\x89\x02\x06\xd9NjI\x87\x80\x00\u07d4\x19\xe5\u07a37\n,tj\xae4\xa3|S\x1fA\xda&N\x83\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\x19\xe7\xf3\xeb{\xf6\u007f5\x99
\x9e\xbe\b\xb6*\xd32\u007f\x8c\u0789lk\x93[\x8b\xbd@\x00\x00\u07d4\x19\xe9Nb\x00P\xaa\xd7f\xb9\xe1\xba\xd91#\x83\x12\u053fI\x89\x81\xe3-
\xf9r\xab\xf0\x00\x00\u07d4\x19\xec\xf2\xab\xf4\f\x9e\x85{%/\xe1\xdb\xfd=L]\x8f\x81n\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x19\xf5\xca\xf4\xc4\x0ei\b\x81<\aE\xb0\xae\xa9Xm\x9d\xd91\x89#\xfe\xd9\xe1\xfa+`\x00\x00\u07d4\x19\xf6C\xe1\xa8\xfa\x04\xae\x16\x00`(\x13\x833\xa5\x9a\x96\u0787\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x19\xf9\x9f,\vF\u0389\x06\x87]\xc9\xf9\n\xe1\x04\xda\xe3U\x94\x89\xf4WZ]M\x16*\x00\x00\u07d4\x19\xff$O\xcf\xe3\xd4\xfa/O\u065f\x87\xe5[\xb3\x15\xb8\x1e\xb6\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\x1a\x04\xce\xc4
\xadC\"\x15$mw\xfe\x17\x8d3\x9e\u0435\x95\x89\x11!a\x85\u009fp\x00\x00\xe0\x94\x1a\x04\xd58\x9e\xb0\x06\xf9\u0388\f0\xd1SS\xf8\xd1\x1cK1\x8a\x03\x9d\x84\xb2\x18m\xc9\x10\x00\x00\u07d4\x1a\bA\xb9*\u007fpuV\x9d\xc4b~kv\u02b0Z\u0791\x89Rf<\u02b1\xe1\xc0\x00\x00\xe0\x94\x1a\b]C\xec\x92AN\xa2{\x91O\xe7g\xb6\xd4k\x1e\xefD\x8a\x06A\xe8\xa15c\xd8\xf8\x00\x00\u07d4\x1a\t\xfd\xc2\u01e2\x0e#WK\x97\u019e\x93\u07bag\xd3r
\x89lO\xd1\xee$nx\x00\x00\u07d4\x1a\n\x1d\u07f01\xe5\xc8\xcc\x1dF\xcf\x05\x84-
P\xfd\xdcq0\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x1a\x1c\x9a&\xe0\xe0$\x18\xa5\xcfh}\xa7Z'\

\b,\x94@\x8a\x01\x0f\f\xf0d\xddY \x00\x00\u07d4\x1a
\x1bC'\u03a7\xf3\x99\x04bF\xa3\xc8~n\x03\xa3\u0368\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x1
a$4\xccwD\"\u050dS\u055c]V,\u0384\a\xc9K\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\u07d4\x1a%\xe1\u
017c~_P\xec\x16\xf8\x88_!\x0e\xa1\xb98\x80\x0e\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x1a
&\x94\xec\a\xcf^Mh\xba@\xf3\xe7\xa1LS\xf3\x03\x8cn\x8966\xcd\x06\xe2\xdb:\x80\x00\u07d4\x1a
5 E5\x82\xc7\x18\xa2\x1cB7[\xc5\as%RS\xe1\x89*\xd3s\xcef\x8e\x98\x00\x00\xe0\x94\x1a7n\x1b-
/Y\ai\xbb\x85\x8dEu2\rN\x14\x99p\x8a\x01\x06q%v9\x1d\x18\x00\x00\u07d4\x1a:3\x0eO\xcbi\xdb\
xef^i\x01x;\xf5\x0f\xd1\xc1SB\x89\u3bb5sr@\xa0\x00\x00\u07d4\x1aN\u01a0\xae\u007fZ\x94'\xd2
=\xb9rL\r\f\xff\xb2\xab/\x89\t\xb4\x1f\xbf\x9e\n\xec\x00\x00\u07d4\x1aP^b\xa7N\x87\xe5wG>O:\xf
a\x16\xbe\xdd<\xfaR\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x1a^\xe53\xac\xbf\xb3\xa2\xd7
m[hRw\xb7\x96\xc5j\x05+\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1adJP\xcb\u00ae\xe8#\xbd+\xf2
C\xe8%\xbeMG\xdf\x02\x89\x05k\xe0<\xa3\xe4}\x80\x00\u07d4\x1apD\xe28?\x87\b0[I[\xd1\x17k\x
92\xe7\xef\x04:\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x1ay\xc7\xf4\x03\x9cg\xa3\x9du\x13\x88L\xdc\x0e,4\"$\x90\x89\x01\x15\x8eF\t\x1
3\xd0\x00\x00\u07d4\x1a\x89\x89\x9c\xbe\xbd\xbbd\xbb&\xa1\x95\xa6<\bI\x1f\u035e\xee\x89lk\x9
3[\x8b\xbd@\x00\x00\xe0\x94\x1a\x8a\\\xe4\x14\u079c\xd1r\x93~7\xf2\u055c\xffq\xceW\xa0\x8a\x
02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x1a\x95\xa8\xa8\b.FR\xe4\x17\r\xf9'\x1c\xb4\xbbC\x0
5\xf0\xb2\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\x1a\x95\u0277Tk]\x17\x86\u00c5\x8f\x
b1#dF\xbc\f\xa4\u0389j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x1a\x98~?\x83\xdeu\xa4/\x1b\xde|\x99|\
x19!{J_$\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1a\x9ep/8]\xcd\x10^\x8b\x9f\xa4(\xee\xa2\x1cW\xff
R\x8a\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\x1a\xa1\x02\x1fU\n\xf1X\xc7Gf\x8d\xd1;F1`\xf9Z@\x8
9O\xb0Y\x1b\x9b08\x00\x00\u07d4\x1a\xa2v\x99\xca\u068d\u00e7oy3\xaaf\xc7\x19\x19\x04\x0e\
x88\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x1a\xa4\x02p\xd2\x1e\\\u0786\xb61m\x1a\xc3\xc5
3IKy\xed\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x1a\xb5:\x11\xbc\xc6=\u07ea@\xa0+\x9e\x1
8d\x96\u037b\x8a\xff\x89l?*\xac\x80\f\x00\x00\x00\u07d4\x1a\xbcN%;\b\n\xebCy\x84\xab\x05\xbc\
xa0\x97\x9a\xa4>\x1c\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x1a\xc0\x89\u00fcM\x82\xf0j
\x05\x1a\x9ds-
\xc0\xe74\xcba\x89%\xf6\x9dc\xa6\xce\x0e\x00\x00\xe0\x94\x1a\xd4V>\xa5xk\xe1\x15\x995\xab\x
b0\xf1\u0547\x9c>sr\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x1a\xd7-
\xa7n\u007f\xcckv@X\xf4\x8dA}Io\xa6\u0349lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x1a\xda\xf4\xab\xf
a\x86}\xb1\u007f\x99\xafj\xbe\xbfpz<\xf5]\xf6\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x1a\xf6
\x03C6\x0e\v-u%R\x107W
\xdf!\xdb\\}\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x1a\xfc\u0145\x89l\xd0\xed\xe1)\xee-
\xe5\xc1\x9e\xa8\x11T\vd\x89\xaf*\xba\f\x8e[\xef\x80\x00\u07d4\x1b\x05\xeajj\u022f|\xb6\xa8\xb9\
x11\xa8\xcc\xe8\xfe\x1a*\xcf\u0209lk\x93[\x8b\xbd@\x00\x00\u07d4\x1b\v1\xaf\xffKm\xf3e:\x94\xd
7\xc8yx\xae5\xf3J\xae\x89\x139\x10E?\xa9\x84\x00\x00\u07d4\x1b\r\ah\x17\xe8\u058e\xe2\xdfN\
x1d\xa1\xc1\x14-
\x19\x8cD5\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\x1b\x13\ro\xa5\x1d\\H\xec\x8d\x1dR\u070a\
"{\xe8s\\\x8a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1b#\u02c6cUHq\xfb\xbe\r\x9e`9~\xfbo\xae\xdc
>\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x1b&9X\x8bU\xc3D\xb0#\xe8\xde_\xd4\b{\x1f\x04\x03a\x89QP\xae\x84\xa8\xcd\xf
0\x00\x00\u07d4\x1b9

\xd0\x01\xc4>r\xb2N|\xa4o\x0f\xd6\xe0\xc2\n_\xf2\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x1b<\xb8\x1eQ\x01\x1bT\x9dx\xbfr\v\r\x92J\xc7c\xa7\u008av\x95\xa9,
\xd6\xfe\x00\x00\x00\u07d4\x1bC#,\xcdH\x80\xd6\xf4o\xa7Q\xa9l\xd8$s1XA\x89\x04V9\x18$O@\x00\x00\u07d4\x1bK\xbc\xb1\x81e!\x1b&[(\a\x16\xcb?\x1f!!v\xe8\x89\x19\x9a\xd3}\x03\xd0`\x80\x00\u07d4\x1bM\a\xac\u04c1\x83\xa6\x1b\xb2x=+{\x17\x8d\xd5\x02\xac\x8d\x89\n\u05ce\xbcZ\xc6\x00\x00\u07d4\x1bckzIo\x04MsYYn5:\x10F\x16Cok\x89\x13\x88\xea\x95\xc3?\x1d\x00\x00\u07d4\x1bd\x95\x89\x12@\xe6NYD\x93\xc2f!q\xdb^0\xce\x13\x89\tX\x87\u0595\xedX\x00\x00\u07d4\x1bf\x10\xfbh\xba\xd6\xed\x1c\xfa\xa0\xbb\xe3:$\xeb.\x96\xfa\xfb\x89\b=lz\xabc`\x00\x00\u07d4\x1by\x903\xefm\xc7\x12x\"\xf7EB\xbb\"\xdb\xfc\t\xa3\b\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x1b~\xd9t\xb6\xe24\u0381$t\x98B\x9a[\u0520\xa2\xd19\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1b\x82o\xb3\xc0\x12\xb0\xd1Y\u253a[\x8aI\x9f\xf3\xc0\xe0<\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1b\x8a\xa0\x16\f\u05df\x00_\x88Q\nqI\x13\xd7\n\u04fe3\x89\n\xef\xfb\x83\a\x9a\xd0\x00\x00\xe0\x94\x1b\x8b\xd6\xd2\xec\xa2\x01\x85\xa7\x8e}\x98\xe8\xe1\x85g\x8d\xacH0\x8a\x03\x89O\x0eo\x9b\x9fp\x00\x00\u07d4\x1b\x9b-
\u0096\x0eL\xb9@\x8ft\x05\x82|\x9bY\a\x16\x12\xfd\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x1b\xa9\"\x8d8\x87'\xf3\x89\x15\x0e\xa0;s\xc8-
\xe8\xeb.\t\x8a\x01\x89t\xfb\xe1w\xc9(\x00\x00\u07d4\x1b\xa9\xf7\x99~S\x87\xb6\xb2\xaa\x015\xac$R\xfe6\xb4\xc2\r\x89.\x14\x1e\xa0\x81\xca\b\x00\x00\u07d4\x1b\xba\x03\xffkJ\u057f\x18\x18J\xcb!\xb1\x88\xa3\x99\xe9\xebJ\x89a\t=|,m8\x00\x00\u07d4\x1b\xbc\x19\x9eXg\x90\xbe\x87\xaf\xed\xc8I\xc0G&t\\]{\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x1b\xbc`\xbc\xc8\x0e\\\xdc5\xc5Aj\x1f\n@\xa8=\xae\x86{\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1b\xc4L\x87a#\x1b\xa1\xf1\x1f_\xaa@\xfaf\x9a\x01>\x12\u0389\v\tR\xc4Z\xea\xad\x00\x00\u07d4\x1b\xcf4A\xa8f\xbd\xbe\x960\t\xce3\xc8\x1c\xbb\x02a\xb0,\x89\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\u07d4\x1b\u048c\xd5\u01ca\xeeQ5|\x95\xc1\xef\x925\xe7\xc1\x8b\xc8T\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1b\xd8\xeb\xaavt\xbb\x18\u1458\xdb$OW\x03\x13\a_C\x89\b!\xab\rD\x14\x98\x00\x00\u07d4\x1b\xd9\t\xac\rJ\x11\x02\xec\x98\xdc\xf2\u0329j\n\xdc\u05e9Q\x89\x01\x16Q\xac>zu\x80\x00\u07d4\x1b\xe3T,6\x13hte\xf1Zp\xae\xeb\x81f+e\u0328\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1b\xeaM\xf5\x12/\xaf\u07b3`~\xdd\xda\x1e\xa4\xff\u06da\xbf*\x89\x12\xc1\xb6\xee\xd0=(\x00\x00\u07d4\x1b\xecM\x02\u0385\xfcH\xfe\xb6$\x89\x84\x1d\x85\xb1pXj\x9b\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4\x1b\xf9t\u0650OE\u0381\xa8E\xe1\x1e\xf4\xcb\xcf'\xafq\x9e\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\x1c\x04VI\xcdS\xdc#T\x1f\x8e\xd4\xd3A\x81(\b\xd5\u075c\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4\x1c\x12\x8b\xd6\u0365\xfc\xa2uu\xe4\xb4;2S\xc8\xc4\x17*\xfe\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1c\x13\u04c67\xb9\xa4|\xe7\x9d7\xa8oP\xfb@\x9c\x06\a(\x89Hz\x9a0E9D\x00\x00\u07d4\x1c \x10\xbdf-
\xf4\x17\xf2\xa2q\x87\x9a\xfb\x13\xefL\x88\xa3\xae\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x1c%z\u0525Q\x05\xea;X\xed7K\x19\x8d\xa2f\xc8_c\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x1c.6\a\xe1'\xca\xca\x0f\xbd\\YH\xad\xad}\xd80\xb2\x85\x8a\x04+\xf0kx\xed;P\x00\x00\u07d4\x1c5l\xfd\xb9_\xeb\xb7\x14c;(\xd5\xc12\u0744\xa9\xb46\x89\x01Z\xf1\u05cbX\xc4\x00\x00\u07d4\x1c5\xaa\xb6\x88\xa0\u034e\xf8.vT\x1b\xa7\xac9R\u007ft;\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\x1c>\xf0]\xae\x9d\xcb\u0509\xf3\x02D\bf\x9d\xe2D\xc5*\x02\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x1cJ\xf0\xe8c\xd2el\x865\xbco\xfe\xc8\u0759(\x90\x8c\xb5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1c`\x19\x93x\x92\a\xf9e\xbb\x86\\\xbbL\xd6W\xcc\xe7o\xc0\x89\x05T\x1ap7P?\x00

\x00\u07d4\x1cc\xfa\x9e,\xbb\xf2<I\xfc\xde\xf1\u02eb\xfen\r\x1e\x14\xc1\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x1cg\x02\xb3\xb0ZQ\x14\xbd\xbc\xae\xca%S\x1a\xee\xb3H5\xf4\x8a\x05\x85V\xbe\xadE\u072e\x00\x00\u07d4\x1ch\xa6a8x:c\u024c\xc6u\xa9\xecw\xafE\x98\xd3^\x89\x02\xb7F\xf4\x8f\x0f\x12\x00\x00\u07d4\x1cs\xd0\vn%\xd8\xeb\x9c\x1f\xf4\xad\x82{k\x9e\x9c\xf6\xd2\f\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x1cu\x1e\u007f$\u07dd\x94\xa67\xa5\xde\xde\xff\u0142w\xb5\xdb\x19\x89\xae\x8ez\v\xb5u\xd0\x00\x00\xe0\x94\x1c|\xb2\xfek\xf3\xe0\x9c\xbc\xdc\x18z\xf3\x8f\xa8\xf5\x05:p\xb6\x8a\x02\x1c\x84\xf7B\xd0\u03ad\x80\x00\xe0\x94\x1c\x89\x06\x0f\x98|Q\x8f\xa0y\xec,\n^\xbf\xa3\x0f]
\xf7\x8a\b\v\xfb\xef\xcb_\v\xc0\x00\x00\u07d4\x1c\x94\xd66\xe6\x84\xeb\x15X\x95\xcem\xb4\xa2X\x8f\xba\x1d\x00\x1b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1c\x99\xfe\x9b\xb6\xc6\xd1\x06m\x91\x99T\u007f\xd1\U001007ac\u0649lk\x93[\x8b\xbd@\x00\x00\u07d4\x1c\xb4P\x92\x00x\xaa\xb21|}\xb3\xb3\x8a\xf7\xdd)\x8b-
A\x89\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\x1c\xb5\xf3;MH\x896\xd1>1a\xda3\xa1\xda}\xf7\r\x1b\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x1c\xb6\xb2\xd7\xcf\xc5Y\xb7\xf4\x1eoV\xab\x95\xc7\xc9X\xcd\x0eL\x89Hz\x9a0E9D\x00\x00\u07d4\x1c\xc1\xd3\xc1O\x0f\xb8d\x0e6rM\xc42)\xd2\xeaz\x1eH\x89\\(=A\x03\x94\x10\x00\x00\u07d4\x1c\xc9\bv\x00A\t\xcdy\xa3\u07a8f\u02c4\n\xc3d\xba\x1b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1c\xd1\xf0\xa3\x14\u02f2\x00\xde\n\f\xb1\xef\x97\xe9
p\x9d\x97\u0089lk\x93[\x8b\xbd@\x00\x00\u0794\x1c\xdaA\x1b\xd5\x16;\xae\xca\x1eU\x85c`\x1c\xe7
\xe2N\xe1\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x1c\xe8\x1d1\xa7\x920\"\xe1%\xbfH\xa3\xe06\x93\xb9\x8d\xc9\u0749lk\x93[\x8b\xbd@\x00\x00\u07d4\x1c\xeb\xf0\x98]\u007fh\n\xaa\x91\\D\xccb\xed\xb4\x9e\xab&\x9e\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x1c\xedg\x15\xf8b\xb1\xff\x86\x05\x82\x01\xfc\xceP\x82\xb3nb\xb2\x8a\x01j^`\xbe\xe2s\xb1\x00\x00\u07d4\x1c\xf0L\xb1C\x80\x05\x9e\xfd?#\x8be\u057e\xb8j\xfa\x14\u0609\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x1c\xf1\x05\xab#\x02;ULX>\x86\u05d2\x11y\xee\x83\x16\x9f\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x1c\xf2\xebz\x8c\xca\u00ad\xea\xef\x0e\xe8sG\xd55\u04f9@X\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1c\xfc\xf7Q\u007f\f\bE\x97
\x94+dz\u0452\xaa\x9c\x88(\x89+^:\xf1k\x18\x80\x00\x00\xe0\x94\x1d\t\xad$\x12i\x1c\u0141\xc1\xab6\xb6\xf9CL\xd4\xf0\x8bT\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4\x1d\x15|Xv\xc5\xca\xd5S\xc9\x12\xca\xf6\xce-
Rw\xe0\\s\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x1d&\x15\xf8\xb6\xcaP\x12\xb6c\xbd\u0414\xb0\xc5\x13|w\x8d\u07ca\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x1d)\u01ea\xb4+
H\u04b2R%\u0518\u06e6z\x03\xfb\xb2\x89\n\u05ce\xbcZ\xc6
\x00\x00\u0794\x1d4\x1f\xa5\xa3\xa1\xbd\x05\x1f}\xb8\a\xb6\xdb/\u01faO\x9bE\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x1d4N\x96%g\xcb'\xe4M\xb9\xf2\xfa\u01f6\x8d\xf1\xc1\xe6\xf7\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\x1d6h0c\xb7\xe9\xeb\x99F-
\xab\xd5i\xbd\xdc\xe7\x16\x86\xf2\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x1d7aky?\x94\x91\x188\xac\x8e\x19\xee\x94I\u07d2\x1e\u0109QP\xae\x84\xa8\xcd\xf0\x00\x00\xe0\x94\x1d9[0\xad\xda\x1c\xf2\x1f\t\x1aOJ{u3q\x18\x94A\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\x1dEXn\xb8\x03\xca!\x90e\v\xf7H\xa2\xb1t1+\xb5\a\x89K\xe

4\xe7&{j\xe0\x00\x00\u07d4\x1dW.\xdd-
\x87\xca'\x1ag\x14\xc1Z;7v\x1d\u0320\x05\x89\x06\xeb\xd5*\x8d\xdd9\x00\x00\u07d4\x1dc0\x97\
xa8R%\xa1\xffC!\xb1)\x88\xfd\xd5\\+8D\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x1di\xc8=(\xf
f\x04t\xce\xeb\xea\xcb:\xd2'\xa1D\xec\u78ca\x01(\xcc\x03\x92\nb\u0480\x00\u07d4\x1d\x96\xbc\u
0544W\xbb\xf1\xd3\u00a4o\xfa\xf1m\xbf}\x83hY\x89\tIr\t\xd8F~\x80\x00\u07d4\x1d\x9ej\xaf\x80\x
19\xa0_#\x0e]\xef\x05\xaf]\x88\x9b\xd4\xd0\xf2\x89\a?u\u0460\x85\xba\x00\x00\u07d4\x1d\xab\x1
7.\xff\xa6\xfb\xeeSL\x94\xb1~yN\xda\xc5OU\xf8\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x1d\xb9\xa
c\x9a\x9e\xae\xec\nR7W\x05\fq\xf4rx\xc7-
P\x89Hz\x9a0E9D\x00\x00\u07d4\x1d\xbe\x8e\x1c+\x8a\x00\x9f\x85\xf1\xad<\xe8\r.\x055\x0e\u37
09\aW\rn\x9e\xbb\xe4\x00\x00\u07d4\x1d\xc7\xf7\xda\xd8]\xf5?\x12q\x15$\x03\xf4\xe1\xe4\xfd\xb
3\xaf\xa0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x1d\u03bc\xb7em\xf5\u072a3h\xa0U\xd2/\x9e\xd6\xcd\xd9@\x89\x1b\x18\x1eK\xf
24<\x00\x00\xe0\x94\x1d\xd7tA\x84J\xfe\x9c\xc1\x8f\x15\xd8\xc7{\xcc\xfbe^\xe04\x8a\x01\x06\xe
bEW\x99D\x88\x00\x00\u07d4\x1d\xde\xfe\xfd5\xab\x8fe\x8b$q\xe5G\x90\xbc\x17\xaf\x98\u07a4\
x8965\u026d\xc5\u07a0\x00\x00\u07d4\x1d\xee\xc0\x1a\xbe\\\r\x95-
\xe9\x10l=\xc3\x069\xd8P\x05\u0589lk\x93[\x8b\xbd@\x00\x00\u07d4\x1d\xf6\x91\x16rg\x9b\xb0\
xef5\t\x03\x8c\f'\xe3\x94\xfd\xfe0\x89\x1dF\x01b\xf5\x16\xf0\x00\x00\u07d4\x1d\xfa\xee\ar\x12\xf1
\xbe\xaf\x0eo/\x18@Sz\xe1T\xad\x86\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x1e\x06\r\xc6\xc5\
xf1\u02cc\xc7\xe1E.\x02\xee\x16u\b\xb5eB\x8a\x02\xb1O\x02\xc8d\xc7~\x00\x00\xe0\x94\x1e\x1
3\xecQ\x14,\ubde2`\x83A,<\xe3QD\xbaV\xa1\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\x1e\x1aH(\x11\x9b\xe3\t\xbd\x88#nMH+PM\xc5W\x11\x89\xa00\xdc\xeb\xbd/L\x0
0\x00\u07d4\x1e\x1a\ud178leb\u02cf\xa1\xebo\x8f;\xc9\u072eny\x89\xf4\xd2\u0744%\x9b$\x00\x0
0\u07d4\x1e\x1ccQwj\xc3\x10\x919~\xcf\x16\x00-\x97\x9a\x1b-
Q\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\x1e\x1dz_$h\xb9N\xa8&\x98-
\xbf!%y<nJZ\x8964\xf4\x84\x17@\x1a\x00\x00\u07d4\x1e!\x0epG\x88m\xaaR\xaa\xf7\x0fK\x99\x1
d\xach\xe3\x02^\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x1e+\xf4\xba\x8e^\xf1\x8d7\xdemj\xd66\xc4\xca\xe4\x89\xd0\u0309lk\x93[\x8b\xb
d@\x00\x00\u07d4\x1e/\xe4\xe4\xa7}\x14\x1f\xf4\x9a\f\u007f\xbc\x95\xb0\xa2\xb2\x83\xee\xeb\x8
9lk\x93[\x8b\xbd@\x00\x00\u07d4\x1e3\xd1\xc2\xfb^\bO/\x1dT\xbcRgr\u007f\xec?\x98]\x89\x1b\x
1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x1e8\x1a\xdc\xf8\x01\xa3\xbf\x9f\u05ff\xac\x9c\xcc+\x84\x82\
xad^f\x89
\x89r\xc0\x01\rt\x00\x00\xe0\x94\x1e;\xad\xb1\xb6\xe18\x0e'\x03\x9cWj\xe6\".\x96:[S\x8a\x04<3\x
c1\x93ud\x80\x00\x00\u07d4\x1eHM\x06!\xf0\xf53\x1b5\xd5@\x8d\x9a\xaeN\xb1\xac\xf2\x1e\x89\
x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x1eX\x00\"}M\xcfu\xe3\x0fU\x95\u017e\xd3\xf7.4\x1e;\x89\
ru\xda\xcesA~\x00\x00\u07d4\x1eYj\x81\xb3W\xc6\xf2Ip\xcc1=\xf6\xdb\u06ab\xd0\u041e\x89lk\x9
3[\x8b\xbd@\x00\x00\u07d4\x1ei\x15\xeb\u0661\x9c\x81\xb6\x92\xad\x99\xb1!\x8aY,\x1a\u01f1\x
89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x1en\x01S\xfc\x16\x1b\xc0^ek\xbb\x14Lq\x87\xbfO\xe8
M\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1epfU\xe2\x84\xdc\xf0\xbb7\xfe\a]a:\x18\xdc\x12\xffJ\x89\
xedC\xbf\x1e\ue0ac\x00\x00\xe0\x94\x1ex>R*\xb7\xdf\n\u02ac\x9e\xee\xd3Y09\xe5\xacuy\x8a+\x
14F\xddj\xef\xe4\x1c\x00\x00\u07d4\x1e{^M\x1fW+\xec\xf2\xc0\x0f\xc9\f\xb4v{Jn3\u0509\x06\x1f\
xc6\x10u\x93\xe1\x00\x00\u07d4\x1e\x8eh\x9b\x02\x91|\xdc)$]\f\x9ch\xb0\x94\xb4\x1a\x9e\u0589l
k\x93[\x8b\xbd@\x00\x00\u07d4\x1e\xa34\xb5u\b\a\xeat\xaa\u016b\x86\x94\xec_(\xaaw\u03c9\x1

a\xb2\xcf|\x9f\x87\xe2\x00\x00\u07d4\x1e\xa4qU\x04\u01af\x10{\x01\x94\xf4\xf7\xb1\xcbo\xcc\xcd
oK\x89
\x041\x97\xe0\xb0'\x00\x00\u07d4\x1e\xa4\x92\xbc\xe1\xad\x10~3\u007fK\u0527\xac\x9a{\xab\xc
c\u036b\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x1e\xa6\xbf/\x15\xae\x9c\x1d\xbcd\u06a7\xf8\xeaM\r\x81\xaa\xd3\xeb\x89\u
3bb5sr@\xa0\x00\x00\u07d4\x1e\xb4\xbfs\x15j\x82\xa0\xa6\x82
\x80\xc6\xed\xf4\x9cF\x9a\xf8\xb9\x89g\x8a\x93
b\xe4\x18\x00\x00\xe0\x94\x1e\xba\xcbxD\xfd\xc3\"\xf8\x05\x90O\xbf\x19b\x80-
\xb1S|\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x1e\xc4\xecKw\xbf\x19\u0411\xa8h\xe6\
xf4\x91T\x18\x05A\xf9\x0e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x1e\xd0n\xe5\x16b\xa8lcE\x88\xf
bb\xdcC\xc8\xf2~|\x17\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x1e\u063b?\x06w\x8b\x03\x9e\x99a\xd8\x1c\xb7\x1as\xe6x|\x8e\x89lk\x93[\x8b\x
bd@\x00\x00\u07d4\x1e\xda\bNye\x00\xba\x14\xc5\x12\x1c\r\x90\x84of\xe4\xbeb\x89\x1c\xfd\xd7
F\x82\x16\xe8\x00\x00\u07d4\x1e\xeel\xbe\xe4\xfe\x96\xadaZ\x9c\xf5\x85zdy@\u07ccx\x89\x01\r:
\xa56\xe2\x94\x00\x00\u07d4\x1e\xf2\u073f\xe0\xa5\x00A\x1d\x95n\xb8\u0213\x9c=l\xfef\x9d\x89
*\x11)\u0413g
\x00\x00\xe0\x94\x1e\xf5\xc9\xc76P\u03fb\xde\\\x88U1\xd4'\xc7\xc3\xfeUD\x8a\x01EB\xba\x12\xa
37\xc0\x00\x00\u07d4\x1f\x04\x12\xbf\xed\u0356N\x83}\t,q\xa5\xfc\xba\xf3\x01&\xe2\x89\x01\x15\
x8eF\t\x13\xd0\x00\x00\u07d4\x1f\x17O@\xa0Dr4\xe6fS\x91Mu\xbc\x00>V\x90\u0709\b\xacr0H\x
9e\x80\x00\x00\u07d4\x1f!\x86\xde\xd2>\f\xf9R\x16\x94\xe4\xe1dY>i\n\x96\x85\x89\x10CV\x1a\x8
8)0\x00\x00\u07d4\x1f*\xfc\n\xed\x11\xbf\xc7\x1ew\xa9\ae{6\xeav\xe3\xfb\x99\x89\xd8\xd7&\xb7\x
17z\x80\x00\x00\u0794\x1f9Y\xfc)\x11\x10\xe8\x822\xc3kvg\xfcx\xa3ya?\x88\xfc\x93c\x92\x80\x1c
\x00\x00\u07d4\x1f=\xa6\x8f\xe8~\xafC\xa8)\xabm~\u0166\xe0\t\xb2\x04\xfb\x89\x1e\x16\x01u\x8
c,~\x00\x00\u07d4\x1fI\xb8m\r9EY\x06\x98\xa6\xaa\xf1g<7u\\\xa8\r\x89%\xf2s\x93=\xb5p\x00\x00
\u07d4\x1f_;4\xbd\x13K'\x81\xaf\xe5\xa0BJ\u0144l\xde\xfd\x11\x89\x05]\xe6\xa7y\xbb\xac\x00\x0
0\u07d4\x1fo\x0004\x97R\x06\x1c\x96\a+\xc3\xd6\xeb5I
\x8dk\x89\x01K\x8d\xe1\xeb\x88\u06c0\x00\u07d4\x1f}\x8e\x86\xd6\xee\xb0%E\xaa\xd9\x0e\x912
{\xd3i\xd7\xd2\xf3\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x1f\x81\x16\xbd\n\xf5W\x0e\xaf\fV\u
011cz\xb5\xe3zX\x04X\x89lk\x93[\x8b\xbd@\x00\x00\u0794\x1f\x88\xf8\xa13\x8f\xc7\xc1\tv\xab\x
cd?\xb8\u04c5T\xb5\uc708\xb9\xf6]\x00\xf6<\x00\x00\u07d4\x1f\x9c2hE\x8d\xa3\x01\xa2\xbeZ\xb
0\x82W\xf7{\xb5\xa9\x8a\xa4\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\x1f\xa21\x9f\xed\x8c-
F*\xdf.\x17\xfe\xecjo0Qn\x95\x89\x06\xca\xe3\x06!\xd4r\x00\x00\u07d4\x1f\xb4c\xa08\x99\x83\xdf
}Y?{\xddmxI\u007f\xed\x88y\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x1f\xb7\xbd1\r\x95\xf2\xa
6\u067a\xaf\x8a\x8aC\n\x9a\x04E:\x8b\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x1f\xcc|\xe6\xa8H
X\x95\xa3\x19\x9e\x16H\x1fr\xe1\xf7b\xde\xfe\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x1f\xcf\xd1\
xd5\u007f\x87\"\x90V\f\xb6-
`\x0e\x1d\xef\xbe\xfc\xcc\x1c\x89P\xc5\xe7a\xa4D\b\x00\x00\u0794\x1f\u0496\xbe\x03\xads|\x92\
xf9\u0186\x9e\x8d\x80\xa7\x1cW\x14\xaa\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\x1f\xdd\xd8_\u
024b\xe9\xc4\x04Ya\xf4\x0f\x93\x80^\xccEI\xe5\x89\b\xe3\xf5\v\x17<\x10\x00\x00\u07d4
\x01\xbe\xf7{f\xf5\x1e\x15\x99\xb0/\xb1\x10\x19J\x00\x99\xb7\x8d\x89lk\x93[\x8b\xbd@\x00\x00\u
07d4
\x02d\xa0\x9f\x8ch\xe3\xe6b\x97\x95(\x0fV%O\x86@\u0409\x01\x15\x8eF\t\x13\xd0\x00\x00\u07

d4
\x03qy\a\xa7%`\xf40\u007f\x1b\xee\xccT6\xf4=!\xe7\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4
\r\xfc\vq\xe3Y\xb2\xb4eD\n6\xa6\xcd\xc3Rw0\a\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4
\x13L\xbf\xf8\x8b\xfa\xdcFkR\xec\ua9d8W\x89\x1d\x83\x1e\x8965\u026d\xc5\u07a0\x00\x00\u07d
4 \x14&\x1f\x01\b\x9fSyV0\xba\x9d\xd2O\x9a4\xc2\xd9B\x89Hz\x9a0E9D\x00\x00\u07d4
\x16\x89]\xf3,\x8e\xd5G\x82iF\x84#\xae\xa7\xb7\xfb\xceP\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u0
7d4 \x18\x1cKA\xf6\xf9r\xb6iX!_\x19\xf5p\xc1]\xdf\xf1\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4
\x18d\xa8\xf7\x84\xc2'{\v|\x9e\xe74\xf7\xb3w\xea\xb6H\x89\xf2(\x14\x00\xd1\xd5\xec\x00\x00\u07
d4 \xb8\x1a\xe59&\xac\xe9\xf7\xd7AZ\x05\f\x03\x1dX_
\x89\x12\u007f\x19\xe8>\xb3H\x00\x00\xe0\x94
<b\x83\xf2\r\xf7\xbc\x86T/\u07f4\xe7c\xec\u06fb\xee\xf5\x8a\x05K@\xb1\xf8R\xbd\xa0\x00\x00\xe
0\x94
J\u0248g\xa7\xc9\xc7\xedq\x1c\xb8/(\xa8x\xca\xf6\x9bH\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u0
7d4 R7\u013e\x14o\xba\x99G\x8f:}\xad\x17\xb0\x918\u0695\x89lk\x93[\x8b\xbd@\x00\x00\u07d4
S\xac\x97T\x8a\fN\x8b\x80\xbcrY\f\u05a0\x98\xfeu\x16\x89\n#%u;F\f\x00\x00\u07d4
_Qf\xf1$@\xd8Wb\xc9g\u04ee\x86\x18O\x8fM\x98\x89\x17r$\xaa\x84Lr\x00\x00\xe0\x94
_\xc8C\xe1\x9aI\x13\u0448\x1e\xb6\x9bi\xc0\xfa;\xe5\xc5\v\x8a\x02\r\u058a\xaf2\x89\x10\x00\x00
\u07d4
d\x82\xeeo\x13\x8aw\x8f\xe1\xadb\xb1\x80\u0385o\xbb#\xe6\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x
94 fwM\x82'\x93\xff%\xf1v\t\tG\x9c\xf6$\x91\xbf\x88\x8a\v\xae:\u0185\xcbr\xe0\x00\x00\u07d4
mU\xd5y*QN\xc1\b\xe0\x90Y\x9f*\x06^P\x11\x85\x89\n\xdf0\xbap\u0217\x00\x00\u07d4
p~B]*\x11\xd2\u021f9\x1b+\x80\x9fUlY$!\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94
~\xf8\v]`\xb6\xfb\xff\xc5\x1f:d\xb8\xc7 6\xa5\xab\xbd\x8a\x01je\x02\xf1Z\x1eT\x00\x00\xe0\x94
\x82K\xa1\xdb\xeb\xbe\xf9\x84n\xf3\xd0\xf6\xc1\xb0\x17\xe6\x91.\u010a\x01\x84\xb2nM\xaf\x1d5\
x00\x00\u07d4
\x84\xfc\xe5\x05\xd9{\xeb\xf1\xad\x8c_\xf6\x82o\xc6E7\x1f\xb2\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\u
07d4
\x8cEs,\n7\x8f\x17\xac\x83$\x92mE\x9b\xa8\xb6X\xb4\x89\xa00\xdc\xeb\xbd/L\x00\x00\u07d4
\x93w\xb6\xad?\xe1\x01\xc9h[5vT\\k\x16\x84\xe7<\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4
\x9e\x8e)\xd3;\xea\xe8\xfbk\xaax=\x13>\x1d\x9e\xc1\xbc\v\x89-C\xf3\xeb\xfa\xfb,\x00\x00\u07d4
\xa1RV\xd5\f\xe0X\xbf\x0e\xacC\xaaS:\xa1n\u0273\x80\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d
4
\xa2\x9cPy\xe2k?\x181\x8b\xb2\xe5\x0e\x8e\x8b4n[\xe8\x89\x1b\x1a\xb3\x19\xf5\xecu\x00\x00\u0
7d4 \xa8\x16\x80\xe4e\xf8\x87\x90\xf0\aO`\xb4\xf3_]\x1ej\xa5\x89Ea\x80'\x8f\fw\x80\x00\u07d4
\xb9\xa9\u6f48\x80\u0659J\xe0\r\u0439(*\v\xea\xb8\x16\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u0
7d4 \u0084\xba\x10\xa2\b0\xfc=i\x9e\xc9}-\xfa'\xe1\xb9^\x89lk\x93[\x8b\xbd@\x00\x00\u07d4
\xd1A\u007f\x99\xc5i\u3fb0\x95\x85e0\xfe\x12\xd0\xfc\uaa89@\x15\xf9K\x11\x83i\x80\x00\u07d4
\u074f\u02f4n\xa4o\u3066\x8b\x8c\xa0\xea[\xe2\x1f\u9949lk\x93[\x8b\xbd@\x00\x00\xe0\x94
\xff>\u078c\xad\xb5\xc3{H\xcb\x14X\x0f\xb6^#\t\n{\x8a\b\xe4\xd3\x16\x82v\x86@\x00\x00\xe0\x94
!\x008\x1d`\xa5\xb5J\xdc\t\u0456\x83\xa8\xf6\u057bK\xfb\u02ca\x02\x1e\x19\xe0\u027a\xb2@\x0
0\x00\xe0\x94!\x18\xc1\x16\xab\f\xdfo\xd1\x1dT\xa40\x93\a\xb4w\xc3\xfc\x0f\x8a\x02\x1e\x19\xe0
\u027a\xb2@\x00\x00\xe0\x94!\x1b)\xce\xfcy\xae\x97gD\xfd\xeb\u03bd<\xbb2\xc5\x13\x03\x8a\x0

2\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4!
l\xe2.\xa4\x80\xe8Y@\xd3\x13\x14\xe0\xd6ONM:\x04\x8965\u026d\xc5\u07a0\x00\x00\u07d4!2\xc0Qj.\x17\x17J\xc5G\xc4;{\x00
\xd1\xebLY\x895e\x9e\xf9?\x0f\xc4\x00\x00\xe0\x94!@\x8bMz,\x0en\xcaAC\xf2\xca\u037b\u033a\x12\x1b\u060a\x04<3\xc1\x93ud\x80\x00\x00\u07d4!Kt9U\xa5\x12\xden\r\x88j\x8c\xbd\x02\x82\xbe\xe6\u04a2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4!L\x89\u017d\x8e}\"\xbcWK\xb3^H\x95\x02\x11\xc6\xf7v\x89\x01\x06T\xf2X\xfd5\x80\x00\xe0\x94!Ti\x14\xdf\u04ef*\xddA\xb0\xff>\x83\xff\xdat\x14\xe1\xe0\x8a\x01C\x95\xe78ZP.\x00\x00\u07d4!X.\x99\xe5\x02\xcb\xf3\xd3\xc2;\xdf\xfbv\xe9\x01\xacmV\xb2\x89\x05k\xc7^-
c\x10\x00\x00\u07d4!Y$\b\x13\xa70\x95\xa7\xeb\xf7\u00f3t>\x80(\xae_\t\x89lk\x93[\x8b\xbd@\x00\x00\u07d4!`\xb4\xc0,\xac\n\x81\u0791\b\xdeCE\x90\xa8\xbf\xe6\x875\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94!nA\x86N\xf9\x8f\x06\r\xa0\x8e\xca\xe1\x9a\xd1\x16j\x17\xd06\x8a\x016\x9f\xb9a(\xacH\x00\x00\u07d4!\x84o/\xdfZA\xed\x8d\xf3n^\xd8TM\xf7Y\x88\xec\xe3\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94!\xa6\xdbe'F{\xc6\xda\xd5K\xc1n\x9f\xe2\x95;g\x94\xed\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4!\xa6\xfe\xb6\xab\x11\xc7f\xfd\xd9w\xf8\xdfA!\x15_G\xa1\xc0\x89\x03\x19\xcf8\xf1\x00X\x00\x00\u07d4!\xb1\x82\xf2\xda+8D\x93\xcf_5\xf8=\x9d\x1e\xe1O*!\x89lk\x93[\x8b\xbd@\x00\x00\u07d4!\xbf\xe1\xb4\\\xac\xdebt\xfd\x86\b\u0661x\xbf>\xebn\u0709l\xee\x06\u077e\x15\xec\x00\x00\u07d4!\xc0s\x80HOl\xbc\x87$\xad2\xbc\x86L;Z\xd5\x00\xb7\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94!\u00e8\xbb\xa2g\xc8\u0322{\x1a\x9a\xfa\xba\xd8o`z\xf7\b\x8a\x01\xe4\xa3Il\u06580\x00\x00\u07d4!\xcem[\x90\x18\xce\xc0J\u0596yD\xbe\xa3\x9e\x800\xb6\xb8\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4!\xd0'\x05\xf3\xf6I\x05\xd8\x0e\xd9\x14y\x13\xea\x8cs\a\u0595\x89l\xed\xb1\xc0\x98\x876\x00\x00\u07d4!\xd1?\f@$\xe9g\xd9G\a\x91\xb5\x0f\"\xde:\xfe\xcf\x1b\x89\xf1Z\xd3^.1\xe5\x00\x00\xe0\x94!\xdb\u06c1z\r\x84\x04\u01bd\xd6\x15\x047N\x9cC\xc9!\x0e\x8a\x02\x1e\x18\xb9\xe9\xabE\xe4\x80\x00\xe0\x94!\xdf\x1e\xc2KNK\xfey\xb0\xc0\x95\u03ba\xe1\x98\xf2\x91\xfb\u044a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94!\xdf-
\u036ft\xb2\xbf\x804\x04\xddM\xe6\xa3^\xab\xec\x1b\xbd\x8a\x01w\"J\xa8D\xc7
\x00\x00\u07d4!\xe2\x19\u021c\xa8\xac\x14\xaeL\xbaa0\xee\xb7}\x9em9b\x89*\u035f\xaa\xa08\xee\x00\x00\u07d4!\xe5\u04ba\xe9\x95\xcc\xfd\b\xa5\xc1k\xb5$\xe1\xf60D\x8f\x82\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4!\xe5\xd7s 0L
\x1c\x1eS\xb2a\xa1#\u0421\x06>\x81\x89\x04\xb6\xfa\x9d3\xddF\x00\x00\xe0\x94!\xea\xe6\xfe\xff\xa9\xfb\xf4\u0347OG9\xac\xe50\u033eY7\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4!\xec\xb2\u07e6Wy\xc7Y-
\x04\x1c\xd2\x10Z\x81\xf4\xfdNF\x8965\u026d\xc5\u07a0\x00\x00\u07d4!\uff20\x9b5\x80\xb9\x8es\xf5\xb2\xf7\xf4\xdc\v\xf0,R\x9c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4!\xfd\v\xad\xe5\xf4\xeftt\xd0X\xb7\xf3\xd8T\xcb\x13\x00RN\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94!\xfdG\xc5%`\x12\x19\x8f\xa5\xab\xf11\xc0mj\xa1\x96_u\x8a\x01\xab,\xf7\xc9\xf8~
\x00\x00\u07d4!\xfdl]\x97\xf9\xc6\x00\xb7h!\xdd\xd4\xe7v5\x0f\xce+\xe0\x89lj\u04c2\xd4\xfba\x00\x00\u07d4\"\r\u018d\xf0\x19\xb6\xb0\u033f\xfbxKZZ\xb4\xb1]@`\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\"\x0e+\x92\xc0\xf6\xc9\x02\xb5\x13\xd9\xf1\xe6\xfa\xb6\xa8\xb0\xde\xf3\u05c9+^:\xf1k\x18\x80\x00\x00\u07d4\"V\x1cY1\x14560\x9c\x17\xe82X{b\\9\v\x9a\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\"W\xfc\xa1jn*d|<)\xf3l\xe2)\xab\x93\xb1~\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\"]5\xfa\xed\xb3\x91\u01fc-

\xb7\xfa\x90q\x16\x04\x05\x99m\x00\x89\t\x18T\xfc\x18bc\x00\x00\u07d4\"_\x9e\xb3\xfbo\xf3\xe9\xe3\xc8D~\x14\xa6n\x8dO7y\xf6\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\"r\x18n\xf2}\xcb\xe2\xf5\xfc70P\xfd\xae\u007f*\xce#\x16\x8a\x03h\xc8b:\x8bM\x10\x00\x00\u07d4\"s\xba\u05fcNHv\"\xd1u\xefzf\x98\x8bj\x93\xc4\xee\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\"v&K\xec\x85&\xc0\xc0\xf2pgz\xba\xf4\xf0\xe4A\xe1g\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\"\x82B\xf83n\xec\xd8$.\x1f\x00\x0fA\x93~q\xdf\xfb\xbf\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\"\x84*\xb80\xdaP\x99\x13\xf8\x1d\xd1\xf0O\x10\xaf\x9e\xdd\x1cU\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\"\x94O\xbc\xa9\xb5yc\bN\xb8M\xf7\xc8_\xb9\xbc\u07f8V\x89\xfc\x11\x8f\uf43a8\x80\x00\u07d4\"\x9c\xc4q\x1bbu^\xa2\x96DZ\u00f7\u007f\xc63\x82\x1c\xf2\x89\x02#\xe8\xb0R\x192\x80\x00\u0794\"\x9eC\r\xe2\xb7OD&Q\xdd\u0377\x01v\xbc\x05L\xadT\x88\xbb\xf9\x81\xbcJ\xaa\x80\x00\u07d4\"\x9fO\x1a*OT\atP[G\a\xa8\x1d\xe4D\x10%[\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\"\x9f\xf8\v\xf5p\x80\t\xa9\xf79\xe0\xf8\xb5`\x91@\x16\u0566\x89\x12\x11\xec\xb5m\x13H\x80\x00\u07d4\"\xa2X\x12\xabV\xdc\xc4#\x17^\xd1\u062d\xac\xce3\xcd\x18\x10\x89dI\xe8NG\xa8\xa8\x00\x00\xe0\x94\"\xb9j\xb2\xca\xd5]\xb1\x00\xb50\x01\xf9\xe4\xdb7\x81\x04\xc8\a\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\"\xbd\xff\xc2@\xa8\x8f\xf7C\x1a\xf3\xbf\xf5\x0e\x14\xda7\xd5\x18>\x8965\u026d\xc5\u07a0\x00\x00\u07d4\"\xce4\x91Y\xee\xb1D\xef\x06\xff&6X\x8a\xefy\xf6(2\x89\n1\x06+\xee\xedp\x00\x00\u07d4\"\xdbU\x9f,<\x14u\xa2\xe6\xff\xe8:YyY\x91\x96\xa7\xfa\x8965\u026d\xc5\u07a0\x00\x00\u07d4\"\xe1QX\xb5\xee>\x86\xeb\x032\xe3\u6a6cl\u0675^\u0349\b\xacr0H\x9e\x80\x00\x00\u07d4\"\xe2H\x8e-
\xa2jI\xae\x84\xc0\x1b\xd5K!\xf2\x94x\x91\u0189]\u0212\xaa\x111\xc8\x00\x00\u07d4\"\xe5\x12\x14\x9a\x18\xd3i\xb7<q\xef\xa4>\x86\xc9\xed\xab\xaf\x1d\x89N\xe0.g\x14a\\\x00\x00\u07d4\"\xeb}\xb0\xbaV\xb0\xf8\xb8\x16\u0332\x06\xe6\x15\xd9)\x18[\r\x89\x04])s~\"\xf2\x00\x00\u07d4\"\xee\xd3'\xf8\xeb\x1d\x138\xa3\xcb{\x0f\x8aK\xaaY\a\u0355\x89\x01E]_Hw\b\x80\x00\xe0\x94\"\xf0\x04\u07cd\xe9\xe6\xeb\xf5#\u032c\xe4W\xac\xcb&\xf9r\x81\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u0794\"\xf2\xdc\xffZ\u05cc>\xb6\x85\v\\\xb9Q\x12{e\x95\"\u623e -
j\x0e\xda\x00\x00\u07d4\"\xf3\xc7y\xddy\x02>\xa9*x\xb6\\\x1a\x17\x80\xf6-
\\J\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\"\xfe\x88M\x907)\x1bMR\xe6(Z\xe6\x8d\xea\v\xe9\xff\xb5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4#\x06\u07d3\x1a\x94\rX\xc0\x16e\xfaM\b\x00\x80,\x02\xed\xfe\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94#\t\xd3@\x91D[22Y\v\xd7\x0fO\x10\x02[,\x95\t\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4#\x12\x00F\xf6\x83!\x02\xa7R\xa7fVi\x1c\x86>\x17\u5709\x11\xe0\xe4\xf8\xa5\v\xd4\x00\x00\u07d4#\x1a\x15\xac\xc1\x99\u021f\xa9\xcb\"D\x1c\xc7\x030\xbd\xcc\xe6\x17\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4#\x1d\x94\x15]\xbc\xfe*\x93\xa3\x19\xb6\x17\x1fc\xb2\v\u04b6\xfa\x89\xcf\x14{\xb9\x06\xe2\xf8\x00\x00\u07d4#(2\xcdYw\xe0\nL0\xd0\x16?.$\xf0\x88\xa6\xcb\t\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4#,m\x03\xb5\xb6\xe6q\x1e\xff\xf1\x90\xe4\x9c(\xee\xf3l\x82\xb0\x89Hz\x9a0E9D\x00\x00\xe0\x94#,\xb1\xcdI\x99<\x14J?\x88\xb3a\x1e#5i\xa8k\u058a\x03L`lB\u042c`\x00\x00\u07d4#,\xe7\x82Pb%\xfd\x98`\xa2\xed\xc1Jz0Gsm\xa2\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4#/R]U\x85\x9b}N`\x8d
H\u007f\xaa\xdb\x00)15\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94#4\u0150\u01e4\x87i\x100E\u0176SL\x8a4i\xf4J\x8a\x03\xb1\x99\a=\xf7-
\xc0\x00\x00\u07d4#7n\u02bftl\xe53!\xcfB\xc8fI\xb9+g\xb2\xff\x89lk\x93[\x8b\xbd@\x00\x00\u07d4#7\x8fB\x92m\x01\x84\xb7\x93\xb0\xc8'\xa6\xdd>=3O\u0349\x03\t'\xf7L\x9d\xe0\x00\x00\u07d4#8B\xb1\xd0i/\xd1\x11@\xcfZ\u0364\xbf\x960\xba\xe5\xf8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4#9\x

e9I(p\xaf\xea%7\xf3\x89\xac/\x83\x83\x02\xa3<\x06\x89lk\x93[\x8b\xbd@\x00\x00\u07d4#;\xdd\xd
d]\xa9HR\xf4\xad\xe8\xd2\x12\x88V\x82\xd9\ak\u0189\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4#OF
\xba\xb7?\xe4]1\xbf\x87\xf0\xa1\xe0Fa\x99\xf2\ubb09\x1aJ\xba\"\\
t\x00\x00\u07d4#U\x1fV\x97_\xe9+1\xfaF\x9cI\xeaf\xeefb\xf4\x1e\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4#V\x95B\xc9}V`\x18\xc9\a\xac\xfc\xf3\x91\xd1@g\xe8~\x89lk\x93[\x8b\xb
d@\x00\x00\xe0\x94#_\xa6l\x02^\xf5T\x00p\xeb\xcf\r7-
\x81w\xc4g\xab\x8a\a\x12\x9e\x1c\xdf7>\xe0\x00\x00\xe0\x94#r\xc4\xc1\u0253\x9fz\xafl\xfa\xc0@
\x90\xf0\x04t\x84\n\t\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4#s\f5z\x91\x02nD\xb1\xd0\x
e2\xfc*Q\xd0q\xd8\xd7{\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4#v\xad\xa9\x033\xb1\u0441\bL
\x97\xe6E\xe8\x10\xaa[v\xf1\x89(\xa8WBTf\xf8\x00\x00\u07d4#x\xfdC\x82Q\x1e\x96\x8e\u0452\x
10g7\xd3$\xf4T\xb55\x8965\u026d\xc5\u07a0\x00\x00\u07d4#\x82\xa9\u050e\xc8>\xa3e(\x90\xfd\
x0e\u7710{[-
\xc1\x89\a?u\u0460\x85\xba\x00\x00\u07d4#\x83\xc2\"\xe6~\x96\x91\x90\xd3!\x9e\xf1M\xa3xP\xe
2lU\x89lk\x93[\x8b\xbd@\x00\x00\u07d4#\x8akv5%/RDHl\n\xf0\xa7:
s\x85\xe09\x89JD\x91\xbdm\xcd(\x00\x00\u07d4#\x9as>k\x85Z\u0152\xd6c\x15a\x86\xa8\xa1t\xd
2D\x9e\x89X\xbe7X\xb2A\xf6\x00\x00\xe0\x94#\xab\t\xe7?\x87\xaa\x0f;\xe0\x13\x9d\xf0\xc8\xebk\
xe5cO\x95\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94#\xab\xd9\xe9>yW\xe5\xb66\xbeey\x05\x
1c\x15\xe5\xce\v\x0e\x8a\x03\xa3\xc8\xf7\xcb\xf4,8\x00\x00\u07d4#\xb1\u0111\u007f\xbd\x93\xee
=H8\x93\x06\x95s\x84\xa5Il\xbf\x89\xd8\xd8X?\xa2\xd5/\x00\x00\xe0\x94#\xba8d\xdaX=\xabV\xf4
\x87<7g\x96\x90\xe0/\x00\x8a\x02\x13BR\r_\xec
\x00\x00\u07d4#\xc5Z\xebW9\x87o\n\xc8\xd7\xeb\xea\x13\xber\x96\x85\xf0\x00\x89Hz\x9a0E9D\
x00\x00\u07d4#\u025b\xa0\x87D\x8e\x19\xc9p\x1d\xf6n\f\xabR6\x831\xfa\x89lk\x93[\x8b\xbd@\x
00\x00\u07d4#\xcc\xc3\u01ac\xd8\\.F\fO\xfd\xd8+\xc7]\xc8I\xea\x14\x89\xd8\xd7&\xb7\x17z\x80\x
00\x00\u07d4#\xcd%\x98\xa2\x0e\x14\x9e\xad*\u0593yWn\xce\xdb`\u3389lk\x93[\x8b\xbd@\x00\
x00\u07d4#\u07cfH\xee\x00\x92V\xeay~\x1f\xa3i\xbe\xeb\xcfk\xc6c\x89|\xd3\xfa\xc2m\x19\x81\x8
0\x00\u07d4#\xe2\u01a8\xbe\x8e\n\u03e5\xc4\xdf^6\x05\x8b\xb7\u02ecZ\x81\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4#\xeaf\x9e5d\x81\x9a\x83\xb0\xc2l\x00\xa1m\x9e\x82olF\x89M\x8dl\xa9h\xca\x1
3\x00\x00\u07d4#\xebo\xd8Vq\xa9\x06:\xb7g\x8e\xbe&Z
\xf6\x1a\x02\xb3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4#\xf9\xec\xf3\xe5\xdd\u0723\x88\x15\xd3\xe
5\x9e\xd3K[\x90\xb4\xa3S\x89\v\x17\x81\xa3\xf0\xbb
\x00\x00\u07d4#\xfa~\xb5\x1aH\"\x95\x98\xf9~v+\xe0\x86\x96R\xdf\xfcf\x8965\u026d\xc5\u07a0\x
00\x00\xe0\x94$\x03\x05rs\x13\xd0\x1esT,w_\xf5\x9d\x11\xcd5\xf8\x19\x8a\x01A\x88Vf\x80\u007f
\\\x80\x00\u07d4$\x04k\x91\u069ba\xb6)\u02cb\x8e\xc0\xc3Q\xa0~\a\x03\xe4\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4$\x0eU\x9e'J\xae\xf0\xc2X\x99\x8c\x97\x9fg\x1d\x11s\xb8\x8b\x89\xd8\xd7&\xb
7\x17z\x80\x00\x00\xe0\x94$\x13aU\x9f\xee\xf8\x0e\xf170!S\xbd\x9e\xd2\xf2]\xb3\xef\x8a\x04<3\x
c1\x93ud\x80\x00\x00\xe0\x94$;;\xcaj)\x93Y\xe8\x86\xce3\xa3\x03A\xfa\xfeMW=\x8a\x04<3\xc1\x
93ud\x80\x00\x00\u07d4$<\x84\xd1$ W\f\xc4\xef;\xab\xa1\xc9Y\u0083$\x95
\x89\u007f\x1fi\x93\xa8S\x04\x00\x00\xe0\x94$CJ>2\xe5N\xcf'/\xe3G\v_oQ/gU
\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d4$HYo\x91\xc0\x9b\xaa0\xbc\x96\x10j-
7\xb5p^](\x89lk\x93[\x8b\xbd@\x00\x00\u0794$Xn\xc5E\x175\xee\xaa\xebG\r\xc8sj\xaeu/\x82\xe5
\x88\xf4?\xc2\xc0N\xe0\x00\x00\u07d4$X\xd6U_\xf9\x8a\x12\x9c\xce@7\x95=\x00
n\xffB\x87\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4$b\x91\x16[Y3-

\xf5\xf1\x8c\xe5\u0248V\xfa\xe9X\x97\u0589\\(=A\x03\x94\x10\x00\x00\u07d4$g\u01a5\u0196\xed
\xe9\xa1\xe5B\xbf\x1a\xd0k\xccK\x06\xac\xa0\x89\x01\x00\xbd3\xfb\x98\xba\x00\x00\u07d4$v\xb2
\xbbu\x1c\xe7H\xe1\xa4\xc4\xff{#\v\xe0\xc1]\"E\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4$z\n\x1
1\xc5\u007f\x03\x83\xb9I\xdeT\vf\xde\xe6\x86\x04\xb0\xa1\x899\xfb\xae\x8d\x04-
\xd0\x00\x00\u07d4$\x87\xc3\u013e\x86\xa2r=\x91|\x06\xb4XU\x01p\xc3\xed\xba\x8965\u026d\xc
5\u07a0\x00\x00\u07d4$\x89\xac\x12i4\xd4\u05a9M\xf0\x87C\xda{v\x91\xe9y\x8e\x8965\u026d\x
c5\u07a0\x00\x00\u07d4$\x9d\xb2\x9d\xbc\x19\xd1#]\xa7)\x8a\x04\b\x1c1WB\u9b09a\xac\xff\x81\
xa7\x8a\xd4\x00\x00\u07d4$\xa4\xeb6\xa7\xe4\x98\xc3o\x99\x97\\\x1a\x8dr\x9f\u05b3\x05\u05c9\
r\xfcx!\x0e\xb2\xc8\x00\x00\u07d4$\xa7P\xea\xe5\x87G\x11\x11m\xd7\xd4{q\x86\u0399\r1\x03\x8
9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4$\xaa\x11Q\xbbv_\xa3\xa8\x9c\xa5\x0e\xb6\xe1\xb1\xc7\x06A\u007f\u0509\xa8\r$
g~\xfe\xf0\x00\x00\u0794$\xac\xa0\x8d[\xe8^\xbb\x9f12\xdf\xc1\xb6
\x82N\xdf\xed\xf9\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4$\xb2\xbe\x11\x8b\x16\u0632\x17Gi\xd
1{L\xf8O\a\u0294m\x89lk\x93[\x8b\xbd@\x00\x00\u07d4$\xb8\xb4F\u07bd\x19G\x95]\u0404\xf2\x
c5D\x933F\u04ed\x89\xeaim\x90@9\xbd\x80\x00\u07d4$\xb9^\xbe\xf7\x95\x00\xba\xa0\xed\xa7.
w\xf8wA]\xf7\\3\x891T\xc9r\x9d\x05x\x00\x00\u07d4$\xb9\xe6dOk\xa4\xcd\xe1&'\r\x81\xf6\xab`\xf
2\x86\xdf\xf4\x89\a?u\u0460\x85\xba\x00\x00\u07d4$\xbdY\x04\x05\x90\x91\xd2\xf9\xe1-
j&\xa0\x10\xca\"\xab\x14\xe8\x89e\xea=\xb7UF`\x00\x00\u07d4$\xc0\u020bT\xa3TG\t\x82\x8a\xb
4\xab\x06\x84\x05Y\xf6\xc5\u2250\xf54`\x8ar\x88\x00\x00\u07d4$\xc1\x17\xd1\u04b3\xa9z\xb1\x1
aFy\u025awJ\x9e\xad\xe8\u044965\u026d\xc5\u07a0\x00\x00\u07d4$\xcf\xf0\xe93j\x9f\x80\xf9\xb
1\u02d6\x8c\xafk\x1d\x1cI2\xa4\x89\n\xdaUGK\x814\x00\x00\u07d4$\u06aa\xdd\xf7\xb0k\xbc\ua6
c0Y\x00\x85\xa8\x85gh+N\x89\x11K
\x15\u04bb\xd0\x00\x00\u07d4$\xdc\xc2K\xd9\xc7!\f\xea\u03f3\r\xa9\x8a\xe0JM{\x8a\xb9\x8965\u
026d\xc5\u07a0\x00\x00\u07d4$\xf7E\r\xdb\xf1\x8b\x02\x0f\xeb\x1a
2\xd9\xd5Kc>\xdf7\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4$\xfcs\xd2\a\x93\t\x8e\t\u076b
W\x98Pb$\xfa\x1e\x18P\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4$\xfd\x9al\x87L/\xab?\xf3n\x9a\xfb\xf8\xce\r2\xc7\u0792\x89Hz\x9a0E9D\x00\x00\u
07d4%\n@\xce\xf3
#\x97\xf2@F\x95H\xbe\xb5bj\xf4\xf2<\x89\x05\x03\xb2\x03\xe9\xfb\xa2\x00\x00\u07d4%\niC\av\xf
64w\x03\xf9R\x97\x83\x95Za\x97\xb6\x82\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4%\x0e\xb7\xc6o\
x86\x9d\xdfI\u0685\xf39>\x98\f\x02\x9a\xa44\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4%\x10j\xb
6u]\xf8mkc\xa1\x87p;\f\xfe\xa0\u5520\x89\x01|@Z\xd4\x1d\xb4\x00\x00\xe0\x94%\x18_2Z\xcf-
dP\x06\x98\xf6\\v\x9d\xdfh0\x16\x02\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4%\x1c\x12r,hy\"y\x92\xa3\x04\xeb5v\xcd\x18CN\xa5\x89lk\x93[\x8b\xbd@\x00\x00\
u07d4%\x1eh8\xf7\xce\u0173\x83\xc1\xd9\x01F4\x12t\xda\xf8\xe5\x02\x89\a\xff\x1c\xcbua\xdf\x0
0\x00\u07d4%%\x9d\x97Z!\xd8:\xe3\x0e3\xf8\x00\xf5?7\u07e0\x198\x89\x01\x15\x8eF\t\x13\xd0\x
00\x00\u07d4%({\x81_\\\x828\ns\xb0\xb1?\xba\xf9\x82\xbe$\xc4\u04c9\x02+\x1c\x8c\x12'\xa0\x00
\x00\xe0\x94%+eU\xaf\u0700\xf2\xd9m\x97-\x17\u06c4\xeaZ\xd5!\xac\x8a\x01\xab,\xf7\xc9\xf8~
\x00\x00\u07d4%8S)6\x81<\x91\xe6S(O\x01|\x80\u00f8\xf8\xa3o\x89l\x87T\xc8\xf3\f\b\x00\x00\xe
0\x94%>2\xb7N\xa4I\n\xb9&\x06\xfd\xa0\xaa%{\xf2=\u02cb\x8a\x02\x1e\x19\xe0\u027a\xb2@\x0
0\x00\xe0\x94%?\x1et*,\uc1b0\u05f3\x06\xe5\xea\xcbl\xcb/\x85T\x8a\x04>^\xde\x1f\x87\x8c
\x00\x00\u07d4%A1J\v@\x8e\x95\xa6\x94DIwq*Pq5\x91\xab\x89X\x9e\x1a]\xf4\u05f5\x00\x00\u0

7d4%L\x1e\xccc\f(w\u0780\x95\xf0\xa8\u06e1\xe8\xbf\x1fU\f\x89\\(=A\x03\x94\x10\x00\x00\u07d4%Z\xbc\x8d\b\xa0\x96\xa8\x8f=j\xb5_\xbcsR\xbd\u0739\u0389\x04t6\x821>\u0780\x00\u07d4%[\xdddt\u0302b\xf2j\"\u00cfE\x94\x0e\x1c\ue99b\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4%`\xb0\x9b\x89\xa4\xaehI\xedZ<\x99XBf1qDf\x89\\(=A\x03\x94\x10\x00\x00\u07d4%a\xa18\xdc\xf8;\xd8\x13\xe0\xe7\xf1\bd+\xe3\xde=o\x05\x8964\xf4\x84\x17@\x1a\x00\x00\u0794%a\xec\x0f7\x92\x18\xfe^\xd4\xe0(\xa3\xf7D\xaaAuLr\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u0794%b\x92\xa1\x91\xbd\xda4\xc4\xdakk\u0591G\xbfu\u2a6b\x88\xc2\xff.\r\xfb\x03\x80\x00\u07d4%i~\xf2\f\u032ap\xd3-
7o\x82r\xd9\xc1\a\f=x\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4%o\xa1P\u0307\xb5\x05j\a\xd0\x04\xef\xc8E$s\x9eb\xb5\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4%r\x1c\x87\xb0\xdc!7|r\x00\xe5$\xb1J\"\xf0\xafi\xfb\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4%\x899\xbb\xf0\f\x9d\xe9\xafS8\xf5\xd7\x14\xab\xf6\xd0\xc1\xc6q\x89T\x06\x923\xbf\u007fx\x00\x00\xe0\x94%\x90\x12hp\xe0\xbd\xe8\xa6c\xab\x04\nr\xa5W=\x8dA\u008a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4%\x9e\xc4\xd2e\xf3\xabSk|p\xfa\x97\xac\xa1Bi,\x13\xfc\x89\x01\x1b\x1b[\xea\x89\xf8\x00\x00\xe0\x94%\xa5\x00\xee\xeczf*\x84\x15R\xb5\x16\x8bp{\r\xe2\x1e\x9e\x8a\x02\x1f/o\x0f\xc3\xc6\x10\x00\x00\xe0\x94%\xa5\xa4M8\xa2\xf4Lj\x9d\xb9\u037ck\x1e.\x97\xab\xb5\t\x8a\x03\x99\x92d\x8a#\u0220\x00\x00\u07d4%\xa7L*\xc7]\u023a\xa8\xb3\x1a\x9c|\xb4\xb7\x82\x9b$V\u0689lk\x93[\x8b\xbd@\x00\x00\xe0\x94%\xad\xb8\xf9o9I,\x9b\xb4|^\u0708bNF\aV\x97\x8a\x05\xa9\x94\v\xc5hyP\x00\x00\u07d4%\xae\xe6\x8d\t\xaf\xb7\x1d\x88\x17\xf3\xf1\x84\xecV/x\x97\xb74\x89lk\x93[\x8b\xbd@\x00\x00\u07d4%\xb0S;\x81\xd0*a{\x92)\xc7\xec]o/g.[Z\x8965\u026d\xc5\u07a0\x00\x00\u07d4%\xb7\x8c\x9f\xad\x85\xb43C\xf0\xbf\xcd\x0f\xac\x11\u0254\x9c\xa5\xeb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4%\xbcI\xef(\x8c\xd1e\xe5%\xc6a\xa8\x12\u03c4\xfb\xec\x8f3\x89\x12Y!\xae\xbd\xa9\xd0\x00\x00\u07d4%\xbd\xfa>\xe2o8Ia{#\x00bX\x8a\x97\xe3\xca\xe7\x01\x8965\xe6\x19\xbb\x04\xd4\x00\x00\u07d4%\xc1\xa3~\xe5\xf0\x82e\xa1\xe1\r=\x90\xd5G)U\xf9x\x06\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4%\xc6\xe7O\xf1\xd9(\u07d8\x13z\xf4\u07c40\xdf$\xf0|\u05c9\x15$VU\xb1\x02X\x00\x00\xe0\x94%\xcf\xc4\xe2\\5\xc1;i\xf7\xe7}\xbf\xb0\x8b\xafXuk\x8d\x8a\bxg\x83&\xea\xc9\x00\x00\x00\xe0\x94%\xda\u0515\xa1\x1a\x86\xb9\xee\xec\xe1\xee\xec\x80^W\xf1W\xfa\xff\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\xe0\x94%\xe07\xf0\n\x18'\v\xa5\xec4
\"\x9d\xdb\n,\u33e2\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4%\xe6a\xc99\x86:\xcc\x04No\x17\xb5i\x8c\xce7\x9e\xc3\u0309JD\x91\xbdm\xcd(\x00\x00\u07d4&\x04\x8f\xe8M\x9b\x01\nb\xe71b~I\xbc.\xb7?@\x8f\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4&\x06\u00f3\xb4\xca\x1b\t\x14\x98`,\xb1\x97\x8b\xf3\xb9R!\xc0\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4&\n#\x0eDe\a~\v\x14\xeeDB\xa4\x82\u0570\xc9\x14\xbf\x89Z\xf6\x06\xa0k[\x11\x80\x00\u07d4&\r\xf8\x94:\x8c\x9a]\xbayE2\u007f\xd7\xe0\x83|\x11\xad\a\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4&\x14\xf4-
]\xa8D7ux\xe6\xb4H\xdc$0[\xef+\x03\x89lk\x93[\x8b\xbd@\x00\x00\u07d4&\x15\x10\x0e\xa7\xe2[\xba\x9b\xcat`X\xaf\xbb\xb4\xff\xbeBD\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4&\x15u\xe9\xcfY\xc8\"o\xa7\xaa\xf9\x1d\xe8o\xb7\x0fZ\u00ee\x89\x10C\xa4CjR?\x00\x00\xe0\x94&\x1e\x0f\xa6LQ\x13te\xee\xcf[\x90\xf1\x97\xf7\x93\u007f\xdb\x05\x8a\x03\xcf\xc8.7\xe9\xa7@\x00\x00\u07d4&*\x8b\xfd}\x9d\xc5\xdd:\u05c1a\xb6\xbbV\b$76U\x89?j\x83\x84\a+v\x00\x00\xe0\x94&*\xedK\xc0\xf4\xa4\xb2\xc6\xfb5y>\x83ZI\x18\x9c\xdf\xec\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94&-\xc16L\xcfm\xf8\\C&\x8e\xe1\x82UM\xaei.)\x8a\x01\v
/\xect\xce\xd8\x00\x00\u07d4&8\x140\x9d\xe4\xe65\xcfX^\r6Tw\xfc@\xe6l\xf7\x89\a\xea(2uw\b\x0

0\x00\u07d4&9\xee\xe9\x87<\xee\xc2o\u0314T\xb5H\xb9\xe7\xc5J\xa6\\\x8965\u026d\xc5\u07a0\
x00\x00\xe0\x94&>W\xda\xcb\xe0\x14\x9f\x82\xfee\xa2fH\x98\x86o\xf5\xb4c\x8a\b\v\xfb\xef\xcb_\
v\xc0\x00\x00\u07d4&GT\x19\xc0m_\x14z\xa5\x97$\x8e\xb4l\xf7\xbe\xfad\xa5\x89X\xe7\x92n\xe8
X\xa0\x00\x00\u07d4&L\xc8\bj\x87\x10\xf9\x1b!r\t\x05\x91,\u05d6J\xe8h\x89\x01s\x17\x90SM\xf2\
x00\x00\xe0\x94&S\x83\u058bR\xd04\x16\x1b\xfa\xb0\x1a\xe1\xb0G\x94/\xbc2\x8a\x04rq\xde\xe2
\rt\\\x00\x00\u07d4&Y\xfa\xcb\x1e\x83CeS\xb5\xb4)\x89\xad\xb8\a_\x99S\xed\x89\x01\x97evw\x1
a^\x00\x00\xe0\x94&o-\xa7\xf0\b^\xf3\xf3\xfa\t\xba\xee#+\x93\xc7D\xdb.\x8a\f\xb4\x9bD\xba`-
\x80\x00\x00\u07d4&qH\xfdr\xc5Ob\nY/\xb9'\x991\x9c\xc4S+\\\x89\x169\u46fa\x16(\x00\x00\xe0\x
94&xJ\u0791\u0228:\x8e9e\x8c\x8d\x82wA<\u0319T\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4
&z~n\x82\xe1\xb9\x1dQ\xde\u0776D\xf0\xe9m\xbb\x1f\u007f~\x89\x01\x15\x8eF\t\x13\xd0\x00\x0
0\u07d4&\x80q=@\x80\x8e*P\xed\x011P\xa2\xa6\x94\xb9j\u007f\x1d\x89a\t=|,m8\x00\x00\u07d4
&\x97\xb39\x81;\f-
\x96K$q\xeb\x1c`oN\u02d6\x16\x89>\x8e\xf7\x95\u0610\xc8\x00\x00\u07d4&\xa6\x8e\xab\x90Z\x
8b=\xce\x00\xe3\x170\x82%\u06b1\xb9\xf6\xb8\x89kV\x05\x15\x82\xa9p\x00\x00\u07d4&\xb1\x1d
\x06e\x88\xcet\xa5r\xa8Zc(s\x92\x12\xaa\x8b@\x89lk\x93[\x8b\xbd@\x00\x00\u07d4&\xba\xbfB\x
b2g\xfd\xcf8a\xfd\xd4#j^GHH\xb3X\x8965\u026d\xc5\u07a0\x00\x00\u07d4&\xc0\x05Kp\r:|-
\xcb\xe2uh\x9dOL\xad\x16\xa35\x89lk\x93[\x8b\xbd@\x00\x00\u07d4&\xc2\xff\xc3\x0e\xfd\xc5'>v\
x18:\x16\xc2i\x8dnS\x12\x86\x89*\x11)\u0413g
\x00\x00\u07d4&\u025f\x88I\u0240+\x83\xc8a!\u007f\xd0z\x9e\x84\u0377\x9d\x89\x10CV\x1a\x88
)0\x00\x00\u07d4&\xcf\xff\xd0R\x15+\xb3\xf9W\xb4x\xd5\xf9\x8b#:|+\x92\x89\xd8\xd7&\xb7\x17z\x
80\x00\x00\u07d4&\u0521h\x91\xf5)\"x\x92\x17\xfc\u0606\xf7\xfc\xe2\x96\xd4\x00\x89lk\x93[\x8b\
xbd@\x00\x00\u07d4&\xd4\xec\x17\xd5\u03b2\u0214\xbd\u015d\nji]\xad+C\u0309\x9f\x1fxv\x1d4\
x1a\x00\x00\u07d4&\xe8\x01\xb6,\x82q\x91\xddh\xd3\x1a\x01\x19\x90\x94\u007f\xd0\xeb\xe0\x89
\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4&\xe9\xe2\xadr\x97\x02bd\x17\xef%\xde\r\xc8\x00\xf7\xa7
y\xb3\x8965\u026d\xc5\u07a0\x00\x00\u07d4&\xf9\xf7\xce\xfd~9K\x9d9$A+\xf2\u0083\x1f\xaf\x1f\
x85\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94&\xfe\x17L\xbfRfP\xe0\xcd\x00\x9b\xd6\x12e\x02
\u038ehM\x8a\x02w\x01s8\xa3\n\xe0\x00\x00\xe0\x94&\xff\nQ\xe7\xce\u0384\x00'ix\xdb\xd6#n\xf
1b\xc0\xe6\x8a\x15.\x18V'T\nP\x00\x00\u07d4'\x10\x1a\x0fV\u04da\x88\u0168O\x9b2L\xdd\xe3>\\
\xb6\x8c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4'\x14L\xa9\xa7w\x1a\x83j\xd5\x0f\x80?d\xd8i\xb2\xa
e+
\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4'\x14i\x13V:\xa7E\xe2X\x840\xd94\x8e\x86\xea|5\x10\
x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4'\x1d=H\x1c\xb8\x8evq\xad!iI\xb66^\x060=\xe0\x89\xd8
\xd7&\xb7\x17z\x80\x00\x00\u07d4'
\xf9\xcaBn\xf2\xf2\xcb\xd2\xfe\xcd9\x92\fO\x1a\x89\xe1m\x89lk\x93[\x8b\xbd@\x00\x00\u07d4'*\x
13\x1aZejz:\xca5\u023d
\"\"\xa7Y\"X\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4'D\xffgFA!\xe3Z\xfc)\"\x17qd\xfa/\xcb\x02g\x89\x
05k\xc7^-
c\x10\x00\x00\u07d4'J=w\x1a=p\x97\x96\xfb\xc4\xd5\xf4\x8f\xce/\xe3\x8cy\u0589\x01\x15\x8eF\t\
x13\xd0\x00\x00\u07d4'Mi\x17\x0f\xe7\x14\x14\x01\x88+\x88j\xc4a\x8cj\xe4\x0e\u06c93\xc5I\x901
r\f\x00\x00\u07d4'R\x1d\xeb;n\xf1An\xa4\u01c1\xa2\xe5\u05f3n\xe8\x1ca\x89lk\x93[\x8b\xbd@\x0
0\x00\u07d4'Xu\xffO\xbb\f\xf3\xa40!1'H\u007fv\b\xd0L\xba\x89\x1b\x1c\x01\x0evmX\x00\x00\u07d
4'j\x00n0(\xec\xd4L\xdbb\xba\nw\u0394\xeb\xd9\xf1\x0f\x89a\x94\x04\x9f0\xf7

\x00\x00\u07d4'k\x05!\xb0\xe6\x8b'}\xf0\xbb2\xf3\xfdH2cP\xbf\xb2\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4'o\xd7\xd2O\x8f\x88?Zz()[\xf1qQ\u01e8K\x03\x89lk\x93[\x8b\xbd@\x00\x00\u07d4'p\xf1N\xfb\x16]\u07bay\xc1\v\xb0\xaf1\xc3\x1eY3L\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4'vw\xab\xa1\xe5,;S\xbf\xa2\a\x1dN\x85\x9a\n\xf7\xe8\xe1\x8965\u026d\xc5\u07a0\x00\x00\u07d4'\x82Ff\xd2x\xd7\x04#\xf0=\xfe\x1d\u01e3\xf0/C\u2d4966\xc2^f\xec\xe7\x00\x00\u07d4'\x83\f_`#\xaf\xaa\xf7\x97Egl
J\x0f\xac\u0360\xba\x89\r\x02\xabHl\xed\xc0\x00\x00\xe0\x94'\x84\x90?\x1d|\x1b\\\xd9\x01\xf8\x87]\x14\xa7\x9b<\xbe*V\x8a\x04\xbd\xa7\xe9\xd7J\xd5P\x00\x00\u07d4'\x8c\v\xdec\x0e\u00d3\xb1\xe7&\u007f\xc9\xd7\xd9p\x19\xe4\x14[\x89lk\x93[\x8b\xbd@\x00\x00\u07d4'\x98q\x10\"\x1a\x88\b&\xad\xb2\xe7\xab^\xcax\xc6\xe3\x1a\xec\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94'\xac\a;\xe7\x9c\xe6W\xa9:\xa6\x93\xeeC\xbf\x0f\xa4\x1f\xef\x04\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4'\xb1iN\xaf\xa1e\xeb\xd7\xcc{\u025et\x81J\x95\x14\x19\u0709+^:\xf1k\x18\x80\x00\x00\u07d4'\xb6(\x16\xe1\xe3\xb8\u045by\xd1Q=]\xfa\x85[\f:*\x89\x05j\xf5\xc1\xfdiP\x80\x00\u07d4'\xbf\x94<\x163\xfe2\xf8\xbc\xcc\xdbc\x02\xb4\a\xa5rND\x892\xf8Lm\xf4\b\xc0\x80\x00\u07d4'\xbf\x9fD\xba}\x05\xc35@\u00e5;\xb0,\xbb\xff\xe7\xc3\u0189lk\x93[\x8b\xbd@\x00\x00\u07d4'\xc2\xd7\xcaPM\xaa=\x90f\xdc\t\x13}\xc4/:\xaa\xb4R\x89
\x86\xac5\x10R`\x00\x00\u07d4'\xd1X\xac=>\x11\t\xabnW\x0e\x90\xe8]8\x92\xcdv\x80\x89\x05k\xc7^-c\x10\x00\x00\u07d4'\xe69\x89\xca\x1e\x90;\xc6
\xcf\x1b\x9c?g\xb9\xe2\xaee\x81\x89Hz\x9a0E9D\x00\x00\xe0\x94'\xf0<\xf1\xab\xc5\xe1\xb5\x1d\xbcDK(\x9eT,\x9d\u07f0\xe6\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4'\xfc\x85\xa4\x9c\xff\x90\xdb\xcf\xda\u071d\xdd@\u05b9\xa2!\nl\x89\x05k\xc7^-c\x10\x00\x00\u07d4(\x05A^\x1d\u007f\xde\xc6\xde\u07f8\x9eR\x1d\x10Y-t<\x10\x89\x05k\xc7^-c\x10\x00\x00\u07d4(\a>\xfc\x17\xd0\\\xab1\x95\xc2\xdb3+a\x98Gw\xa6\x12\x8965\u026d\xc5\u07a0\x00\x00\u07d4(\x12P\xa2\x91!'\nN\xe5\u05cd$\xfe\xaf\xe8,p\xba:\x8965\u026d\xc5\u07a0\x00\x00\u07d4(\x13\xd2c\xfc_\xf2G\x9e\x97\x05\x95\u05b6\xb5`\xf8\xd6\xd6\u0449lk\x93[\x8b\xbd@\x00\x00\u07d4(.\x80\xa5T\x87ZVy\x9f\xa0\xa9\u007fU\x10\u7557LN\x8965\u026d\xc5\u07a0\x00\x00\u07d4(3\x96\xce<\xac9\x8b\xcb\xe7\"\u007f2>x\xff\x96\u0407g\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4(4\x9f~\xf9t\xeaU\xfe6\xa1X;4\xce\xc3\xc4Pe\xf0\x89\f\xb63\u051eeY\x00\x00\u07d4(6\x120F\xb2\x84\xe5\xef\x10+\xfd\"\xb1v^P\x81\x16\xad\x89\x16S\xfb\xb5\xc4'\xe4\x00\x00\u07d4(<#\x14(<\x92\u0530d\xf0\xae\xf9\xbbRF\xa7\x00\u007f9\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4(>\x11
7I\xb1\xfaO2\xfe\xbbq\xe4\x9d\x13Y\x198*\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94(>bR\xb4\xef\xcfFT9\x1a\xcbu\xf9\x03\u015bx\xc5\xfb\x8a\x02\x8a\x85t%Fo\x80\x00\x00\xe0\x94(Q\x0en\xff\x1f\xc8)\xb6WoC(\xbc98\xecze\x80\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4(X\xac\xac\xaf!\xea\x81\u02b7Y\x8f\xdb\xd8kE.\x9e\x8e\x15\x89$\x1a\x9bOaz(\x00\x00\u07d4(Z\xe5\x1b\x95\x00\u014dT\x13e\xd9ui\xf1K\xb2\xa3p\x9b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4(f\xb8\x1d\xec\xb0.\xe7\n\xe2P\xce\xe5\xcd\xc7{Y\u05f6y\x89lk\x93[\x8b\xbd@\x00\x00\u07d4(i\x06\xb6\xbdIr\xe3\xc7\x16U\xe0K\xaf6&\f|\xb1S\x89\x12nr\xa6\x9aP\xd0\x00\x00\u07d4(k\x18ma\xea\x1f\u05cd\x990\xfe\x12\xb0e7\xb0\\=Q\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94(t\xf3\xe2\x98]_{@f'\xe1{\xaaw+\x01\xab\u031e\x8a\x01F\x05\x04\x10v_8\x00\x00\xe0\x94(|\xf9\u0410.\xf8\x19\xa7\xa5\xf1ID[\xf1w^\xe8\xc4|\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07d4(\x81\x8e\x18\xb6\x10\x00\x13!\xb3\x1d\xf6\xfe}(\x15\u036d\xc9\xf5\x8965\u026d\xc5\u07a0\x00\x00\u07d4(\x86\x83$3~\x11\xba\x10l\xb4\x81\

u0696/:\x84S\x80\x8d\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94(\x90K\xb7\xc40)C\xb7\t\xb1Myp\xe4
+\x83$\u184a\x02\x1f\x97\x84j\a-
~\x00\x00\u07d4(\x95\xe8\t\x99\xd4\x06\xadY.+&'7\xd3_}\xb4\xb6\x99\x89i*\xe8\x89p\x81\xd0\x00
\x00\u07d4(\x96r\x80!N!\x8a\x12\f]\xda7\x04\x1b\x11\x1e\xa3mt\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4(\xa3\xda\t\xa8\x19H\x19\xae\x19\x9f.m\x9d\x13\x04\x81~(\xa5\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4(\xab\x16_\xfbi\xed\xa0\xc5I\xae8\xe9\x82o_\u007f\x92\xf8S\x89FM\xf6\xd7\xc8
DY\x00\x00\u07d4(\xb7u\x85\xcb=U\xa1\x99\xab)\x1d:\x18\u018f\u8684\x8a\x89j@v\xcfy\x95\xa0\
x00\x00\xe0\x94(\xd4\xeb\xf4\x1e=<E\x1e\x94;\xdd~\x1f\x17_\xae\x93*=\x8a\x01EB\xba\x12\xa37
\xc0\x00\x00\xe0\x94(\xd7\xe5\x86o\x1d\x85\xfd\x1c\xeb2\xbf\xbe\x1d\xfc6\xdbCEf\x8a\x01\x86B1
\xc6\x105\x1c\x00\x00\u0794(\xd8\xc3_\xb7\xee\xa6\"X!5\xe3\xadG\xa2'\u0266c\xbd\x88\xfc\x93c\
x92\x80\x1c\x00\x00\u07d4(\xe4\xaf0\u0353\xf6\x86\xa1\"\xad{\xb1\x9f\x8a\x87\x85\xee\xe3B\x89
q\xe5;pl\u01f4\x00\x00\u07d4(\xea\xeax\xcdM\x95\xfa\xec\xfbh\x83n\xaf\xe85
\U000fbdc9\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94(\xef\xaecVP\x9e\u07ec\xe8\x9f\xc6\x1a\u007f\xdc\xdb9\xee\xa8\xe5\x8a\x01!\xe
ah\xc1\x14\xe5\x10\x00\x00\xe0\x94(\xfa%\x80\xf9\xeb\xe4
\xf3\xe5\xee\xfd\xd3qc\x8e;z\xf4\x99\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4)\x01\xf8\a\u007f
4\x19\v\xb4z\x8e\"\u007f\xa2\x9b0\xce\x11;1\x89\x05k\xc7^-
c\x10\x00\x00\u07d4)\x05\xb1\x92\xe8<\xe6Y\xaa5[\x9d\f
N>\x95\xf9\xbb\x9a\x89u#\\\x1d\x009>\x80\x00\u07d4)\nV\xd4\x1fn\x9e\xfb\xdc\xea\x03B\u0dd2\x
9a\x8c\xdf\xcb\x05\x89\x12\xa5\xf5\x81h\xee`\x00\x00\u07d4)\x15bK\xcbg\x917\xb8\xda\xe9\xab
W\xd1\x1bI\x05\xea\xeeK\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4)\x1e\xfe\x00\x81\xdc\xe8\xc
1G\x99\xf7\xb2\xa46\x19\xc0\u00f3\xfc\x1f\x89A\rXj
\xa4\xc0\x00\x00\u07d4)\x1f\x92\x9c\xa5\x9bT\xf8D>=Mu\xd9]\xee$<\xefx\x89\x1b\x1a\b\x927\a=\
x00\x00\xe0\x94))\x8c\xcb\xdf\xf6\x89\xf8\u007f\xe4\x1a\xa6\xe9\x8f\u07f5=\xea\xf3z\x8a\x041\\2\
xd7\x1a\x9e`\x00\x00\u07d4)/\"\x8b\n\x94t\x8c\x8e\xeca-
$o\x98\x93c\xe0\x8f\b\x89\n\ad\a\xd3\xf7D\x00\x00\u07d4)3\x84\xc4+o\x8f)\x05\xceR\xb7
\\\"t7la+\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4)4\xc0\xdf{\xbc\x17+l\x18k\vrTz\u038b\xf7TT\x89\x03
@\xaa\xd2\x1b;p\x00\x00\u07d4)<#\x06\xdf6\x04\xaeO\xda\r
z\xbasog\xde\a\x92\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94)I\xfd\x1d\xef\\v\xa2\x86\xb3\x87$$\x80\x9a\a\xdb9f\xf3\x8a\x01\x1b\xd9\x06\u06
a0\xc9C\x80\x00\u07d4)OIK?.\x14</\xfc\x978\xcb\xfd\x95\x01\x85\v\x87N\x89yn>\xa3\xf8\xab\x00
\x00\x00\u07d4)U\xc3W\xfd\x8fu\xd5\x15\x9a=\xfai\u0178z5\x9d\ua309lk\x93[\x8b\xbd@\x00\x00\
u07d4)a\xfb9\x1ca\x95|\xb5\xc9\xe4\a\u0762\x938\u04f9,\x80\x8964\xfb\x9f\x14\x89\xa7\x00\x00\
u07d4)h\x1d\x99\x12\xdd\xd0~\xaa\xbb\x88\xd0]\x90\xf7f\xe8bA}\x8965\u026d\xc5\u07a0\x00\x00
\u07d4)kq\xc0\x01X\x19\xc2B\xa7\x86\x1eo\xf7\xed\xed\x8a_q\xe3\x89lh\xcc\u041b\x02,\x00\x00\
u07d4)mf\xb5!W\x1aNA\x03\xa7\xf5b\xc5\x11\xe6\xaas-
\x81\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4)o\x00\xde\x1d\u00fb\x01\xd4z\x8c\xcd\x1e]\x1d\u0661\
xebw\x91\x8965\u026d\xc5\u07a0\x00\x00\u07d4)s\x85\xe8\x864FV\x85\xc21\xa3\x14\xa0\xd5\xd
c\xd1F\xaf\x01\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4)v=\xd6\u069a|\x16\x11s\x88\x83!\ub9b6
<\x8f\xb8E\x89\x11\xc7\xea\x16.x
\x00\x00\u07d4)yt\x11t\xa8\xc1\xea\v\u007f\x9e\xdfe\x81w\x85\x94\x17\xf5\x12\x89\x19\x01\x96l\
x84\x96\x83\x80\x00\u07d4)z\x88\x92\x1b_\xca\x10\u5edd\xed`\x02T7\xae\"\x16\x94\x89\n\u05ce

\xbcZ\xc6
\x00\x00\xe0\x94)}]\xbe\"//\xb5%1\xac\xbd\v\x01=\xc4F\xacsh\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4)\x82N\x94\xccCH\xbc\x962y\xdc\xdfG9\x17\x152L\u04c9i*\xe8\x89p\x81\xd0\x00\x00\u07d4)\x82\xd7j\x15\xf8G\xddA\xf1\x92*\xf3h\xfeg\x8d\x0eh\x1e\x89\x05k\xc7^-
c\x10\x00\x00\u07d4)\x88\x87\xba\xb5|[\xa4\xf0aR)\xd7R_\xa1\x13\xb7\ua249\x02+\x1c\x8c\x12'\xa0\x00\x00\xe0\x94)\x8e\xc7kD\r\x88\a\xb3\xf7\x8b_\x90\x97\x9b\xeeB\xedC\u06ca\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4)\x93h`\x90B\xa8X\xd1\xec\xdf\x1f\xc0\xad\xa5\xea\xce\xca)\u03c9lk\x93[\x8b\xbd@\x00\x00\u07d4)\x9e\v\xcaU\xe0i\u0785\x04\xe8\x9a\xcan\xca!\u04ca\x9a]\x89\x03\x027\x9b\xf2\xca.\x00\x00\u07d4)\xac+E\x84T\xa3l~\x96\xc7:\x86g\"*\x12$,q\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94)\xad\u03c3\xb6\xb2\n\u01a44\xab\xb1\x99<\xbd\x05\xc6\x0e\xa2\xe4\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94)\xae\xf4\x8d\xe8\xc9\xfb\xadK\x9eL\xa9pyzU3\xebr-
\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4)\xb3\xf5a\xeezn%\x94\x1e\x98\xa52[x\xad\u01d7\x85\xf3\x89\x05k\xc7^-
c\x10\x00\x00\u07d4)\xbd\xc4\xf2\x8d\xe0\x18\x0fC<&\x94\xebt\xf5PL\xe9C7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4)\u0300M\x92+\xe9\x1fY\t\xf3H\xb0\xaa\xa5\xd2\x1b`x0\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94)\xda>5\xb2;\xb1\xf7/\x8e\"X\xcf\u007fU3Y\xd2K\xac\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94)\xe6y\x90\xe1\xb6\xd5.\x10U\xff\xe0I\xc51\x95\xa8\x15B\u03ca\x04<3\xc1\x93ud\x80\x00\x00\u07d4)\uab82v\x17b\xf4\xd2\xdbS\xa9\u018b\x0fk\vmNf\x89lk\x93[\x8b\xbd@\x00\x00\u07d4)\xeb~\xef\xda\xe9\xfe\xb4I\xc6?\xf5\xf2y\xd6u\x10\xeb\x14\"\x89\x01\r:\xa56\xe2\x94\x00\x00\u07d4)\xf0\xed\xc6\x038\xe7\x11
\x85\xa1\xd1\x14\u068cB\u038fU\u0589\xa0Z\u007f\x0f\xd8%x\x00\x00\u07d4)\xf8\xfb\xa4\xc3\ar\xb0W\xed\xbb\xe6*\xe7B\f9\x05r\xe1\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94)\xf9(I\x0es\x8d\x17!\xa6\x91\u01b9Z\xb3\u0667\x97\xed\xe8\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4*\b^%\xb6Hb\xf5\xe6\x8dv\x8e+\x0fz\x85)\x85\x8e\xee\x89k\x88:\xcdWf\xcd\x00\x00\u07d4**\xb6\xb7Lz\xf1\xd9Gk\xb5\xbc\xb4RG\x97\xbe\xdc5R\x8965\u026d\xc5\u07a0\x00\x00\u07d4*9\x19\nO\u0783\u07f3\xdd\xcbL_\xbb\x83\xacIIu\\\x8965\u026d\xc5\u07a0\x00\x00\u07d4*@\r\xff\x85\x94\xder(\xb4\xfd\x15\xc3#\"\xb7[\xb8}\xa8\x89\x051\xa1\u007f`z-
\x00\x00\xe0\x94*D\xa7!\x8f\xe4Me\xa1\xb4\xb7\xa7\u0671\xc2\xc5,\x8c>4\x8a\r-
\x06\xc3\x05\xa1\xebW\x80\x00\u07d4*F\xd3Swqv\xff\x8e\x83\xff\xa8\x00\x1fOp\xf9s:\xa5\x89\x05\xbf\v\xa6cOh\x00\x00\u07d4*Y_\x16\xee\xe4\xcb\f\x17\u0662\xd99\xb3\xc1\x0flgrC\x89;\xa1\x91\v\xf3A\xb0\x00\x00\u07d4*Y\xe4~\xa5\xd8\xf0\xe7\xc0(\xa3\xe8\xe0\x93\xa4\x9c\x1bP\xb9\xa3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4*[\xa9\xe3L\u054d\xa5L\x9a'\x12f:;\xe2t\xc8\xe4{\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94*^:@\xd2\xcd\x03%vm\xe7:=g\x18\x96\xb3b\xc7;\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94*cY\x0e\xfe\x99\x86\xc3\xfe\xe0\x9b\n\n3\x8b\x15\xbe\xd9\x1f!\x8a\x01^\x1cN\x05\xee&\xd0\x00\x00\u07d4*gf\n\x13h\xef\xcdbn\xf3k+\x1b`\x19\x80\x94\x1c\x05\x89\a?u\u0460\x85\xba\x00\x00\u07d4*t+\x89\x10\x94\x1e\t2\x83\n\x1d\x96\x92\xcf\u0484\x94\xcf@\x89\x1b\x1a\xb3\x19\xf5\xecu\x00\x00\u07d4*tl\xd4@'\xaf>\xbd7\xc3x\xc8^\xf7\xf7T\xab_(\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4*\x81\xd2|\xb6\xd4w\x0f\xf4\xf3\u0123\xba\x18\xe5\xe5\u007f\aQ|\x89lk\x93[\x8b\xbd@\x00\x00\u07d4*\x91\xa9\xfe\xd4\x1b}\x0e\\\xd2\xd81X\xd3\xe8\xa4\x1a\x9a-
q\x89i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94*\x9cW\xfe{k\x13\x8a\x92\rgo<v\xb6\u00a0\xee\xf6\x99\

x8a\x01\xfd\x934\x94\xaa_\xe0\x00\x00\u07d4*\x9c\x96\xc1\x91Q\xff\xcb\xe2\x9aF\x16\xd0\xc5+9
3\xb4e\x9f\x89\x03\xc17\x9b\x87e\xe1\x80\x00\u07d4*\xa1\x92w|\xa5\xb9x\xb6\xb2\xc2\xff\x80\n\
xc1\x86\x0fu?G\x89\x12)\x0f\x15\x18\v\xdc\x00\x00\xe0\x94*\xaa5'Mt%Fg\vt&&E!\x03*\xf4\xf4\u0
0ca\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4*\xae\xa1\xf1\x04o0\xf1\t\xfa\xec\x1cc\xef\\u\x94
\xeb\b\u0689\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4*\xb9~\x8dY\xee\xe6H\xabl\xaf\x86\x96\xf8\x
997\x148d\u0589\xcf\x15&@\xc5\xc80\x00\x00\xe0\x94*\xbc\u1009@\xcdN\xf5\xb5\xe0R\x85\xf8-
\xf7\xa9\xab^\x03\x8a\x02\x13BR\r_\xec
\x00\x00\xe0\x94*\xbd\xf1\xa67\xeflB\xa7\xe2\xfe!ws\xd6w\xe8\x04\xeb\u074a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4*\xc1\xf8\u05ffr\x1f<\xfet\xd2\x0f\ua6c7\xa2\x8a\xaa\x98,\x89\b\xbaR\xe6\xfcE\xe4\
x00\x00\xe0\x94*\u031c\x1a2$\vM[/wz.\xa0R\xb4/\xc1'\x1c\x8a\b\xd8\a\xee\x14\xd86\x10\x00\x00
\u07d4*\xd6\xc9\xd1\f&\x18\x19\xa1\xa0\xca,H\xd8\u01f2\xa7\x17(\u07c965\u026d\xc5\u07a0\x00
\x00\xe0\x94*\xe58f\xfc-
\x14\xd5r\xabs\xb4\xa0e\xa1\x18\x82g\xf5'\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4*\xe7:y\xae
\xa0'\x853\xac\xcf!\a\t\"\xb1a?\x8f2\x89\xa7\xe9K\xbe\xaep\x1a\x80\x00\u07d4*\xe8-
\xab\x92\xa6c\x89\ue86b\xb9\x01\xd1\xd5\u007fZ|\xca\v\x89lk\x93[\x8b\xbd@\x00\x00\u07d4*\uc
01d\xf92[\x9fH9\x96\xe9\x9fs1\t\u007f\b\xaa\x0e\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94*\xe
d,\xe51\xc0V\xb0\t~\xfc<m\xe1\fGb\x00N\u064a\x025iS\xab}\xdc8\x00\x00\u07d4*\xfb\x05\x8c=1\
x03+5;\xf2O\t\xae
\xd5M\xe5}\xbe\x89;\xa1\x91\v\xf3A\xb0\x00\x00\xe0\x94+\x03bc6\x14\xbf\xcbX5iC\x8e\xccN\xa5{
\x1d3~\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4+\x10\x1e\x82,\xd9b\x96*\x06\x80\n,\b\u04f1]\x82
\xb75\x89\b=lz\xabc`\x00\x00\u07d4+\x12\x9c&\xb7]\xde\x12\u007f\x83
\xbd\x0fcA\f\x92\xa9\xf8v\x89wC\"\x17\xe6\x83`\x00\x00\xe0\x94+$\x1f\x03s7\xebJ\xcca\x84\x9b\
xd2r\xac\x13?|\xdfK\x8aP\vk\u0296*\xb8@\x00\x00\u07d4+:h\xdbk\f\xae\x8a|zGk\xdf\u03fdb\x05\
xe1\x06\x87\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4+<\xf9s\x11\xff0\xf4`\x94Z\x9d\x80\x99\xf4\
xa8\x8e&\xd4V\x89lk\x93[\x8b\xbd@\x00\x00\u07d4+I\xfb\xa2\x9806\x0f\u0376\xda#\xbb\xfe\xa5\
xc0\xbb\xacR\x81\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u0794+OE\a\xbbk\x98\x17\x94,\xe43x\x1b
p\x8f\xbc\xd1f\xfd\x88\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94+P\x16\xe2Es\x87\x95ebXq\x15\xaa\
x87Y\xd8i_\u07ca*Z\x05\x8f\u0095\xed\x00\x00\x00\xe0\x94+\\`\xe8E5\xee\xb4\u0540\xde\x12z\x
12\xeb&w\u0333\x92\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4+\\\ud647\xc0v_\x90\x0eI\u03dd\xa
2\xd9\xf9\xc1\x13\x88U\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94+_K?\x1e\x11pz\"z\xa5\u67e
4\x9d\xde\xd3?\xb3!\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4+h0k\xa7\xf8\u06afs\xf4\xc6D\xe
f}'C\xc0\xf2hV\x89.\xe1\x82\xca\x17\xdd\xd0\x00\x00\xe0\x94+n\u049a\x95u<:\xd9H4\x8e>{\x1a%
\x10\x80\xff\xb9\x8a4\xf0\x86\xf3\xb3;h@\x00\x00\u07d4+p\x1d\x16\xc0\xd3\xcc\x1eL\xd8TE\xe6\
xad\x02\ue92c\x01-\x89
\x86\xac5\x10R`\x00\x00\xe0\x94+q|\xd42\xa3#\xa4e\x909\x84\x8d;\x87\xde&\xfc\x95F\x8ai\xe1\r\
xe7fv\u0400\x00\x00\u07d4+t\xc3s\xd0K\xfb\x0f\xd6\n\x18\xa0\x1a\x88\xfb\xe8Gp\u5309\x02+\x1
c\x8c\x12'\xa0\x00\x00\u07d4+w\xa4\u060c\rV\xa3\xdb\xe3\xba\xe0J\x05\xf4\xfc\u0477W\xe1\x89
\x10CV\x1a\x88)0\x00\x00\xe0\x94+\x84\x88\xbd-
<\x19z=&\x15\x18\x15\xb5\xa7\x98\xd2qh\u070a\x01j\x1f\x9f_\xd7\xd9`\x00\x00\u07d4+\x8a\r\xee
\\\xb0\xe1\xe9~\x15\xcf\xcan\x19\xad!\xf9\x95\ufb49\x1bUC\x8d\x9a$\x9b\x00\x00\xe0\x94+\x8f\x
e4\x16n#\xd1\x19c\xc0\x93+\x8a\u078e\x01E\xea\ap\x8a\t(\x96R\x9b\xad\u0708\x00\x00\xe0\x94
+\x99\xb4.OBa\x9e\xe3k\xaa~J\xf2\xd6^\xac\xfc\xba5\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4+\

xab\x0f\xbe(\u0544 \xb5
6w\n\x12\xf9\x95*\xeai\x11\x89\xcf\x15&@\xc5\xc80\x00\x00\u07d4+\xad\xe9\x1d\x15E\x17b\x0f\u05349\xac\x97\x15zA\x02\xa9\xf7\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4+\xaf\x8dn\"\x11t\x12H
\xeeI+\x94Y\xecO\xad\xaf\xbb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4+\xaf\xbf\x9e\x9e\xd2\xc2\x19\xf7\xf2y\x13t\xe7\xd0\\\xb0gw\xe7\x89\v\xed\x1d\x02c\xd9\xf0\x00\x00\xe0\x94+\xb3f\xb9\xed\xcb\r\xa6\x80\xf0\xe1\v;n(t\x81\x90\xd6\u00ca\x01:b\u05f5v@d\x00\x00\xe0\x94+\xb6\xf5x\xad\xfb\u7ca1\x16\xb3UO\xac\xf9\x96\x98\x13\xc3\x19\x8a\x01\x91'\xa19\x1e\xa2\xa0\x00\x00\u07d4+\xbeb\xea\xc8\f\xa7\xf4\xd6\xfd\xee~}\x8e(\xb6:\xcfw\x0e\x89\x81\xe3-
\xf9r\xab\xf0\x00\x00\u07d4+\xbeg*\x18WP\x8fc\x0f*^\xdbV=\x9e\x9d\xe9(\x15\x89lk\x93[\x8b\xbd@\x00\x00\u07d4+\xc4)\xd6\x18\xa6jL\xf8-\xbb-
\x82N\x93V\xef\xfa\x12j\x89lj\xccg\u05f1\xd4\x00\x00\u07d4+\xd2R\xe0\xd72\xff\x1d|x\xf0\xa0.l\xb2T#\xcf\x1b\x1a\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4+\xdd\x03\xbe\xbb\xee';l\xa1\x05\x9b4\x99\x9a[\xbda\xbby\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4,\x04\x11\\>R\x96\x1b\r\xc0\xb0\xbf1\xfb\xa4ToYf\xfd\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94,\x06\u0752+aQJ\xaf\xed\xd8D\x88\xc0\u008em\xcf\x0e\x99\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94,\f\xc3\xf9QH,\u0222\x92X\x15hN\xb9\xf9N\x06\x02\x00\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4,\x0e\xe14\u0633aE\xb4{\xee\u7bcd'8\xdb\xda\b\xe8\x89\n\xe5os\x0em\x84\x00\x00\u07d4,\x0f[\x9d\xf46%y\x8e~\x03\xc1\xa5\xfdjm\t\x1a\xf8+\x89\x01\xb0\xfc\xaa\xb2\x000\x00\x00\u07d4,\x12\x8c\x95\xd9W!Q\x01\xf0C\u074f\u0142EmA\x01m\x89-
C\xf3\xeb\xfa\xfb,\x00\x00\u07d4,\x18\x00\xf3_\xa0-
>\xb6\xff[%(_^J\xdd\x13\xb3\x8d\x891\"\u04ed\xaf\xde\x10\x00\x00\u07d4,\x1c\x19\x11N=m\xe2xQHK\x8d'\x15\xe5\x0f\x8a\x10e\x89\x05k\xc7^-
c\x10\x00\x00\u07d4,\x1c\xc6\xe1\x8c\x15$\x88\xba\x11\xc2\xcc\x1b\xce\xfa-
\xf3\x06\xab\u0449Z\x87\xe7\xd7\xf5\xf6X\x00\x00\xe0\x94,\x1d\xf8\xa7oH\xf6\xb5K\u03dc\xafV\xf0\xee\x1c\xf5z\xb3=\x8a\x02$\u007fu\x00\x89\xdaX\x00\x00\u07d4,!G\x94z\xe3?\xb0\x98\xb4\x89\xa5\xc1k\xff\xf9\xab\xcdN*\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4,#OP\\\xa8\xdc\xc7}\x9b~\x01\xd2W\xc3\x18\xcc\x199m\x89\x05k\xc7^-
c\x10\x00\x00\u07d4,$(\xe4\xa6it\xed\xc8\"\xd5\xdb\xfb$\x1b'(\aQX\x89lk\x93[\x8b\xbd@\x00\x00\u07d4,-
\x15\xff9V\x1c\x1br\xed\xa1\xcc\x02\u007f\xfe\xf27C\xa1D\x89\u0500\xed\x9e\xf3+@\x00\x00\u07d4,-
\xb2\x8c3\t7^\xea<mr\xcdm\x0e\xec\x14Z\xfc\xc0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4,BN\xe4\u007fX<\xdc\xe0z\xe3\x18\xb6\xfa\xd4b8\x1dM+\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94,KG\x03\a\xa0Y\x85@U\xd9\x1e\xc3yM\x80\xb5=\x0fJ\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4,R\u0244\x10.\xe0\xcd>1\x82\x1b\x84\xd4\b\x93\x0e\xfa\x1a\u01c9lk\x93[\x8b\xbd@\x00\x00\u07d4,Z-
\n\xbd\xa0;\xbe!W\x81\xb4\xff)l\x8ca\xbd\xba\xf6\x89\x01\xa8\xe5oH\xc0\"\x80\x00\u07d4,[}{\x19Z7\x1b\xf9\xab\u0774/\xe0O/\x1d\x9a\x99\x10\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4,]\xf8ffj\x19K&\u03bb@~J\x1f\xd7>
\x8d^\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94,`?\xf0\xfe\x93alCW>\xf2y\xbf\xea@\x88\x8dj\xe7\x8a\x01\x00\xf4\xb6\xd6gW\x90\x00\x00\xe0\x94,hF\xa1\xaa\x99\x9a\"F\xa2\x87\x05`\x00\xbaM\u02e8\xe6=\x8a\x02\x1f/o\x0f\xc3\xc6\x10\x00\x00\u0794,j\xfc\xd4\x03|\x1e\xd1O\xa7O\xf6u\x8e\tE\

xa1\x85\xa8\xe8\x88\xf4?\xc2\xc0N\xe0\x00\x00\u07d4,ki\x9d\x9e\xad4\x9f\x06\u007fEq\x1a\aJd\x1d\xb6\xa8\x97\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4,o\\\x12L\u01c9\xf8\xbb9\x8e?\x88\x97Q\xbcK`-
\x9e\x89\x01Y\xf2\v\xed\x00\xf0\x00\x00\u07d4,\x83\xae\xb0/\xcf\x06}e\xa4p\x82\xfd\x97x3\xab\x1c\uc449\b'8#%\x8a\xc0\x00\x00\xe0\x94,\x89\xf5\xfd\xca=\x15T\t\xb68\xb9\x8at.U\xebFR\xb7\x8a\x14\u06f2\x19\\\xa2(\x90\x00\x00\u07d4,\x96HI\xb1\xf6\x9c\xc7\u03a4D%8\xed\x87\xfd\xf1l\xfc\x8f\x89lk\x93[\x8b\xbd@\x00\x00\u0794,\x9f\xa7,\x95\xf3}\b\xe9\xa3`\t\u7930\u007f)\xba\xd4\x1a\x88\xdfn\xb0\xb2\xd3\xca\x00\x00\u07d4,\xafk\xf4\xec}Z\x19\xc5\xe0\x89z^\xeb\x01\x1d\xce\xceB\x10\x89\a\x93H5\xa01\x16\x00\x00\u07d4,\xb4\xc3\xc1k\xb1\xc5^|kz\x19\xb1'\xa1\xac\x93\x90\xcc\t\x89\xb8'\x94\xa9$O\f\x80\x00\xe0\x94,\xb5IZPS6\xc2FT\x10\xd1\xca\xe0\x95\xb8\xe1\xba\\\u074a\x04<3\xc1\x93ud\x80\x00\x00\u07d4,\xb6\x15\a:@\xdc\u06d9\xfa\xa8HW.\x98{;\x05n\xfb\x89+X\xad\u06c9\xa2X\x00\x00\u07d4,\xbam]\r\xc2\x04\xea\x8a%\xad\xa2\xe2oVu\xbd_/\u0709H#\xef}\u06da\xf3\x80\x00\u07d4,\xbb\fs\u07d1\xb9\x17@\xb6i;wJ}\x05\x17~\x8eX\x89dI\xe8NG\xa8\xa8\x00\x00\u07d4,\xcbfIM\n\xf6\x89\xab\xf9H=6]x$D\xe7\u07ad\x8965\u026d\xc5\u07a0\x00\x00\u07d4,\xcc\x1f\x1c\xb5\xf4\xa8\x00.\x18k
\x88]\x9d\xbc\x03\f\b\x94\x89lk\x93[\x8b\xbd@\x00\x00\u07d4,\u03c0\xe2\x18\x98\x12^\xb4\xe8\a\u0342\xe0\x9b\x9d(Y/n\x89lk\x93[\x8b\xbd@\x00\x00\u07d4,\u0456\x94\u0452j\x0f\xa9\x18\x9e\u07ba\xfcg\x1c\xf1\xb2\u02a5\x8965\u026d\xc5\u07a0\x00\x00\u07d4,\u04d34\xac~\xacyrW\xab\xe3sa\x95\xf5\xb4\xb5\xce\x0f\x89\x05kGx^7&\x00\x00\u07d4,\u05de\xb5
'\xb1,\x18\x82\x8e>\xaa\xb2\x96\x9b\xfc\u0487\xe9\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4,\xd8xfV\x8d\xd8\x1a\xd4}\x9d:\u0404nZePss\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4,\xdb9De\x06\x16\xe4|\xb1\x82\xe0`2/\xa1Hyx\u0389b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4,\xe1\x1a\x92\xfa\xd0$\xff+>\x87\xe3\xb5B\xe6\xc6\r\xcb\u0656\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4-
\x03&\xb2?\x04\t\xc0\xc0\xe9#hc\xa13\aZ\x94\xba\x18\x89\vg\x9b\xe7[\xe6\xae\x00\x00\u07d4-
\r\xecQ\xa6\xe8s0\xa6\xa8\xfa*\x0fe\u060dJ\xbc\xdfs\x89\n\ad\a\xd3\xf7D\x00\x00\u07d4-
#vkok\x05s}\xad\x80\xa4\x19\xc4\x0e\xdaMw\x10>\x89\xcf\x15&@\xc5\xc80\x00\x00\u07d4-
+\x03#Y\xb3c\x96O\xc1\x1aQ\x82c\xbf\xd0T1\xe8g\x89\b\x1c\x1d\xf7b\x9ep\x00\x00\u07d4-
4\x80\xbf\be\aJr\xc7u\x9e\xe5\x13{Mp\xc5\x1c\xe9\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4-
5\xa9\xdfbu\u007f\u007f\xfa\xd1\x04\x9a\xfb\x06\xcaJ\xfcFLQ\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4-@U\x8b\x06\xf9\n9#\x14U\x92\x12;gt\xe4n1\xf4\x8965\u026d\xc5\u07a0\x00\x00\u07d4-
Bi\x12\xd0Y\xfa\xd9t\v.9\n.\xea\xc0To\xf0\x1b\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4-S-
\xf4\xc69\x11\xd1\u0391\xf6\xd1\xfc\xbf\xf7\x96\x0fx\xa8\x85\x89Z\x85\x96\x8aXx\u0680\x00\u07d4-S\x91\xe98\xb3HX\u03d6[\x84\x051\xd5\xef\xdaA\v\t\x89K\xe4\xe7&{j\xe0\x00\x00\xe0\x94-
[B\xfcY\xeb\xda\r\xfdf\xae\x91K\u008c\x1b\nn\xf8:\x8a+\u0235\x9f\xdc\xd86c\x80\x00\u07d4-
]s5\xac\xb06+G\u07e3\xa8\xa4\xd3\xf5\x94\x95D\u04c0\x89\n\u05ce\xbcZ\xc6 \x00\x00\xe0\x94-
a\xbf\xc5hs\x92<+\x00\t]\xc3\xea\xa0\xf5\x90\u062e\x0f\x8a\x04ef\xdf\xf8\xceU`\x00\x00\u07d4-
e\x11\xfdz8\x00\xb2hT\xc7\xec9\xc0\u0735\xf4\xc4\xe8\xe8\x89\x15\xad\u077a/\x9ew\x00\x00\u07d4-}\\@\u076f\xc4P\xb0Jt\xa4\u06bc+\xb5\xd6e\x00.\x89lk\x93[\x8b\xbd@\x00\x00\u07d4-
\x89\xa8\x00jO\x13z \xdc+\xecF\xfe.\xb3\x12\xea\x96T\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4-
\x8cR2\x9f8\u04a2\xfa\x9c\xba\xf5\u0143\xda\xf1I\v\xb1\x1c\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4-
\x8e\x06\x18\x92\xa5\xdc\xce!\x96j\xe1\xbb\a\x88\xfd>\x8b\xa0Y\x89\r\x8e\\\xe6\x17\xf2\xd5\x00\x

00\u07d4-
\x8e[\xb8\xd3R\x16\x95\xc7~|\x83N\x02\x91\xbf\xac\xeet\b\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4-
\x90\xb4\x15\xa3\x8e.\x19\xcd\xd0/\U000ed069z\xf7\xcb\xf6r\x89\x05\xf3\xc7\xf6A1\xe4\x00\x00\u07d4-\x9b\xado\x1e\xe0*p\xf1\xf1=\xef\\\u0332z\x9a'@1\x89a\t=|,m8\x00\x00\u07d4-
\x9c_\xec\u04b4O\xbbj\x1e\xc72\xea\x05\x9fO\x1f\x9d+\\\x896\xca2f\x1d\x1a\xa7\x00\x00\xe0\x94-\xa6\x17iP\t\xccW\xd2j\u0510\xb3*]\xfb\xeb\x93N^\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4-
\xa7k|9\xb4 \u323a,\x10 \xb0\x85k\x02pd\x8a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4-
\u01ddn\u007fU\xbc\xe2\xe2\xd0\xc0*\xd0|\uca3bR\x93T\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\xe0\x94-
\xca\x0eD\x9a\xb6F\xdb\xdf\u04d3\xa9fb\x96\v\u02b5\xae\x1e\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4-
\xd3%\xfd\xff\xb9{\x19\x99R\x84\xaf\xa5\xab\xdbWJ\x1d\xf1j\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4-\xd5x\xf7@}\xfb\xd5H\xd0^\x95\xcc\u00dcHT)bj\x89\u3bb5sr@\xa0\x00\x00\u07d4-
\xd8\xee\xef\x87\x19J\xbc,\xe7X]\xa1\xe3[|\xeax\f\xb7\x8965\xc6 G9\u0640\x00\u07d4-
\xdf@\x90Wi\xbc\xc4&\xcb,)8\xff\xe0w\xe1\u8758\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4-
\xe0\x96D\x00\u0082\xbd\u05ca\x91\x9ck\xf7|k_yay\x89\n\u05ce\xbcZ\xc6 \x00\x00\xe0\x94-
\xe3\x1a\xfd\x18\x9a\x13\xa7o\xf6\xfes\xea\xd9\xf7K\xb5\u0126)\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4-\xec\x982\x9d\x1f\x96\u00e5\x9c\xaay\x81uTR\xd4\xdaI\u0549\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94-\ue422\x8f\x19-
gj\x87s#+V\xf1\x8f#\x9e/\xad\x8a\x03\xef\xa7\xe7G\xb6\u046d\x00\x00\xe0\x94.\b\x80\xa3E\x96#\a \xf0Z\xc8\xf0e\xaf\x86\x81\u0736\u008a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4.\fW\xb4qP\xf9Z\xa6\xa7\xe1j\xb9\xb1\xcb\xf5C(\x97\x9a\x89\x05k\xc7^-
c\x10\x00\x00\u07d4.\x10\x91\v\xa6\xe0\xbc\x17\xe0UUf\x14\u02c7\t\x0fM~[\x89\n\u05ce\xbcZ\xc6 \x00\x00\xe0\x94.$\xb5\x97\x87;\xb1A\xbd\xb27\xea\x8aZ\xb7Gy\x9a\xf0-
\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4.(\x10\xde\xe4J\xe4\xdf\xf3\xd8cB\xab\x12fW\xd6S\xc36\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4.,\xbdz\xd8%G\xb4\xf5\xff\x8b:\xb5o\x94*dE\xa3\xb0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4.-
~\xa6k\x9fG\xd8\xccR\xc0\x1cR\xb6\u147c}G\x86\x89\xd8\xd4`,&\xbfl\x00\x00\u07d4.C\x93H\u07caBw\xb2*v\x84W\xd1\x15\x8e\x97\xc4\t\x04\x89*\x1e\x9f\xf2o\xbfA\x00\x00\xe0\x94.F\xfc\xeej;\xb1E\xb5\x94\xa2C\xa3\x91?\xce]\xado\xba\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u0794.G\xf2\x87\xf4\x98#7\x13\x85\r1&\x82<\xc6}\xce\xe2U\x88\u029d\x9e\xa5X\xb4\x00\x00\u07d4.N\u1b99j\xa0\xa1\xd9$(\xd0fR\xa6\xbe\xa6\xd2\xd1]\x89lk\x93[\x8b\xbd@\x00\x00\u07d4.R\x91+\xc1\x0e\xa3\x9dT\xe2\x93\xf7\xae\u05b9\x9a\x0fLs\xbe\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4.a\x9fW\xab\xc1\u91ea\x93j\xe3\xa2&Ib\xe7\xeb-
\x9a\x89(\xfb\x9b\x8a\x8aSP\x00\x00\u07d4.d\xa8\xd7\x11\x11\xa2/L]\xe1\xe09\xb36\xf6\x8d9\x8a|\x89lk\x93[\x8b\xbd@\x00\x00\u07d4.i3T=O,\xc0\vSP\xbd\x80h\xba\x92C\u05be\xb0\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94.~\x05\xe2\x9e\u0767\xe4\xae%\xc5\x175C\xef\xd7\x1fm=\x80\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4.\u007fFU
\xec5\xcc#\u058eue\x1b\xb6h\x95D\xa1\x96\x898\xec[r\x1a\x1a&\x80\x00\u07d4.\x8e\xb3\nqn_\xe1\\t#>\x03\x9b\xfb\x11\x06\xe8\x1d\x12\x89\x05k\xc7^-

c\x10\x00\x00\xe0\x94.\x98$\xb5\xc12\x11\x1b\xca$\xdd\xfb\xa7\xe5u\xa5\xcdr\x96\xc1\x8a\x03\xa4\x84Qnm\u007f\xfe\x00\x00\u07d4.\xa5\xfe\xe6?3z7nK\x91\x8e\xa8!H\xf9MH\xa6&\x89e\x0f\x8e\r\u0493\xc5\x00\x00\u07d4.\xafN*F\xb7\x89\xcc\u0088\xc8\xd1\xd9)N?\xb0\x858\x96\x89lk\x93[\x8b\xbd@\x00\x00\u07d4.\xaf\xf9\xf8\xf8\x110d\u04d5z\xc6\xd6\xe1\x1e\xeeB\xc8\x19]\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4.\xba\fn\xe5\xa1\x14\\\x1cW9\x84\x96:`]\x88\nz
\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4.\xc9X\"\xeb\x88{\xc1\x13\xb4q*M\xfd\u007f\x13\xb0\x97\xb5\xe7\x8965\u026d\xc5\u07a0\x00\x00\u07d4.\xcaj<]\x9fD\x9d\tV\xbdC\xfa{M{\xe8CYX\x89lk\xdaip\x9c\xc2\x00\x00\xe0\x94.\xca\xc5\x04\xb23\x86n\xb5\xa4\xa9\x9e{\u0490\x13Y\xe4;=\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4.\xeb\xf5\x942\xb5(\x92\xf98\v\xd1@\xaa\x99\xdc\xf8\xad\f\x0f\x89\b=lz\xabc`\x00\x00\u07d4.\xee\xd5\x04q\xa1\xa2\xbfS\xee0\xb1#.n\x9d\x80\xef\x86m\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4.\xefk\x14\x17\u05f1\x0e\xcf\xc1\x9b\x12:\x8a\x89\xe7>RlX\x89
\x86\xac5\x10R`\x00\x00\u07d4.\xf8i\xf05\vW\xd54x\xd7\x01\xe3\xfe\xe5)\xbc\x91\x1cu\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4.\xf9\xe4eqj\xca\u03f8\xc8%/\xa8\xe7\xbcyi\xeb\xf6\u4255\x9e\xb1\xc0\xe4\xae
\x00\x00\xe0\x94.\xfcLd}\xacj\xca\xc3Uw\xad\"\x17X\xfe\xf6ao\xaa\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4/\x13eu&\xb1w\xca\xd5G\u00d0\x8c\x84\x0e\xffd{E\u0649?v\x84\x9c\xf1\xee,\x80\x00\u07d4/\x18}ZpMZ3\x8c[(v\xa0\x90\xdc\xe9d(N)\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94/%#\u0303O\x00\x86\x05$\x02bb\x96gQ\x86\xa8\u508a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07d4/(*\xbb\xb6\u0523\xc3\xcd;\\\xa8\x12\xf7d>\x800_\x06\x89dI\xe8NG\xa8\xa8\x00\x00\u07d4/+\xba\x1b\x17\x96\x82\x1avo\xced\xb8O(\xech\xf1Z\xea\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4/1]\x90\x16\xe8\xee_Sf\x81
/\x90\x84\xb02TMM\x898<\xd1+\x9e\x86<\x00\x00\u07d4/M\xa7SC\x0f\xc0\x9es\xac\xbc\xcd\xcd\xe9\xdad\u007f+]7\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4/P\x80\xb8?~-
\xc0\xa1\xdd\x11\xb0\x92\xad\x04+\xffx\x8fL\x89\xb4\xf8\xfby#\x1d+\x80\x00\u07d4/a\uf941\x9dp_+\x1eN\xe7T\xae\xb8\xa8\x19Pju\x89O%\x91\xf8\x96\xa6P\x00\x00\xe0\x94/f\xbf\xbf\"b\xef\u030d+\xd0DO\u0170ib\x98\xff\x1e\x8a\x02\x1a\xd95\xf7\x9fv\xd0\x00\x00\u07d4/m\xce\x130\u015e\xf9!`!TW-
MK\xac\xbd\x04\x8a\x8965\u026d\xc5\u07a0\x00\x00\u07d4/}2\x90\x85\x1b\xe5\u01b4\xb4?}Et2\x9fa\xa7\x92\u00c9\x05k\xc7^-
c\x10\x00\x00\u07d4/\x858\x17\xaf\u04f8\xf3\xb8n\x9f`\xeew\xb5\xd9ws\xc0\xe3\x89N\xae\xeaD\xe3h\xb9\x00\x00\u07d4/\xa4\x91\xfbY \xa6WN\xbd(\x9f9\xc1\xb2C\r-
\x9aj\x89lk\x93[\x8b\xbd@\x00\x00\u07d4/\xb5f\xc9K\xbb\xa4\xe3\xcbg\xcd\xda}_\xadq1S\x91\x02\x89lk\x93[\x8b\xbd@\x00\x00\u07d4/\xbbPJ]\xc5'\xd3\xe3\xeb\x00\x85\xe2\xfc<}\xd58\xcbz\x89C\u00b1\x8a\xec<\n\x80\x00\u07d4/\xbc\x85y\x8aX5\x98\xb5\"\x16mn\x9d\xda\x12\x1db}\xbc\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4/\xbc\xef3\x84\xd4
\xe4\xbfa\xa0f\x99\x90\xbcpT\U00065bc9lk\x93[\x8b\xbd@\x00\x00\xe0\x94/\xc8.\xf0v\x93#A&Oaz\f\x80\xddW\x1ej\xe99\x8a\x01\x84$\xf5\xf0\xb1\xb4\xe0\x00\x00\u07d4/\u075by\u07cd\xf50\xadc\xc2\x0eb\xafC\x1a\xe9\x92\x16\xb8\x89\x01#n\xfc\xbc\xbb4\x00\x00\u07d4/\xe0\x02?W\"e\x0f:\x8a\xc0\x10\t\x12^t\xe3\xf8.\x9b\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4/\xe0\xccBKS\xa3\x1f\t\x16\xbe\b\xec\x81\xc5\v\xf8\xea\xb0\xc1\x89
\x86\xac5\x10R`\x00\x00\u07d4/\xe1:\x8d\a\x85\u0787X\xa5\xe4\x18v\xc3n\x91l\xf7Pt\x89\xd8\xd

7&\xb7\x17z\x80\x00\x00\u07d4/\xea\x1b/\x83O\x02\xfcT3?\x8a\x80\x9f\x048\xe5\x87\n\xa9\x89\x01\x18T\xd0\xf9\xce\xe4\x00\x00\u07d4/\xee6\xa4\x9e\xe5\x0e\xcfqo\x10G\x91VFw\x9f\x8b\xa0?\x899B\"\xc4\u0686\xd7\x00\x00\u07d4/\xef\x81G\x8aK.\x80\x98\xdb_\xf3\x87\xba!S\xf4\xe2+y\x896'\xe8\xf7\x127<\x00\x00\u07d4/\xf1`\xc4Or\xa2\x99\xb5\xec-
q\xe2\x8c\xe5Dm/\u02ef\x89\x13\x84\x00\xec\xa3d\xa0\x00\x00\u07d4/\xf1\xcaU\xfd\x9c\xec\x1b\x1f\xe9\U00029af7LQ<\x1e*\xaa\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94/\xf5\u02b1,\r\x95\u007f\xd33\xf3\x82\xee\xb7Q\a\xa6L\xb8\xe8\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94/\xf80\xcfU\xfb\x00\u0560\xe05\x14\xfe\xcdD1K\xd6\xd9\xf1\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4/\xfe\x93\xec\x1aV6\xe9\xee4\xafp\xdf\xf5&\x82\xe6\xffpy\x89lk\x93[\x8b\xbd@\x00\x00\u07d40\x03y\x88p&q\xac\xbe\x89,\x03\xfeW\x88\xaa\x98\xaf(z\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d40$\x8dX\xe4\x14\xb2\x0f\xed:lH+Y\xd9\xd8\xf5\xa4\xb7\xe2\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4019\xbcYd\x03\xd5\u04d3\x1fwLf\u013aFtT\u06c9\\%\xe1J\xea(?\x00\x00\u079408\x00\x87xie\x14\x9e\x81B;\x15\xe3\x13\xba2\xc5\u01c3\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d40:0\xacB\x86\xae\x17\xcfH=\xad{\x87\fk\xd6M{J\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d40?\xba\xeb\xbeF\xb3[n[t\x94j_\x99\xbc\x15\x85\xca\xe7\x89/\x9a\xc0i_[\xba\x00\x00\u07d40ADZ3\xba\x15\x87A\x16\r\x9c4N\xb8\x8e\\0o\x94\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d40H\x01d\xbc\xd8It\xeb\xc0\xd9\f\x9b\x9a\xfa\xb6&\xcd\x1cs\x89+^:\xf1k\x18\x80\x00\x00\u07d40N\u019atTW!\xd71j\xefM\u03f4\x1a\u015e\xe2\xf0\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x940Q\x182\x91\x8d\x804\xa7\xbe\xe7.\xf2\xbf\xeeD\x0e\u02fc\xf6\x8a\x03h\xc8b:\x8bM\x10\x00\x00\u07d40Q?\u029f6\xfdx\x8c\xfe\xa7\xa3@\xe8m\xf9\x82\x94\xa2D\x89\x18;_\x03\xb1G\x9c\x00\x00\u07d40U\xef\xd2`)\xe0\xd1\x1b\x93\r\xf4\xf5;\x16,\x8c?\xd2\u0389\x1b\x1a\b\x927\a=\x00\x00\u07d40]&\xc1\v\xdc\x10?k\x9c!'.\xb7\xcb-
\x91\b\xc4~\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d40_x\xd6\x18\xb9\x90\xb4)[\xac\x8a-
\xfa&(\x84\xf8\x04\xea\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x940d\x89\x9a\x96<Gy\xcb\xf6\x13\xcdi\x80\x84j\xf1\xe6\xece\x8a\x01{w<\xe6\xe5\xdf\n\x00\x00\u07d40s\x04f\xb8\xebm\xc9\rT\x96\xaav\xa3G-
}\xbe\v\xbe\x89lh\xcc\u041b\x02,\x00\x00\u07d40t,\xcd\xf4\xab\xbc\xd0\x05h\x1f\x81Y4\\\x9ey\x05K\x1a\x89$=M\x18\"\x9c\xa2\x00\x00\xe0\x940\x83\xef\x0e\xd4\xc4@\x11\x96wJ\x95\xcfN\u0703\xed\xc1HO\x8a#\xff\xb7\xedee\xd6@\x00\x00\u07d40\x8d\xd2\x1c\xeb\xe7U\x12g\x04\xb4\x8c\x0f\r\xc24\xc6\v\xa9\xb1\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d40\x90\xf8\x13\x0e\xc4Df\xaf\xad\xb3n\xd3\xc9&\x139cg{\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d40\x95D\xb6#,=\xd77\xf9E\xa01\x93\u045b_?e\xb9\x89:\xf3B\xf6~\xf6\xc8\x00\x00\u07d40\x96\u0723A\b\b[\xcf\x04\xaer\xb9Et\xa1>\x1a>\x1d\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d40\x98\xb6]\xb9>\xca\xca\xf75<H\x80\x83\x90\xa2#\xd5v\x84\x89\x18d\x84\xcf{\xb6\xa4\x80\x00\u07d40\xa9\xdarWLQ\xe7\xee\t\x04\xba\x1fs\xa6\xb7\xb8;\x9b\x9d\x89\x01\x18T\xd0\xf9\xce\xe4\x00\x00\u07d40\xac\xd8X\x87_\xa2N\xef\rW/\xc7\xd6*\xad\x0e\xbd\xdc5\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d40\xb6aP\xf1\xa64W\x02?\xddE\xd0\xccl\xb5N\f\x0f\x06\x8965\u026d\xc5\u07a0\x00\x00\u07d40\xbbCW\xcdi\x10\xc8m\"8\xbfr|\xbe\x81Vh\x0eb\x89\x05k\xf9\x1b\x1ae\xeb\x00\x00\u07d40\xbfa\xb2\xd8w\xfe\x10cQ&2o\xa1\x89\u4c31\u00f0\x897\xb4\x89\x85\xa5\xd7\xe6\x00\x00\u07d40\xc0\x11B\x90z\xcb\x15e\xf7\x048\xb9\x98\n\xe71\x81\x878\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x940\xc2j\x8e\x97\x1b\xaa\x18U\xd63\xbap?\x02\x8c\u01c71@\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d40\xdbk\x9b\x10~b\x10/CJ\x9d\u0416\f !\xf5\xceL\x89

\x83\x17\x9bnBS\x00\x00\u07d40\xe33X\xfc!\xc8P\x06\xe4\x0f25}\u0209Y@\xaa\xf0\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d40\xe6\t\x00\xca\xccr\x03\xf3\x14\xdc`CG%Qg\xfc*\x0f\x89lk\x93[\x8b\xbd@\x00\x00\u07d40\u7273\xd2F^\x94nb\x10\xfa[5\xdeN\x8c\x93\b_\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x940\xe9i\x8c\xf1\xe0\x8a\x9d\x04\x8b\xd8\xd8\x04\x8f(\xbe~\xd9@\x9f\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d40\xe9\u0560\b\x8f\x1d\xdb/\u04c0\xe2\xa0I\x19\"f\xc5\x1c\xbf\x89\n\xac\xac\u0679\xe2+\x00\x00\u07d40\xea\xc7@\xe4\xf0,\xb5n\xef\x05&\xe5\xd3\x002&\x00\xd0>\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d40\uc4d2$J!\b\u0247\xbc\\\xdd\xe0\ud7c3z\x81{\x89T\x99%\xf6\xc9\xc5%\x00\x00\xe0\x940\xed\x11\xb7{\xc1~^f\x94\u023c[nG\x98\xf6\x8d\x9c\xa7\x8a\x1eo\xb3B\x1f\xe0)\x9e\x00\x00\u07d40\xf7\xd0%\xd1o{\xee\x10U\x80Ho\x9fV\x1c{\xae?\xef\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x940\xfb\xe5\x88_\x9f\xcc\xe9\xea^\u06c2\xedJ\x11\x96\xdd%\x9a\xed\x8a\x01\x19\xe4\u007f!8\x1f@\x00\x00\u07d41\x04}p?c\xb94$\xfb\xbdn/\x1f\x9et\xde\x13\xe7\t\x89\x9a\x81f\xf7\u6ca7\x80\x00\u07d411?\xfdc[\xf2\xf32HA\xa8\x8c\a\xed\x14aD\xce\xeb\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d41Y\xe9\fH\xa9\x15\x90J\xdf\u24b2/\xa5\xfd^ryk\x896\xaf\xe9\x8f&\x06\x10\x00\x00\u07d41]\xb7C\x9f\xa1\u0574#\xaf\xa7\xddq\x98\xc1\xcft\xc9\x18\xbc\x89
\x86\xac5\x10R`\x00\x00\u07d41^\xf2\xdab\x0f\xd30\xd1.\xe5]\xe5\xf3)\xa6\x96\xe0\xa9h\x89\b!\xab\rD\x14\x98\x00\x00\u07d41n\x92\xa9\x1b\xbd\xa6\x8b\x9e/\x98\xb3\xc0H\x93N<\xc0\xb4\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d41n\xb4\xe4}\xf7\x1bB\xe1mo\xe4h%\xb72{\xaf1$\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d41q\x87~\x9d\x82\f\xc6\x18\xfc\t\x19\xb2\x9e\xfd3?\xdaI4\x8965\u026d\xc5\u07a0\x00\x00\u07d41|\xf4\xa2<\xb1\x91\xcd\xc5c\x12\u009d\x15\xe2\x10\xb3\xb9\xb7\x84\x89\a\xcef\xc5\x0e(@\x00\x00\u07d41\x8b.\xa5\xf0\xaa\xa8y\xc4\xd5\xe5H\xac\x9d\x92\xa0\xc6t\x87\xb7\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d41\x8cv\xec\xfd\x8a\xf6\x8dpUSR\xe1\xf6\x01\xe3Y\x88\x04-
\x89\x1b1\x19.h\xc7\xf0\x00\x00\u07d41\x8f\x1f\x8b\xd2
\xb0U\x8b\x95\xfb3\x10\x0f\xfd\xbbd\r|\xa6\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d41\xaa;\x1e\xbe\x8cM\xbc\xb6\xa7\b\xb1\xd7H1\xe6\x0eIv`\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d41\xab\b\x89f\xec\xc7\"\x92X\xf6\t\x8f\xceh\xcf9\xb3\x84\x85\x8965\u026d\xc5\u07a0\x00\x00\xe0\x941\xadM\x99F\xef\t\xd8\xe9\x88\xd9F\xb1\"\u007f\x91A\x90\x176\x8a\x04\xd8S\xc8\xf8\x90\x89\x80\x00\x00\xe0\x941\xb4;\x01]\x00\x81d<l\xdaF\xa7\a:m\xfd\xbc\xa8%\x8a\n\x97\x91e
\xcd\x18\xe8\x00\x00\u07d41\xcc\xc6\x16\xb3\x11\x82h\xe7]\x9a\xb8\x99l\x88X\xeb\xd7\xf3\u00c9\x15\xae\x0fw\x1c\xa1R\x00\x00\u07d41\xd8\x1dRl\x19^?\x10\xb5\xc6\xdbR\xb5\xe5\x9a\xfb\u0a55\x89\x0eO\xbciD\x9f \x00\x00\u07d41\xe9\xc0\x0f\f
jNN~\x05\"\x17\r\xc8\x1e\x88\xf3\xebp\x89\x91\x8d\xdc:B\xa3\xd4\x00\x00\u07d41\xea\x12\u051a5\xa7@x\r\xde\xea\xec\xe8L\b5\xb2bp\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d41\xean\xab\x19\xd0\ad\xe9\xa9^\x18?+\x1b\"\xfc}\xc4\x0f\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d41\xeb\x12<\x95\xc8+\xf6\x85\xac\xe7\xa7Z\x18\x81\xa2\x89\xef\xca\x10\x891\xe0\t`sq\xbd\x00\x00\u07d41\ud147\x88\xbd\xa4\xd5'\t\x92\"\x1c\xc0B\x06\xecba\r\x89?\xc0GIH\xf3`
\x00\x00\u07d41\xf0\x06\xf3IN\xd6\xc1n\xb9*\xaf\x90D\xfa\x8a\xbb_\u0563\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x942\x01%\x9c\xafsJ\xd7X\x1cV\x10Q\xba\v\xca\u007f\u0594k\x8a&\x1d\xd1\xce/
\x88\x80\x00\x00\u07d42\x03N\x85\x81\xd9HN\x8a\xf4*(\xdf\x19\x012\xec)\xc4f\x89\xbb\x91%T\"c\x90\x00\x00\u07d42 !\x02&x\xa0\x16m

K:\xaaz\xd4\xecK\x88\xb7\u0409\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d42%\xc1\xca_*\x9c\x88\x15k\xb7\xd9\xcd\xc4J2fS\xc2\x14\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d42'\x88\xb5\xe2\x9b\xf4\xf5\xf5Z\xe1\u0773 \x85\xfd\xa9\x1b\x8e\xbe\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d42-o\x9a\x14\r!?L\x80\xcd\x05\x1a\xfe%\xc6
\xbfL}\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d42.\\C\xb0\xf5$8\x96U\xa9\xb3\xff$\xf2\xd4\xdb=\xa1\x0f\x89\xfc\x13\xb6\x9b>~h\x00\x00\u07d424\x86\xcad\xb3uGO\xb2\xb7Y\xa9\xe7\xa15\x85\x9b\xd9\xf6\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d427I\xa3\xb9q\x95\x9eF\u0234\x82-\xca\xfa\xf7\xaa\xf9\xbdn\x89\x01\x16q\xa5\xb2Ep\x00\x00\u07d42:\xadA\xdfKo\xc8\xfe\u038c\x93\x95\x8a\xa9\x01\xfah\bC\x894\x95tD\xb8@\xe8\x00\x00\xe0\x942;<\xfe>\xe6+\xbd\xe2\xa2a\xe5<\xb3\xec\xc0X\x10\xf2\u018a\x02\ub3b1\xa1r\u0738\x00\x00\u07d42?\xca^\xd7\u007fi\x9f\x9d\x990\xf5\xce\xef\xf8\xe5oY\xf0<\x89Hz\x9a0E9D\x00\x00\u07d42H\\\x81\x87(\xc1\x97\xfe\xa4\x87\xfb\xb6\xe8)\x15\x9e\xba\x83p\x899!\xb4\x13\xbcN\xc0\x80\x00\xe0\x942P\xe3\xe8X\xc2j\xde\u032d\xf3jVc\xc2*\xa8LAp\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x942Y\xbd/\xdd\xfb\xbco\xba\u04f6\xe8t\xf0\xbb\xc0,\xda\x18\xb5\x8a\x02\x84`VI[\r\x18\x80\x00\u07d42uIo\xd4\u07491\xfdi\xfb\n\v\x04\xc4\xd1\xff\x87\x9e\xf5\x89\x18-~L\xfd\xa08\x00\x00\u07d42{\xb4\x9euOo\xb4\xf73\xc6\xe0o9\x89\xb4\xf6]K\xee\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d42\x82y\x1do\xd7\x13\xf1\xe9OK\xfdV^\xaax\xb3\xa0Y\x9d\x89Hz\x9a0E9D\x00\x00\u07d42\x83\xeb\u007f\x917\xdd9\xbe\xd5_\xfek\x8d\xc8E\xf3\xe1\xa0y\x89\x03\x97\n\xe9!Ux\x00\x00\u07d42\x86\t\x97\xd70\xb2\xd8;s$\x1a%\xd3f}Q\xc9\b\xef\x89\x1b\x1a\b\x927\a=\x00\x00\xe0\x942\x86\u047cez1,\x88G\xd9<\xb3\xcbyP\xf2\xb0\xc6\xe3\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x942\xa2\r\x02\x8e,b\x18\xb9\xd9[D\\w\x15$cj\"\xef\x8a\x02\x02\xfe\xfb\xf2\xd7\xc2\xf0\x00\x00\u07d42\xa7\x06\x91%\\\x9f\xc9y\x1aOu\u0238\x1f8\x8e\n%\x03\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d42\xb7\xfe\xeb\xc5\u015b\xf6^\x86\x1cL\v\xe4*v\x11\xa5T\x1a\x89w\u9aa8R\\\x10\x00\x00\xe0\x942\xba\x9a}\x04#\xe0:R_\xe2\xeb\xebf\x1d
\x85w\x8b\u060a\x04<3\xc1\x93ud\x80\x00\x00\u07d42\xbb.\x96\x93\xe4\xe0\x854M/\r\xbdF\xa2\x83\u3807\xfd\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x942\xc2\xfd\u2daa\xbb\x80\u5ba2\xb9I\xa2\x17\xf3\xcb\t\"\x83\x8a\x010a`\xaf\xdf
7\x80\x00\u07d42\xd9P\xd5\xe9>\xa1\u0574\x8d\xb4qO\x86{\x03
\xb3\x1c\x0f\x897\b\xba\xed=h\x90\x00\x00\u07d42\u06f6qIT\xe81e\x82\x9aJ\xbb6uxI\xb6\xe4}\x8965\u026d\xc5\u07a0\x00\x00\u07d42\xebd\xbe\x1b]\xed\xe4\b\u01bd\xef\xben@\\\x16\xb7\xed\x02\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d42\xef\\\xdcg\x1d\xf5V*\x90\x1a\xee]\xb7\x16\xb9\xbev\xdc\xf6\x89lk\x93[\x8b\xbd@\x00\x00\u07d42\xf2\x9e\x87'\xa7LkC\x01\xe3\xff\xff\x06\x87\xc1\xb8p\xda\xe9\x8965\u026d\xc5\u07a0\x00\x00\u07d42\xfa\x0e\x86\xcd\b}\u058di1\x90\xf3-\x931\t\t\xedS\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d42\xfb\xee\xd6\xf6&\xfc\xdf\xd5\x1a\xca\xfbs\v\x9e\xef\xf6\x12\xf5d\x89lk\x93[\x8b\xbd@\x00\x00\u07943\x00\xfb\x14\x9a\xde\xd6[\u02e6\xc0N\x9c\u05b7\xa0;\x89;\xb1\x88\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x943\x01\xd9\xca/;\xfe\x02by\xcdh\x19\xf7\x9a)=\x98\x15n\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\xe0\x943\b\xb04f\xc2z\x17\xdf\xe1\xaa\xfc\xeb\x81\xe1m)4Vo\x8a\x03\x99\x92d\x8a#\u0220\x00\x00\u07943\x1a\x1c&\xcci\x94\xcd\xd3\xc1K\xec\xe2v\xff\xffK\x9d\xf7|\x88\xfaz\xed\xdfO\x06\x80\x00\xe0\x943&\xb8\x8d\xe8\x06\x18DT\xc4\v'\xf3\t\xd9\xddm\u03f9x\x8a\x03\xca\\f\u067cD0\x00\x00\xe0\x943)\xeb;\xafCE\xd6\x00\xce\xd4\x0en\x99ueo\x117B\x8a\x01\x0f\b\xed\xa8\xe5U\t\x80\x00\u07d432\r\xd9\x0f+\xaa\x11\r\xd34\x87*\x99\x8f\x14\x84&E<\x8965f3\xeb\xd8\xea\x00\x00\u07d436\xc3\xefn\x8bP\xee\x90\xe0

7\xb1d\xb7\xa8\xea_\xaa\xc6]\x89\x0e\u0223\xa7\x1c\"T\x00\x00\xe0\x9438\fo\xffZ\xcd&Q0\x96)\u06daq\xbf? \u017a\x8a\x03h\xc8b:\x8bM\x10\x00\x00\u07d43:\xd1Yd\x01\xe0Z\xea-6\xcaG1\x8e\xf4\xcd,\xb3\u07c9\x9d\xc0\\\xce(\u00b8\x00\x00\u07d43C@\xeeK\x9c\u0701\xf8P\xa7Q\x16\xd5\x0e\u9d98%\xbf\x89lk\x93[\x8b\xbd@\x00\x00\u07d43H\x1e\x85n\xbe\u050e\xa7\b\xa2t&\xef(\xe8g\xf5|\u0449\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x943V[\xa9\xda,\x03\xe7x\xce\x12)O\b\x1d\xfe\x81\x06M$\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07943X\x1c\xee#0\x88\xc0\x86\r\x94N\f\xf1\u03ab\xb8&\x1c.\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d43XX\xf7I\xf1i\u02bc\xfeR\xb7\x96\xe3\xc1\x1e\xc4~\xa3\u0089\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x943^\"\x02[zw\u00e0t\u01cb\x8e=\xfe\a\x13A\x94n\x8a\x02'\xcas\n\xb3\xf6\xac\x00\x00\u07d43b\x9b\xd5/\x0e\x10{\xc0q\x17ld\xdf\x10\x8fdw}I\x89\x01\xcf\xddth!n\x80\x00\u07d43{;\u07c6\xd7\x13\xdb\xd0{]\xbf\xcc\x02+z{\x19F\xae\x89\xd7\xc1\x98q\x0ef\xb0\x00\x00\u07d43|\xfe\x11W\xa5\u0191
\x10\xddV\x153y\x17i\u00b6\xa6\x8965\u026d\xc5\u07a0\x00\x00\u07d43\xb36\xf5\xba^\xdb{\x1c\xcc~\xb1\xa0\u0644\xc1#\x1d\x0e\u0709lk\x93[\x8b\xbd@\x00\x00\u07d43\xc4\a\x13;\x84\xb3\xcaL=\xed\x1fFX\x90\f8\x10\x16$\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\xe0\x943\xd1r\xab\a\\Q\xdb\x1c\xd4\n\x8c\xa8\xdb\xff\r\x93\xb8C\xbb\x8a\x016x\x05\x10\xd1-\xe3\x80\x00\u07d43\xe9\xb7\x18#\x95.\x1ff\x95\x8c'\x8f\u008b\x11\x96\xa6\u0164\x89\x05k\xc7^-
c\x10\x00\x00\u07d43\xeakxU\xe0[\a\xab\x80\u06b1\xe1M\xe9\xb6I\xe9\x9bl\x89\x1c\xd6\xfb\xadW\xdb\xd0\x00\x00\u07d43\xf1R#1\rD\u078bf6h_:L=\x9cVU\xa5\x89\r\x94b\xc6\xcbKZ\x00\x00\u07d43\xf4\xa6G\x1e\xb1\xbc\xa6\xa9\xf8[;Hr\xe1\aU\xc8+\xe1\x89lk\x93[\x8b\xbd@\x00\x00\u07d43\xfbWzM!O\xe0\x10\xd3,\xca|>\xed\xa6?\x87\xce\xef\x8965\u026d\xc5\u07a0\x00\x00\u07d43\xfdq\x8f\v\x91\xb5\xce\u020a]\xc1^\xec\xf0\xec\xef\xa4\xef=\x89\x17r$\xaa\x84Lr\x00\x00\u07d44\x14\x80\u030c\xb4v\xf8\xd0\x1f\xf3\b\x12\xe7\xc7\x0e\x05\xaf\xaf]\x89lk\x93[\x8b\xbd@\x00\x00\u07d44'-
^ut1]\xca\u9afd1{\xac\x90(\x9dGe\x89b\xa9\x92\xe5:\n\xf0\x00\x00\xe0\x9440\xa1c\x81\xf8i\xf6\xeaT#\x91XU\xe8\x00\x885%\xa9\x8a\x03\xca\\f\u067cD0\x00\x00\u07d441\x86%\x81\x8e\xc1?\x11\x83Z\xe9sS\xce7}oY\n\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d449<]\x91\xb9\xdeYr\x03\xe7[\xacC\t\xb5\xfa=(\u00c9\n\x84Jt$\xd9\xc8\x00\x00\u07d449\x99\x8b$|\xb4\xbf\x8b\xc8\nm+5'\xf1\xdf\xe9\xa6\u0489\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d44C}\x14ed\v\x13l\xb5\x84\x1c?\x93O\x9b\xa0\xb7\t}\x89\t`\xdbwh\x1e\x94\x00\x00\u07d44J\x8d\xb0\x86\xfa\xedN\xfc7\x13\x1b:\"\xb0x-\xadp\x95\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x944fM\"\x0f\xa7\xf3yX\x02J32\u0584\xbc\xc6\xd4\u023d\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d44f\xf6~9cl\x01\xf4;:!\xa0\xe8R\x93%\xc0\x86$\x89-
\xb1\x16vP\xac\xd8\x00\x00\u07d44\x856\x1e\xe6\xbf\x06\xefe\b\xcc\xd2=\x94d\x1f\x81M>/\x89lk\x93[\x8b\xbd@\x00\x00\u07d44\x85\xf6!%d3\xb9\x8aB\x00\xda\xd8W\xef\xe5Y7\uc609lk\x93[\x8b\xbd@\x00\x00\u07d44\x95\x8aF\xd3\x0e0\xb2s\xec\xc6\xe5\xd3X\xa2\x12\xe50~\x8c\x89lk\x93[\x8b\xbd@\x00\x00\u07d44\x97\xddf\xfd\x11\x80q\xa7\x8c,\xb3n@\xb6e\x1c\xc8%\x98\x89\x05\xf1\x01kPv\xd0\x00\x00\xe0\x944\x9a\x81k\x17\xab='\xbb\xc0\xae\x00Q\xf6\xa0p\xbe\x1f\xf2\x9d\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d44\x9d,\x91\x8f\u041e(\a1\x8ef\xceC)\t\x17k\xd5\v\x89<\xb7\x1fQ\xfcU\x80\x00\x00\u07d44\xa0C\x1f\xff^\xad\x92\u007f<id\x96\x16\xdcn\x97\x94_o\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x944\xa8]m$?\xb1\u07f7\xd1\xd2\xd4OSn\x94zL\ue78a\

x04<3\xc1\x93ud\x80\x00\x00\u07d44\xa9\x01\xa6\x9f\x03k\u03dfxC\xc0\xba\x01\xb4&\xe8\xc3\xdc+\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d44\xb4TAn\x9f\xb4'Nj\xdd\xf8SB\x8a\x01\x98\xd6.\xe1\x89\x16\x10Bw\x9f\x1f\xfc\x00\x00\u07d44\xc8\xe5\xf13\x0f\xcbK\x14\xcau\xcb%\x80\xa4\xb9
=
N6\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x944\u211b\xeaX:\xb0\xcc7\x97Q\x90\xf3\"\xb3\x95\x05U\x82\x8a\x01\xa5\xc5\xe8W\xfd\xf2\xb2\x00\x00\u07d44\xfaw\x92\xba\u063b\xd7\xffd\x05b\x14\xa3>\xb6`\f\x1e\xa8\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d44\xff&\xeb`\xa8\u0469ZH\x9f\xae\x13n\xe9\x1dNX\bL\x89
\x86\xac5\x10R`\x00\x00\u07d44\xffX)R\xff$E\x8f{\x13\xd5\x1f\vO\x98p\"\xc1\xfe\x89\x98\x06\xde=\xa6\xe9x\x00\x00\u07d45\x10k\xa9N\x85c\u0533\xcb<\\i,\x10\xe6\x04\xb7\xce\u0609lk\x93[\x8b\xbd@\x00\x00\xe0\x945\x14_b\x03\x97\u019c\xb8\xe0\tb\x96\x1f\x0fH\x86d9\x89\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d45\x14t0\xc3\x10e\x00\u77e2\xf5\x02F.\x94p<#\xb1\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x945\x17\x87\x845\x05\xf8\xe4\xef\xf4ef\xcc\u695fM\x1c_\xe7\x8a\x01\xf5q\x89\x87fKH\x00\x00\xe0\x945\x1f\x16\xe5\xe0sZ\xf5gQ\xb0\xe2%\xb2B\x11q9@\x90\x8a\x02\xd4\xca\x05\xe2\xb4<\xa8\x00\x00\xe0\x945$\xa0\x00#N\xba\xaf\a\x89\xa14\xa2\xa4\x178<\xe5(*\x8a\x011yU\x94}\x8e,\x00\x00\u07d45&\xee\xce\x1ak\xdc>\xe7\xb4\x00\xfe\x93[HF?1\xbe\u05c9\x04w\x87\x9bm\x140\x00\x00\u07d45*x_J\x92\x162PL\xe5\xd0\x15\xf8<I\xaa\x83\x8dm\x89\xe9\xe7\xe0\xfb5\xb7x\x00\x00\xe0\x945-
)\xa2n\x8aA\x81\x81\x81tdg\xf5\x82\xe6\xe8@\x12\xe0\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d45.w\xc8ain\xf9j\xd5I4\xf8\x94\xaa\x8e\xa3QQ\u074965\u026d\xc5\u07a0\x00\x00\xe0\x945/%\xba\xbfJi\x06s\xe3Q\x95\xef\xa8\xf7\x9d\x05\x84\x8a\xad\x8a\x0e%<9\xben}\xc0\x00\x00\u07d456E3\"\xc1Fl\xb9\x05\xaf\\3\\\xa8\xdbt\xbf\xf1\xe6\x89\x18;_\x03\xb1G\x9c\x00\x00\u07d45=\xbe\xc4/\x92\xb5\x0f\x97Q)\xb9<L\x99su\xf0\x90s\x89l]\xb2\xa4\xd8\x15\xdc\x00\x00\u07d45@\u01fdz\x84B\u057e\xe2\x1a!\x80\xa1\xc4\xed\xff\x16I\xe0\x89C.\xacLo\x05\xb9\x80\x00\u07d45I\xbd@\xbb\xbc+0\t\\\xac\x8b\xe2\xc0z\x05\x88\xe0\xae\u0589\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d45R\xa4\x96\xeb\xa6\u007f\x12\xben\xed\xab6\f\xd16a\xdct\x80\x89\x10CV\x1a\x88)0\x00\x00\u07d45T\x94{{\x94{\x00@\xdaR\xca\x18\t%\xc6\u04f8\x8f\xfe\x89\x03\x9f\xba\xe8\xd0B\xdd\x00\x00\u07d45\\\f9\xf5\xd5p\vA\xd3u\xb3\xf1xQ\xdc\xd5$\x01\xf9\x89\u05f3\xb7\xbaZ\xbfL\x00\x00\u07d45\\\xcf\xe0\xe7}U{\x97\x1b\xe1\xa5X\xbc\x02\u07de\xee\x05\x94\x89_\\\xb1\xaf\xc8e(\x00\x00\xe0\x945q\xcfz\xd3\x04\xec\xae\xe5\x95y/K\xbf\xa4\x84A\x85I\u058a\x01;\xcd\r\x89-
\x9e\x16\x00\x00\u07d45u\xc7pf\x8a\x9d\x17\x9f\x1e\xf7h\u0093\xf8\x01f\xe2\xaa=\x89\x19\xb2\x12H\xa3\xef(\x00\x00\u07d45z\x02\xc0\xa9\xdf\xe2\x87\xdeD\u007f\xb6zp\xec[b6fG\x89\x01s\x17\x90SM\xf2\x00\x00\u07d45\x85^\xc6A\xab\x9e\b\x1e\xd0\u00a6\xdc\xd8\x13T\xd0$J\x87\x89A'\xab\u94e7\xaa\x80\x00\u07d45\x88\x89Z\xc9\xfb\xaf\xec\x01
\x92\xdc\x05\xc0\xc3\x02\xd9\a@\xfa\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d45\x99I<\xe6Wr\u03d3\xe9\x8a\xf1\x19^\xc0\x95]\u0240\x02\x89QQY\fg\xb3(\x00\x00\xe0\x945\xa0\x80\x81y\x91s\xe0\x01\xcc[\xd4j\x02@m\xc9]\x17\x87\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d45\xa5I\xe8\xfdl6\x8dm\u0326\xd2\xe7\u044eM\xb9_R\x84\x89\x1b\x1a\b\x927\a=\x00\x00\u07d45\xa6\x88P\x83\u0219\u06bb\xf50\xedl\x12\xf4\xdd: L\xf5\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d45\xaa\xa0F]\x1c&\fB\x0f\xa3\x0e&)\x86\x9f\xb6U\x92\a\x89&7\x81\xe0\xe0\x87\xc8\x00\x00\u07d45\xac\x1d>\xd7FO\xa3\xdb\x14\xe7r\x92\x13\u03aa7\x8c\t^\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d45\xaf\x04\n\f\xc23zv\xaf(\x81T\xc7V\x1e\x1a#3I\x8965\u026d\xc5\u07a0\x00\x00\u

07d45\xb0>\xa4$W6\xf5{\x85\xd2\xebyb\x8f\x03m\xdc\xd7\x05\x89\xd8\xd7&\xb7\x17z\x80\x00\x0
0\u07d45\xbd$he\xfa\xb4\x90\xac\bz\xc1\xf1\xd4\xf2\xc1\r\f\xda\x03\x89\x15\xaf\x1dx\xb5\x8c@\x
00\x00\xe0\x945\xbff\x88R/5Fz\u007fu0#\x14\xc0+\xa1v\x80\x0e\x8a\x03\xafA\x82\x02\xd9T\xe0\
x00\x00\u07d45\u022d\xc1\x11%C+;w\xac\xd6F%\xfeX\xeb\xee\x9df\x89lk\x93[\x8b\xbd@\x00\x0
0\u07d45\u0497\x0fI\xdc\xc8\x1e\xa9\xeep~\x9c\x8a\n\xb2\xa8\xbbtc\x89N\x10\x03\xb2\x8d\x92\x
80\x00\x00\u07d45\xe0\x96\x12\r\xea\xa5\xc1\xec\xb1d^,\u02cbN\xdb\xd9)\x9a\x89\x1b\x1a\xe4\x
d6\xe2\xefP\x00\x00\u07d45\xea!c\xa3\x8c\u07da\x12?\x82\xa5\xec\x00%\x8d\xae\v\xc7g\x89\xd
8\xd7&\xb7\x17z\x80\x00\x00\u07d45\xf1\xda\x12{\x837o\x1b\x88\xc8*3Y\xf6z^g\xddP\x89g\x8a\x
93
b\xe4\x18\x00\x00\u07d45\xf2\x94\x9c\xf7\x8b\xc2\x19\xbbO\x01\x90|\xf3\xb4\xb3\u04c6T\x82\x8
9\x0f\xb5\xc8l\x92\xe44\x00\x00\u07d45\xf5\x86\x01I\xe4\xbb\xc0K\x8a\u0172r\xbeU\xad\x1a\xca
X\xe0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d46\x02E\x8d\xa8omj\x9d\x9e\xb0=\xaf\x97\xfeV\x19\xd4B\xfa\x89lk\x93[\x8b\xbd@\
x00\x00\u07d46\x057-
\x93\xa9\x01\t\x88\x01\x8f\x9f1]\x03.\u0448\x0f\xa1\x89\x1b\x1b\xcfQ\x89j}\x00\x00\u07d46\x16\x
d4H\x98_]2\xae\xfa\x8b\x93\xa9\x93\xe0\x94\xbd\x85I\x86\x89\v\"\u007fc\xbe\x81<\x00\x00\u07d
46\x16\xfbF\xc8\x15x\xc9\xc8\xebM;\xf8\x80E\x1a\x887\x9d}\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d46\x1cu\x93\x16\x96\xbc=B}\x93\xe7lw\xfd\x13\xb2A\xf6\xf4\x89\x1d\xc5\xd8\xfc&m
\xd6\x00\x00\u07d46\x1d\x9e\xd8\v[\xd2|\xf9\xf1\"o&u2X\xee_\x9b?\x89\xbfi\x14\xba}r\xc2\x00\x0
0\u07d46\x1f;\xa9\xed\x95kw\x0f%}6r\xfe\x1f\xf9\xf7\xb0$\f\x89
\x86\xac5\x10R`\x00\x00\u07d46\"|\u07e0\xfd;\x9d~jtF\x85\xf5\xbe\x9a\xa3f\xa7\xf0\x89\n\xc2s\x0
e\xe9\xc6\xc1\x80\x00\u07d46/\xbc\xb1\x06b7\n\x06\x8f\xc2e&\x02\xa2Wy7\xcc\xe6\x89\n\u05ce\
xbcZ\xc6
\x00\x00\u07d460\xc5\xe5e\u03aa\x8a\x0f\x0f\xfe2\x87^\xae*l\xe6<\x19\x89\t7r+7t\xd0\x00\x00\u
07d463\x9f\x84\xa5\u00b4L\xe5=\xfd\xb6\xd4\xf9}\xf7\x82\x12\xa7\u07c9\x11o\x18\xb8\x17\x15\x
a0\x00\x00\u07d464:\xec\xa0{n\u054a\x0eb\xfaN\xcbI\x8a\x12O\xc9q\x89\x10CV\x1a\x88)0\x00\x
00\u07d46au@4\x81\xe0\xab\x15\xbbQF\x15\u02f9\x89\xeb\u018f\x82\x89lk\x93[\x8b\xbd@\x00\
x00\u07d46ro;\x88Z$\xf9)\x96\u0681b^\u022d\x16\xd8\xcb\xe6\x89S\xafu\u0441HW\x80\x00\xe0\
x946s\x95C\x99\xf6\u07feg\x18\x18%\x9b\xb2x\xe2\xe9.\xe3\x15\x8a*Z\x05\x8f\u0095\xed\x00\x0
0\x00\u07d46u\x8e\x04\x9c\u064b\u03a1\"w\xa6v\xf9)sb\x89\x00#\x89\xd8\xd7&\xb7\x17z\x80\x0
0\x00\u07d46\u007fY\u0302yS)8NA\xe1(1\x15\xe7\x91\xf2j\x01\x89lk\x93[\x8b\xbd@\x00\x00\u07
d46\x81\x0f\xf9\xd2\x13\xa2q\xed\xa2\xb8\xaay\x8b\xe6T\xfaK\xbe\x06\x89lk\x93[\x8b\xbd@\x00\
x00\u07d46\x8cT\x14\xb5k\x84U\x17\x1f\xbf\ab
\xc1\u02e4\xb5\xca1\x89\x1e>\xf9\x11\xe8=r\x00\x00\xe0\x946\x90$k\xa3\xc8\x06y\xe2.\xacD\x1
2\xa1\xae\xfc\xe6\xd7\u0342\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d46\x92\x8bU\xbc\x86\x15\t\x
d5\x1c\x8c\xf1\xd5F\xbf\xecn>\x90\xaf\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d46\x98\"\xf5W\x8b@\x
dd\x1fDqpk\"\u0357\x13R\xdak\x89\x12\xc1\xb6\xee\xd0=(\x00\x00\u07d46\x9e\xf7a\x19_:7>$\xe
c\xe6\xcd\"R\x0f\xe0\xb9\xe8n\x89\x1c\xff\xaf\xc9M\xb2\b\x80\x00\u07d46\xa0\x8f\xd6\xfd\x1a\xc
1|\xe1^\xd5~\xef\xb1*+\u2048\xbf\x89Hz\x9a0E9D\x00\x00\u07d46\xa0\xe6\x1e\x1b\xe4\u007f\xa
8~0\xd3(\x88\xee\x030\x90\x1c\xa9\x91\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x946\xb2\xc8^:\x
ee\xeb\xb7\rc\u0124s\f\xe2\xe8\xe8\x8a6$\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x946\x
bfC\xff5\u07d0\x90\x88$3l\x9b1\xce3\x06~/P\x8aIr\x15\x10\xc1\xc1\xe9H\x00\x00\u07d46\xbf\xe1

\xfa;{p\xc1r\xeb\x04/h\x19\xa8\x97%\x95A>\x8965\u026d\xc5\u07a0\x00\x00\xe0\x946\xc5\x10\xbf\x8dnV\x9b\xf2\xf3}G&]\xbc\xb5\x02\xff+\u038a\x06ZM\xa2]0\x16\xc0\x00\x00\xe0\x946\xd8]\xc3h1V\xe6;\xf8\x80\xa9\xfa\xb7x\x8c\xf8\x14:'\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d46\u07cf\x88<\x12s\xec\x8a\x17\x1fz3\xcf\xd6I\xb1\xfe`u\x89\fRHJ\xc4\x16\x89\x00\x00\xe0\x946\xe1Va\f\xd8\xffd\xe7\x80\u061d\x00T8\\\xa7gU\xaa\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d46\xfe\xc6,,B^!\x9b\x18D\x8a\xd7W\x00\x9d\x8cT\x02o\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d47\x00\xe3\x02t$\xd99\xdb\xde]B\xfbx\xf6\xc4\xdb\xec\x1a\x8f\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d47\x02\xe7\x04\xcc!at9\xadN\xa2zW\x14\xf2\xfd\xa1\xe92\x8965\u026d\xc5\u07a0\x00\x00\u07d47\x035\fMo\xe374,\xdd\xc6[\xf1\xe28k\xf3\xf9\xb2\x89m\x81!\xa1\x94\xd1\x10\x00\x00\xe0\x947\b\xe5\x9d\xe6\xb4\x05P\x88x)\x02\xe0W\x9cr\x01\xa8\xbfP\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d47\x126~^U\xa9mZ\x19\x16\x8fn\xb2\xbc~\x99q\xf8i\x8965\u026d\xc5\u07a0\x00\x00\u07d47\x19Zc]\xccb\xf5jq\x80I\xd4~\x8f\x9f\x96\x83(\x91\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d47'4\x1f&\xc1
\x01\xe3x@^\xe3\x8b-
\x84d\xecq@\x89lk\x93[\x8b\xbd@\x00\x00\u07d47.E:kb\x9f'g\x8c\u022e\xb5\xe5|\xe8^\xc0\xae\xf9\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d474\xcb\x18t\x91\xed\xe7\x13\xae[;-
\x12(J\xf4k\x81\x01\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d477!n\xe9\x1f\x17w2\xfbX\xfa@\x97&r\a\xe2\xcfU\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d47<T~\f\xb5\xcec.\x1cZ\xd6aUr\f\x01\xc4\t\x95\x89\xfeT\xdc\xdc\xe6\xc5Z\x00\x00\u07d47l\xd7Ws\x83\xe9\x02\x95\x1b`\xa2\x01{\xa7\xea)\xe35v\x89lk\x93[\x8b\xbd@\x00\x00\u07d47\x8e\xa1\u070e\xdc\x19\xba\xe8&8\x02\x9e\xa8u,\xe9\x8b\xcf\u0349lk\x93[\x8b\xbd@\x00\x00\u07d47\x8f7$??\xf0\xbe\xf5\xe1\u0705\xebC\b\xd94\f)\xf9\x89lnY\xe6|xT\x00\x00\u07d47\x95\x9c
\xb7\xe9\x93\x1dr\xf5\xa8\xae\x86\x9d\xaf\u076d;m\\\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d47\x9a\u007fuZ\x81\xa1~\xdb}\xaa\xa2\x8a\xfcf]\xfak\xe6:\x89\x01Z\xf1\u05cbX\xc4
\x00\x00\u07d47\x9cqf\x84\x9b\xc2J\x02\xd6S^-\xef\x13\xda\xee\xf8\xaa\x8d\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x947\xa0Z\u03b99\\\x865\xa3\x9a|]&j\xe6\x10\xd1\v\xf2\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d47\xa1\x04Q\xf3af\xcfd=\xd2\xdel\x1c\xbb\xa8\xa0\x11\u03e3\x89\x14\x99\x8f2\xacxp\x00\x00\u07d47\xa7\xa6\xffN\xa3\xd6\x0e\xc3\a\xcaQjH\xd3\x05;\xb7\x9c\xbb\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x947\xabf\b:O\xa28H\xb8\x86\xf9\xe6my\xcd\xc1P\xccp\x8a\x12\xbe\"\xff\xb5\xec\x008\x00\x00\u07d47\xac)\xbd\xa9?I{\u012e\xaa\xb95E,C\x15\x104\x1e\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d47\xb8\xbe\xac{\x1c\xa3\x88)\xd6\x1a\xb5R\xc7f\xf4\x8a\x10\xc3/\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d47\xbb\xc4r\x12\xd8/\xcb^\xe0\x8fR%\xec\xc2\x04\x1a\xd2\xda}\x89\xb1\xcf$\xdd\u0431@\x00\x00\u07d47\u02c6\x8d,?\x95\xb2Wa\x1e\xb3JA\x88\u054bt\x98\x02\x89lk\x93[\x8b\xbd@\x00\x00\u07d47\u0640\xa1.\xe3\xbf#\xcc\\\xdbc\xb4\xaeEi\x1ft\xc87\x89lk\x93[\x8b\xbd@\x00\x00\u07d47\xe1i\xa98\b\xd8\x03V\x98\xf8\x15\xc7#V\x13\xc1\xe6Y\xf2\x8965\u026d\xc5\u07a0\x00\x00\u07d47\xea\u0693\xc4u\xde\xd2\xf7\xe1^w\x87\xd4\x00G\x0f\xa5
b\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d47\xfa\xc1\xe6\xbc\x12.\x93m\xfb\x84\xde\fK\xefn\r`\xc2\u05c9lk\x93[\x8b\xbd@\x00\x00\u07d48\a\xef\xf4:\xa9|v\x91\n\x19u-
\xd7\x15\xee\x01\x82\xd9N\x89\r\x90\x15oo\xc2\xfb\x00\x00\u07d48\x15\xb0t?\x94\xfc\x8c\xc8eO\xd9\u0557\xed}\x8bw\xc5~\x89(\t\xd4)\u0616u\x00\x00\xe0\x948\x1d\xb4\xc8F]\xf4F\xa4\xce\x15\xbf\x81\xd4~/\x17\u0240\xbf\x8a\x06\u01b95\xb8\xbb\xd4\x00\x00\x00\u07d48
,\\\xd7\a\x8dO\x88vs\xab\a\x10\x9a\u062d\xa8\x97

\x8965\u026d\xc5\u07a0\x00\x00\u07d48!\x86$\x93$,\n\ub139\r\xe0]%\f\x1eP\xc0t\x89\x11wlX\xe9F\xdc\x00\x00\xe0\x948%\x91\xe7!{C^\x8e\x88L\xdb\xf4\x15\xfe7zo\u278a\x01\xb2\u07dd!\x9fW\x98\x00\x00\u07d48+\xa7m\xb4\x1bu`m\u050aH\xf0\x13~\x91t\xe01\xb6\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d481u~\xaeuW\u02ca7\xa4\xb1\x06D\xb6>M;<u\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x9483\x04\xddzW
\xb2\x9c\x1a\x10\xf6\x03B!\x9fH\x03/\x80\x8a\x01/\x93\x9c\x99\xed\xab\x80\x00\x00\xe0\x948:|\x89\x9e\xe1\x8b\xc2\x14\x96\x98p\xbct\x82\xf6\xd8\xf3W\x0e\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x948C\x0e\x93\x1d\x93\xbe\x01\xb4\xc3\xef\r\xc55\xf1\xe0\xa9a\x00c\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d48C\x9a\xaa$\xe3co:\x18\xe0
\xea\x1d\xa7\xe1E\x16\r\x86\x89\x8c\xf2?\x90\x9c\x0f\xa0\x00\x00\u07d48E\x8e\x06\x85W<\xb4\u048fS\t\x88)\x90Ep\x17\x92f\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d48Gfp8\xf3;\x01\xc1\xccy]\x8d\xafTu\xef\xf5\xa0\u0509'{\x9b\xf4$lA\x00\x00\u07d48d;\xab\xea`\x111l\u01d7\u0670\x93\u0217\xa1{\xda\xe7\x89\x12
\xbbtE\u06a0\x00\x00\u07d48i_\xc7\xe16|\xeb\x16>\xbb\x057Q\xf9\xf6\x8d\xdb\a\xa0\x89lk\x93[\x8b\xbd@\x00\x00\u07d48r\xf4\x8d\xc5\xe3\xf8\x17\xbck*\xd2\xd00\xfc^\x04q\x19=\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d48~\xea\xfdk@\t\u07af\x8b\u0578Zr\x98:\x8d\xcc4\x87\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d48\x81\xde\xfa\xe1\xc0{<\xe0Lx\xab\xe2k\f\u070ds\xf0\x10\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d48\x83\xbe\xcc\b\xb9\xbeh\xad;\b6\xaa\u00f6
\xdc\x00\x17\xef\x89lk\x93[\x8b\xbd@\x00\x00\u07d48\x85\xfe\xe6q\a\xdc:<t\x1e\xe2\x90\u0249\x18\u0279\x93\x97\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d48\x87\x19,\u007fpP\x06\xb60\t\x12v\xb3\x9a\u0180D\x8dk\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d48\x89\x8b\xbbES\xe0\v\xbf\xd0\xcf&\x8b/\xc4d\xd1T\xad\u0549\x11X\xe4`\x91=\x00\x00\x00\u07d48\x8b\xdc\xda\xe7\x94\xfcD\b.fu\x014A\x18\xea\x96\u0356\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d48\x8c\x85\xa9\xb9
}\x81F\x03?\xe3\x81C\xf6\xd3KY\\G\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d48\x96\xadt5y\u04ce#\x02EM\x1f\xb6\xe2\xabi\xe0\x1b\xfd\x89e\xea=\xb7UF`\x00\x00\u07d48\xa3\xdc\xcf/\xcf\xe0\xc9\x1a&$\xbd\f\xbf\x88\xeeJ\al3\x89lk\x93[\x8b\xbd@\x00\x00\u07d48\xa7D\xef\xa6\xd5\xc2\x13}\xef\xef\x8e\xf9\x18{d\x9e\xee\x1cx\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d48\xacfN\xe8\xe0y^Bu\u02c5+\u02e6\xa4y\xad\x9c\x8d\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d48\xb2\x19q\x06\x123\x87\xa0\xd4\xde6\x841\xa8\xba\xcd\xda0\xe2\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d48\xb3\x96\\!\xfa\x8991\a\x9b\xea\xcf\xff\xaf\x156x\xb6\xeb\x89\t<j\nQ\xe2g\x00\x00\u07d48\xb4\x03\xfb\x1f\xb7\xc1EY\xa2\xd6\xf6VJUR\xbc\xa3\x9a\xff\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x948\xb5\x01F\xe7\x19\x16\xa5D\x8d\xe1*Mt!5\xdc\xf3\x983\x8a\x06\u0450\xc4u\x16\x9a
\x00\x00\u07d48\xbf*\x1fzi\xde\x0e%F\xad\xb8\b\xb3c5d]\xa9\xff\x89lp\x049\u0675`\x00\x00\xe0\x948\xc1\v\x90\xc8Y\u02f7\x81V\x92\xf9\x9d\xaeR\n\xb5\xfe\xbf^\x8a\x02\xc9\xe4\x96o\xa5\xcf$\x00\x00\u07d48\u01c5\x1f_\xfdL\xee\x98\xdf0\xf3\xb2U\x97\xaf\x8al\xa2c\x89\x8e\xad:/}~\x18\x00\x00\u07d48\xd2\xe9\x15Id\xb4\x1c\x8dP\xa7H}9\x1e~\xe2\xc3\xd3\u0089\xbd\xbcA\xe04\x8b0\x00\x00\xe0\x948\xda\x1b\xa2\u079e,\x95K\t-
\xd9\xd8\x12\x04\xfd\x01k\xa0\x16\x8a\x02&\x8e\xd0\x1f4\xb30\x00\x00\u07d48\xdf\fJ\xbe}\xed_\xe0h\xea\xdf\x15J\u0191wC$\xa4\x89a\t=|,m8\x00\x00\u07d48\xe2\xafs9>\xa9\x8a\x1d\x99:t\xdf\\\xd7T\xb9\x8dR\x9a\x89a\t=|,m8\x00\x00\u07d48\xe4m\xe4E<8\xe9A\xe7\x93\x0fC0O\x94\xbb{+\xe8\x89l\xb7\xe7Hg\xd5\xe6\x00\x00\u07d48\xe7\u06e8\xfdO\x1f\x85\r\xbc&I\xd8\xe8O\tR\xe3\xe

b<\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d48\xe8\xa3\x1a\xf2\xd2e\xe3\x1a\x9f\xff-
\x8fF(m\x12E\xa4g\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d48\xeao[Z{\x88AuQ\xb4\x12=\xc1'\xd
f\xe94-\xa6\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d48\xee\xc6\xe2\x17\xf4\xd4\x1a\xa9
\xe4$\xb9RQ\x97\x04\x1c\xd4\u0189\xf0\r%\xeb\x92.g\x00\x00\xe0\x948\xf3\x87\xe1\xa4\xedJs\x
10n\xf2\xb4b\xe4t\xe2\xe3\x14:\u040a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d49\x11a\xb0\xe4<0
f\u898d,\xe7\xe1\x99\xec\xdb\x1dW\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x949\x15\uad6b.Yw\
xd0u\xde\xc4}\x96\xb6\x8bK\\\xf5\x15\x8a\r\a\x01\x81\x85\x12\x0f@\x00\x00\u07d49\x1aw@\\\t\x
a7+^\x846#z\xaa\xf9]h\xda\x17\t\x89\x02\xa9&J\xf3\u0479\x00\x00\u07d49\x1f
\x17m\x126\rrMQG\n\x90p6uYJM\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d49$3\xd2\u0383\xd3\xfbJ
v\x02\u0323\xfa\xcaN\xc1@\xa4\xb0\x89\x02\xc3\xc4e\xcaX\xec\x00\x00\xe0\x949?x;\\\u06c6\"\x1
b\xf0)O\xb7\x14\x95\x9c{E\x89\x9c\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d49?\xf4%^\\e\x8f.\u00
7f\x10\xec\xbd)%rg\x1b\xc2\u0489lk\x93[\x8b\xbd@\x00\x00\u07d49A2`\x0fAU\xe0\u007fME\xbc>
\xb8\xd9\xfbr\xdc\u05c4\x89\x9fn\x92\xed\xea\a\xd4\x00\x00\u07d49Q\xe4\x8e<\x86\x9ekr\xa1C\x
b6\xa4Ph\u0379\xd4f\u0409\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x949T\xbd\xfe\v\xf5\x87\u0195\x
a3\x05\xd9$L=[\xdd\xda\u027b\x8a\x04\x10'\x83'\xf9\x85`\x80\x00\u07d49]m%U
\xa8\xdb)\xab\xc4}\x83\xa5\u06ca\x1a}\xf0\x87\x89\x05k\xc7^-
c\x10\x00\x00\u07d49ck%\x81\x1b\x17j\xbf\xcf\xee\xcad\xbc\x87E/\x1f\xdf\xf4\x89\x15\xaf\x1dx\x
b5\x8c@\x00\x00\u07d49i\xb4\xf7\x1b\xb8u\x1e\xdeC\xc0\x166:zaOv\x11\x8e\x89lk\x93[\x8b\xbd
@\x00\x00\xe0\x949x/\xfe\x06\xacx\x82*<:\x8a\xfe0^P\xa5a\x88\u038a\x02\x1e\x19\xe0\u027a\xb
2@\x00\x00\u07d49zn\xf8v:\x18\xf0\x0f\xac!~\x05\\\r0\x94\x10\x10\x11\x89lk\x93[\x8b\xbd@\x00\
x00\u07d49|\u06cc\x80\xc6yP\xb1\x8deB)a\x0e\x93\xbf\xa6\xee\x1a\x89?\x95\xc8\xe0\x82\x15!\x
00\x00\u07d49\x82O\x8b\xce\xd1v\xfd>\xa2.\u01a4\x93\xd0\xcc\xc3?\xc1G\x89\xd8\xd7&\xb7\x17
z\x80\x00\x00\u07d49\x93l'\x19E\v\x94
\xcc%\"\u03d1\xdb\x01\xf2'\xc1\xc1\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d49\x95\xe0\x96\x
b0\x8aZrh\x00\xfc\xd1}\x9cd\xc6N\b\x8d+\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d49\x9a\xa6\xf5\xd0x\xcb\tp\x88+\u0259
\x06\xf8\xfb\xdf4q\x8965\u026d\xc5\u07a0\x00\x00\u07d49\xaa\x05\xe5m}28T!\u03d36\xe9\r=\x1
5\xa9\xf8Y\x89\x01h\u048e?\x00(\x00\x00\u07d49\xaa\xf0\x85M\xb6\xeb9\xbc{.C\x84jv\x17\x1c\x
04E\u0789dI\xe8NG\xa8\xa8\x00\x00\u07d49\xb1\xc4q\xae\x94\xe1!dE.\x81\x1f\xbb\xe2\xb3\xcdr
u\xac\x89lk\x93[\x8b\xbd@\x00\x00\u07d49\xb2\x992t\x90\xd7/\x9a\x9e\xdf\xf1\x1b\x83\xaf\xd0\x
e9\xd3\xc4P\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d49\xba\u018d\x94xY\xf5\x9e\x92&\b\x9c\x96\xd6.\x9f\xbe<\u0789\x02+\x1c\x8c\x12'
\xa0\x00\x00\xe0\x949\xbf\xd9xh\x9b\xec\x04\x8f\xc7v\xaa\x15$\u007f^\x1d|9\xa2\x8a\x04<3\xc1\
x93ud\x80\x00\x00\u07d49\xc7s6|\x88%\xd3YlhoB\xbf\r\x141\x9e?\x84\x89\a?u\u0460\x85\xba\x0
0\x00\u07d49\u05291@,\fy\xc4W\x18o$\u07c7)\u03d5p1\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07
d49\xd6\xca\xca\"\xbc\xcdjr\xf8~\xe7\u05b5\x9e\v\xde!\xd7\x19\x89l\x87T\xc8\xf3\f\b\x00\x00\u07d
49\xe0\xdbM`V\x8c\x80\v\x8cU\x00\x02l%\x94\xf5v\x89`\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94
9\xeeO\xe0\x0f\xbc\xeddph\xd4\xf5|\x01\xcb\"\xa8\v\xcc\u044a\x01EB\xba\x12\xa37\xc0\x00\x00\
u07d49\xf1\x983\x1eK!\xc1\xb7`\xa3\x15_J\xb2\xfe\x00\xa7F\x19\x89lk\x93[\x8b\xbd@\x00\x00\u
07d49\xf4Fc\xd9%a\t\x1b\x82\xa7\r\xcfY=u@\x05\x97:\x89\n\u05cb.\xdc!Y\x80\x00\u07d4:\x03U\x
94\xc7GGmB\xd1\xee\x96l6\"L\xdd\"I\x93\x89\x13J\xf7Ei\xf9\xc5\x00\x00\u07d4:\x04W(G\xd3\x1e
\x81\xf7v\\\xa5\xbf\xc9\xd5W\x15\x9f6\x83\x89\a6-

\r\xab\xea\xfd\x80\x00\xe0\x94:\x06\xe3\xbb\x1e\xdc\xfd\fD\xc3\aM\xe0\xbb`k\x04\x98\x94\xa2\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u0794:\x10\x88\x8b~\x14\x9c\xae',\x010,2}\n\xf0\x1a\v$\x88\xeb\xec!\xee\x1d\xa4\x00\x00\u07d4:1\b\xc1\u6023;3l!\x13\x134@\x9d\x97\xe5\xad\xec\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4:6\x8e\xfeJ\u05c6\xe2c\x95\xec\x9f\u01adi\x8c\xae)\xfe\x01\x89\"E\x89\x96u\xf9\xf4\x00\x00\u07d4:=\xd1\x04\xcd~\xb0O!\x93/\xd43\xeaz\xff\u04d3i\xf5\x89\x13aO#\xe2B&\x00\x00\u07d4:B\x97\xda<U^F\xc0sf\x9d\x04x\xfc\xe7_/y\x0e\x89j\xc5\xc6-
\x94\x86\a\x00\x00\u07d4:Gk\xd2\xc9\xe6d\xc6:\xb2f\xaaLnJH%\xf5\x16\u00c9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4:H\xe0\xa7\t\x8b\x06\xa9\x05\x80+\x87TW1\x11\x8e\x89\xf49\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4:M\xa7\x8d\xce\x05\xae\xb8}\u9bad\x91\x85rm\xa1\x92g\x98\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4:Y\xa0\x82F\xa8
o\x8dX\xf7\v\xb1\xf0\xd3\\[\xccq\xbd\x89\n\ad\a\xd3\xf7D\x00\x00\xe0\x94:r\xd65\xaa\xde\xeeC\x824\x9d\xb9\x8a\x18\x13\xa4\xcf\xeb=\xf1\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\xe0\x94:}\xb2$\xac\xae\x17\xdew\x98y}\x82\xcd\xf8%0\x17\u07e8\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4:\x80_\xa0\xf78\u007fs\x05[xX\u0285\x19\xed\xd9=cO\x89dI\xe8NG\xa8\xa8\x00\x00\xe0\x94:\x84\xe9P\xedA\x0eQ\xb7\xe8\x80\x10I\xab&4\xb2\x85\xfe\xa1\x8a\x03\xf5/\u06a8\"\xd2\xc8\x00\x00\u07d4:\x86\ue506+t=\xd3OA\ti\xd9N,VR\u052d\x89\n\xed\xe6\x9a\xd3\x0e\x81\x00\x00\u07d4:\x912\xb7\t=>\xc4.\x1eO\xb8\xcb1\xec\xddC\xaew<\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94:\x99`&m\xf6I
cS\x8a\x99\xf4\x87\xc9P\xa3\xa5\uc78a\x05\x15\n\xe8J\x8c\xdf\x00\x00\x00\u07d4:\x9b\x11\x10)\xce\x1f
\xc9\x10\x9czt\xee\xee\xf3OO.\xb2\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4:\x9eTA\xd4K$;\xe5[u\x02z\x1c\ub7ac\xf5\r\xf2\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94:\xa0z4\xa1\xaf\u0216}=\x13\x83\xb9kb\u03d6\xd5\xfa\x90\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94:\xa4,!\xb9\xb3\x1c>'\xcc\xd1~\t\x9a\xf6y\xcd\xf5i\a\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4:\xa9H\xea\x029wU\xef\xfb/\x9d\xc99-
\xf1\x05\x8f~3\x89.\x14\x1e\xa0\x81\xca\b\x00\x00\u07d4:\xad\xf9\x8ba\xe5\u0216\xe7\xd1\x00\xa39\x1d2P\"]a\u07c9\f\xafg\x007\x01h\x00\x00\u07d4:\xaeHr\xfd\x90\x93\xcb\xca\xd1@o\x1e\x80x\xba\xb5\x03Y\xe2\x89\x02\"\xc8\xeb?\xf6d\x00\x00\u07d4:\xbb\x8a\xdf\xc6\x04\xf4\x8dY\x84\x81\x1d\u007f\x1dR\xfe\xf6u\x82p\x89\xf2\x97\x19\xb6o\x11\f\x00\x00\u07d4:\xc2\xf0\xff\x16\x12\xe4\xa1\xc3F\xd53\x82\xab\xf6\u0622[\xaaS\x89lk\x93[\x8b\xbd@\x00\x00\u07d4:\xc9\xdczCj\xe9\x8f\xd0\x1cz\x96!\xaa\x8e\x9d\v\x8bS\x1d\x89a\t=|,m8\x00\x00\xe0\x94:\xd0aI\xb2\x1cU\xff\x86|\xc3\xfb\x97@\u04bc\xc7\x10\x121\x8a)\xb7d2\xb9DQ
\x00\x00\u07d4:\xd7\x02C\u060b\xf0@\x0fW\xc8\xc1\xfdW\x81\x18H\xaf\x16*\x89.\x9e\xe5\u00c6S\xf0\x00\x00\u07d4:\xd9\x15\xd5P\xb7#AV
\xf5\xa9\xb5\xb8\x8a\x85\xf3\x82\xf05\x8965\u026d\xc5\u07a0\x00\x00\u07d4:\xe1`\xe3\xcd`\xae1\xb9\xd6t-
h\xe1Nv\xbd\x96\xc5\x17\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\u07d4:\xe6+\xd2q\xa7`c\u007f\xady\xc3\x1c\x94\xffb\xb4\xcd\x12\xf7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4:\xeaN\x82\xd2@\x02H\xf9\x98q\xa4\x1c\xa2W\x06\r:\"\x1b\x8965\u026d\xc5\u07a0\x00\x00\u07d4:\xf6[>(\x89ZJ\x00\x11S9\x1d\x1ei\xc3\x1f\xb9\xdb9\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4;\a\xdbZ5\u007fZ\xf2HL\xbc\x9dw\xd7;\x1f\xd0Q\x9f\u01c9\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4;\n\u032fK`|\xfea\xd1s4\xc2\x14\xb7\\\xde\xfd\xbd\x89\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;\x13c\x1a\x1b\x89\xcbVeH\x89\x9a\x

1d`\x91\\\xdc\xc4
[\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;\x15\x90\x99\aR\a\u0180vc\xb1\xf0\xf7\xed\xa5J\xc8\xcc\xe3\x89j\xc4\xe6[i\xf9-
\x80\x00\u07d4;\x197\xd5\u74f8\x9bc\xfb\x8e\xb5\xf1\xb1\xc9\xcak\xa0\xfa\x8e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;\"\xda*\x02q\xc8\xef\xe1\x02S'scji\xb1\xc1~\t\x89\x1b6\xa6DJ>\x18\x00\x00\u07d4;\"\u07a3\xc2_\x1bY\u01fd'\xbb\x91\u04e3\xea\xec\xef9\x84\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94;#g\xf8IK_\xe1\x8dh<\x05]\x89\x99\x9c\x9f=\x1b4\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4;,E\x99\x0e!GDQ\xcfOY\xf0\x19U\xb31\xc7\xd7\u0249lk\x93[\x8b\xbd@\x00\x00\xe0\x94;A\x00\xe3\ns\xb0\xc74\xb1\x8f\xfa\x84&\u045b\x191/\x1a\x8a\v\xb5\u046ap\n\xfd\x90\x00\x00\u07d4;B\xa6m\x97\x9fX(4tz\x8b`B\x8e\x9bN\xec\xcd#\x89!\xa1\u01d0\xfa\xdcX\x00\x00\u07d4;Gh\xfdq\xe2\xdb,\xbe\u007f\xa0PH<'\xb4\xeb\x93\x1d\xf3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;Vj\x8a\xfa\u0456\x82\xdc,\xe8g\x9a<\xe4D\xa5\xb0\xfdO\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94;\\%\x1d\u007f\u05c9;\xa2\t\xfeT\x1c\xec\xd0\xce%:\x99\r\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4;^\x8b<w\xf7\x92\xde\xcbz\x89\x85\u07d1n\xfbI\n\xac#\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;n\x81Ow\aH\xa7\u00d9x\x064v\x05H\n?\xd5\t\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;{OS\xc4VU\xf3\xdc_\x01~\xdc#\xb1o\x9b\xc56\xfa\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94;{\x8e'\xde3\xd3\xceya\xb9\x8d\x19\xa5/\xe7\x9fl%\xbe\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4;|w\xdb\xe9]\xc2`,\xe3&\x9a\x95E\xd0Ie\xfe\xfd\xbd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;\x80\x98S?}\x9b\xdc\xd3\a\u06f2>\x17w\xca\x18A\x896\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94;\x93\xb1a6\xf1\x1e\xaf\x10\x99l\x95\x99\r;'9\xcc\xea_\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4;\xabK\x01\xa7\xc8K\xa1?\uea70\xbb\x19\x1bw\xa3\xaa\u0723\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4;\xb55\x98\xcc
\xe2\x05]\xc5S\xb0I@J\u0277\xdd\x1e\x83\x89!W\x1d\xf7|\x00\xbe\x00\x00\u07d4;\xbc\x13\xd0J\xcc\xc0pz\xeb\u072e\xf0\x87\u0438~\v^\u327e\xd1\xd0&=\x9f\x00\x00\x00\u07d4;\xc6\xe3\xeezV\u038f\x14\xa3u2Y\x0fcqk\x99f\xe8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;\xc8]ls[\x9c\xdaK\xba_H\xb2K\x13\xe7\x0600{\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4;\xd6$\xb5H\xcbe\x976\x90~\u062a<\fp^$\xb5u\x89lk\x93[\x8b\xbd@\x00\x00\u07d4;\u0660m\x1b\xd3lN\xdd'\xfc\r\x1f[\b\x8d\xda\xe3\xc7*\x89\x1b\x1azB\v\xa0\r\x00\x00\u0794;\u077c\x814\xf7}UY\u007f\xc9|&\xd2f\x98\t\x06\x04\ub23e-
j\x0e\xda\x00\x00\xe0\x94;\xf8n\u0623\x15>\xc93xj\x02\xac\t\x03\x01\x85^Wk\x8a_J\x8c\x83u\xd1U@\x00\x00\u07d4;\xfb\u04c4|\x17\xa6\x1c\xf3\xf1{R\xf8\ub879`\xb3\U000df262\xa1]\tQ\x9b\xe0\x00\x00\u07d4<\x03\xbb\xc0#\xe1\xe9?\xa3\xa3\xa6\xe4(\xcf\f\xd8\xf9^\x1e\u0189Rf<\u02b1\xe1\xc0\x00\x00\u07d4<\f=\ufb1c\xeaz\xcc1\x9a\x96\xc3\v\x8e\x1f\xed\xabEt\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4<\x15\xb3Q\x1d\xf6\xf04.sH\u0309\xaf9\xa1h\xb7s\x0f\x8965\u026d\xc5\u07a0\x00\x00\u07d4<\x1f\x91\xf3\x01\xf4\xb5e\xbc\xa2GQ\xaa\x1fv\x13\"p\x9d\u0749a\t=|,m8\x00\x00\xe0\x94<(l\xfb0\x14n_\u05d0\xc2\xc8T\x15RW\x8d\xe34\u060a\x02)\x1b\x11\xaa0n\x8c\x00\x00\u07d4<2.a\x1f\u06c2\rG\xc6\xf8\xfcd\xb6\xfa\xd7L\xa9_^\x89\r%\x8e\xce\x1b\x13\x15\x00\x00\u07d4<Z$\x14Y\u01ab\xbfc\x029\u024a0\xd2\v\x8b:\xc5a\x89\b\x8b#\xac\xff\u0650\x00\x00\u07d4<y\xc8c\xc3\xd3r\xb3\xff\foE'4\xa7\xf9pB\xd7\x06\x89\t\x8a}\x9b\x83\x14\xc0\x00\x00\xe0\x94<\x83\xc1p\x1d\xb08\x8bh!\r\x00\xf5q|\u067d2,j\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4<\x86\x0e.f?F\xdbSB{)\xfe>\xa5\xe5\xbfb\xbb\u0309\x05V\xf6L\x1f\xe7\xfa\x00\x00\u07d4<\x86\x9c\tie#\xce\xd8$\xa0pAF\x05\xbbv#\x1f\xf2\x8965\u026d\xc5\u07a0\x00\x00\u07d4<\x92V\x19\u02731DF?\x057\u061

65\x87\x06\xc5
\xb0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4<\x98YK\xf6\x8bW5\x1e\x88\x14\xae\x9em\xfd-
%J\xa0o\x89\x10CV\x1a\x88)0\x00\x00\u07d4<\xad\xeb=>\xed?b1\x1dRU>p\xdfJ\xfc\xe5o#\x89\x
d8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94<\xae\xdbS\x19\xfe\x80eC\xc5nP!\xd3r\xf7\x1b\xe9\x06.\x8
a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4<\xaf\xaf^bPV\x15\x06\x8a\xf8\xeb\"\xa1:\u0629\xe5Pp\x8
9lf\x06E\xaaG\x18\x00\x00\u07d4<\xb1y\xcbH\x01\xa9\x9b\x95\u00f0\xc3$\xa2\xbd\xc1\x01\xa6S
`\x89\x01h\u048e?\x00(\x00\x00\u07d4<\xb5a\u0386BK5\x98\x91\xe3d\xec\x92_\xfe\xff'}\xf7\x89\
n\u05ce\xbcZ\xc6 \x00\x00\u07d4<\xcbq\xaah\x80\xcb\v\x84\x01-
\x90\xe6\a@\xec\x06\xac\u05cf\x89lk\x93[\x8b\xbd@\x00\x00\u07d4<\xce\xf8\x86yW9G\xe9I\x97y
\x8a\x1e2~\b`:e\x89+\xc9\x16\u059f;\x02\x00\x00\xe0\x94<\xd1\xd9s\x1b\xd5H\xc1\xddo\u03a6\x
1b\xebu\xd9\x17T\xf7\u04ca\x01\x16\x1d\x01\xb2\x15\xca\xe4\x80\x00\u07d4<\u04e6\xe95y\xc5
mIAq\xfcS>z\x90\xe6\xf5\x94d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4<\u05b7Y<\xbe\xe7x0\xa8\xb1
\x9d\b\x01\x95\x8f\xcdK\xc5z\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4<\xd7\xf7\xc7\xc257\x
80\xcd\xe0\x81\xee\xecE\x82+%\xf2\x86\f\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4<\xe1\u0717\xfc\u05f7\xc4\u04e1\x8aI\xd6\xf2\xa5\xc1\xb1\xa9\x06\u05c9\n\u05ce
\xbcZ\xc6
\x00\x00\u07d4<\xea0*G*\x94\x03y\xdd9\x8a$\xea\xfd\xba\u07c8\xady\x89\xa2\xa1]\tQ\x9b\xe0\x
00\x00\xe0\x94<\xec\xa9k\xb1\xcd\xc2\x14\x02\x9c\xbc^\x18\x1d9\x8a\xb9M=A\x8a\x10\xf0\xcf\x0
6M\u0552\x00\x00\x00\u07d4<\xf4\x84RO\xbd\xfa\xda\xe2m\xc1\x85\xe3++c\x0f\xd2\xe7&\x89\x1
8TR\xcb*\x91\xc3\x00\x00\u07d4<\xf9\xa1\xd4e\xe7\x8bp9\xe3iDx\xe2b{6\xfc\xd1A\x89J`S*\xd5\x
1b\xf0\x00\x00\u07d4<\xfb\xf0fVYpc\x9e\x13\r\xf2\xa7\xd1k\x0e\x14\xd6\t\x1c\x89\\(=A\x03\x94\x1
0\x00\x00\xe0\x94=\th\x8d\x93\xad\a\xf3\xab\xe6\x8cr'#\xcdh\t\x90C^\x8a\x06ZL\xe9\x9fv\x9en\x0
0\x00\u07d4=1X{_\u0546\x98Ex\x87%\xa6c)\nI\xd3g\x8c\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u
07d4=?\xadI\xc9\xe5\xd2u\x9c\x8e\x8eZzM`\xa0\xdd\x13V\x92\x89\x01\x15\x8eF\t\x13\xd0\x00\x0
0\u07d4=WO\xcf\x00\xfa\xe1\u064c\u023f\x9d\u07e1\xb3\x95;\x97A\xbc\x89j\xcb=\xf2~\x1f\x88\x
00\x00\u07d4=Z\x8b+\x80\xbe\x8b5\xd8\xec\xf7\x89\xb5\xedz\au\xc5\al\x89\x01\x15\x8eF\t\x13\x
d0\x00\x00\u07d4=f\xcdK\xd6M\\\x8c\x1b^\xea(\x1e\x10m\x1cZ\xad#s\x89i\xc4\xf3\xa8\xa1\x10\xa
6\x00\x00\u0794=j\xe0S\xfc\xbc1\x8do\xd0\xfb\xc3S\xb8\xbfT.h\r'\x88\xc6s\xce<@\x16\x00\x00\u
07d4=o\xf8,\x93w\x05\x9f\xb3\r\x92\x15r?`\xc7u\u0211\xfe\x89\r\x8e\\\xe6\x17\xf2\xd5\x00\x00\u0
7d4=y\xa8S\xd7\x1b\xe0b\x1bD\xe2\x97Yel\xa0u\xfd\xf4\t\x89lk\x93[\x8b\xbd@\x00\x00\u07d4=~\
xa5\xbf\x03R\x81\x00\xed\x8a\xf8\xae\xd2e>\x92\x1bng%\x8965\u026d\xc5\u07a0\x00\x00\u07d4
=\x81?\xf2\xb6\xedW\xb97\u06bf+8\x1d\x14\x8aA\x1f\xa0\x85\x89\x05k\xc7^-
c\x10\x00\x00\u07d4=\x88\x143\xf0J}\r'\xf8ID\xe0\x8aQ-\xa3UR\x87\x89A\rXj
\xa4\xc0\x00\x00\u07d4=\x89\xe5\x05\xcbF\xe2\x11\xa5?2\xf1g\xa8w\xbe\xc8\u007fK\n\x89\x01[5
W\xf1\x93\u007f\x80\x00\xe0\x94=\x8d\a#r\x1es\xa6\xc0\xd8`\xaa\x05W\xab\xd1L\x1e\xe3b\x8a\x
01\x0f\f\xf0d\xddY
\x00\x00\u07d4=\x8f9\x88\x1b\x9e\xdf\xe9\x12'\xc3?\xa4\xcd\xd9\x1eg\x85D\xb0\x89\x04\xab\a\x
baC\xad\xa9\x80\x00\u07d4=\x9dk\xe5\u007f\xf8>\x06Y\x85fO\x12VD\x83\xf2\xe6\x00\xb2\x89n\x
ac\xe4?#\xbd\x80\x00\x00\u07d4=\xa3\x9c\xe3\xefJz9f\xb3.\xe7\xeaN\xbc#5\xa8\xf1\x1f\x89lk\x93
[\x8b\xbd@\x00\x00\u07d4=\xaa\x01\u03b7\x0e\xaf\x95\x91\xfaR\x1b\xa4\xa2~\xa9\xfb\x8e\xdeJ\
x89Zc\xd2\u027cvT\x00\x00\u07d4=\xb5\xfejh\xbd6\x12\xac\x15\xa9\x9aa\xe5U\x92\x8e\xec\xea\
xf3\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4=\xb9\xed\u007f\x02L~&7/\xea\xcf+\x05\b\x03D^8\x

10\x89E\xb1H\xb4\x99j0\x00\x00\u07d4=\xbf\r\xbf\xd7x\x90\x80\x053\xf0\x9d\xea\x83\x01\xb9\xf0%\u04a6\x8965\u026d\xc5\u07a0\x00\x00\u07d4=\xce\U0005c18b\x15\xd3N\xdaBn\xc7\xe0K\x18\xb6\x01p\x02\x89lh\xcc\u041b\x02,\x00\x00\xe0\x94=\xd1.Uj`76\xfe\xbaJo\xa8\xbdJ\xc4]f*\x04\x8a#u{\x91\x83\xe0x(\x00\x00\u07d4=\u078b\x15\xb3\u033a\xa5x\x01\x12\xc3\xd6t\xf3\x13\xbb\xa6\x80&\x89`\x1dQZ>O\x94\x00\x00\xe0\x94=\xde\xdb\xe4\x89#\xfb\xf9\xe56\xbf\x9f\xfb\aG\xc9\xcd\u04de\xef\x8a\x03h\xc8b:\x8bM\x10\x00\x00\u07d4=\xea\xe43'\x91?b\x80\x8f\xaa\x1bbv\xa2\xbdch\xea\u0649lk\x93[\x8b\xbd@\x00\x00\u07d4=\xf7b\x04\x9e\u068a\u0192}\x90Lz\xf4/\x94\xe5Q\x96\x01\x89lk\x93[\x8b\xbd@\x00\x00\u07d4>\x04\r@\u02c0\xba\x01%\xf3\xb1_\xde\xfc\xc8?0\x05\xda\x1b\x898E$\xccp\xb7x\x00\x00\u07d4>\v\x8e\xd8n\xd6i\xe1'#\xafur\xfb\xac\xfe\x82\x9b\x1e\x16\x89QM\xe7\xf9\xb8\x12\xdc\x00\x00\xe0\x94>\f\xbejm\xcba\xf1\x10\xc4[\xa2\xaa6\x1d\u007f\xca\xd3\xdas\x8a\x01\xb2\u07dd!\x9fW\x98\x00\x00\u07d4>\x19KN\xce\xf8\xbbq\x1e\xa2\xff$\xfe\xc4\xe8{\xd02\xf7\u0449\x8b\x9d\xc1\xbc\x1a\x03j\x80\x00\xe0\x94>\x1b\"0\xaf\xbb\xd3\x10\xb4\x92jLwmZ\u705cf\x1d\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4>\x1cS0\x0eL\x16\x89\x12\x16<~\x99\xb9]\xa2h\xad(\n\x896b2\\\u044f\xe0\x00\x00\u07d4>\x1c\x96
c\xe0\xd5)YA\xf2\x10\u0723\xabS\x1e\xec\x88\t\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4>,\xa0\xd24\xba\xf6\a\xadFj\x1b\x85\xf4\xa6H\x8e\xf0\n\xe7\x89\x04\xda!\xa3H=V\x80\x00\u07d4>/&#^\x13zs$\xe4\xdc\x15K]\xf5\xafF\xea\x1aI\x89\x017\xaa\xd8\x03-
\xb9\x00\x00\xe0\x94>1a\xf1\xea/\xbf\x12ny\xda\x18\x01\u0695\x12\xb3y\x88\u024a\nm\xd9\f\xaeQ\x14H\x00\x00\xe0\x94>6\xc1rS\xc1\x1c\xf3\x89t\xed\r\xb1\xb7Y\x16\r\xa67\x83\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4><\u04fe\xc0e\x91\xd64o%Kb\x1e\xb4\x1c\x89\x00\x8d1\x895\u07fe\u069f74\x00\x00\u07d4>E\xbdU\u06d0`\xec\xed\x92;\xb9\xcbs<\xb3W?\xb51\x89X\xe7\x92n\xe8X\xa0\x00\x00\u07d4>M\x13\xc5Z\x84\xe4n\xd7\xe9\u02d0\xfd5^\x8a\u0651\u33c965\u026d\xc5\u07a0\x00\x00\u07d4>N\x92e\"<\x9782L\xf2\v\xd0`\x06\xd0\a>\u06cc\x89\a?u\u0460\x85\xba\x00\x00\xe0\x94>O\xbdf\x10\x15\xf6F\x1e\xd6s\\\xef\xef\x01\xf3\x14E\xde:\x8a\x03n4)\x98\xb8\xb0
\x00\x00\xe0\x94>S\xff!\a\xa8\u07be3(I:\x92\xa5\x86\xa7\xe1\xf4\x97X\x8a\x04\xe6\x9c*q\xa4\x05\xab\x00\x00\u07d4>Z9\xfd\xdap\xdf\x11&\xab\r\u011asx1\x1aSz\x1f\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\xe0\x94>Z\xbd\t\xceZ\xf7\xba\x84\x87\xc3Y\xe0\xf2\xa9:\x98k\v\x18\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4>\\\xb8\x92\x8cAx%\xc0:;\xfc\xc5!\x83\xe5\xc9\x1eB\u05c9\xe71\xd9\xc5,\x96/\x00\x00\u07d4>^\x93\xfbL\x9c\x9d\x12F\xf8\xf2G5\x8e\"\xc3\xc5\xd1{j\x89\b!\xab\rD\x14\x98\x00\x00\u07d4>a\x83P\xfa\x01ez\xb0\xef>\xba\xc8\xe3p\x12\xf8\xfc+o\x89\x98\x06\xde=\xa6\xe9x\x00\x00\u07d4>c\xce;$\xca(e\xb4\u0166\x87\xb7\xae\xa3Y~\xf6\xe5H\x89lk\x93[\x8b\xbd@\x00\x00\u07d4>f\xb8GiVj\xb6yE\xd5\xfa\x8175V\xbc\u00e1\xfa\x89\b=lz\xabc`\x00\x00\xe0\x94>v\xa6-
\xb1\x87\xaat\xf68\x17S;0l\xea\xd0\xe8\u03be\x8a\x06\x9bZ\xfa\xc7P\xbb\x80\x00\x00\u07d4>z\x96k]\xc3W\xff\xb0~\x9f\xe0g\xc4W\x91\xfd\x8e0I\x89\x034-
`\xdf\xf1\x96\x00\x00\xe0\x94>\x81w!u#~\xb4\xcb\xe0\xfe-
\xca\xfd\xad\xff\xebj\x19\x99\x8a\x01\xdd\f\x88_\x9a\r\x80\x00\x00\u07d4>\x83I\xb6\u007fWED\x9fe\x93g\u066dG\x12\xdb[\x89Z\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4>\x83TO\x00\x82U%r\u01c2\xbe\xe5\xd2\x18\xf1\xef\x06J\x9d\x89\x05l\xd5_\xc6M\xfe\x00\x00\u07d4>\x84\xb3\\[\"ePpa\xd3\vo\x12\xda\x03?\xe6\xf8\xb9\x89a\t=|,m8\x00\x00\u07d4>\x86A\xd4<B\x00?\n3\xc9)\xf7\x11\a\x9d\xeb+\x9eF\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94>\x87E\xba2/_\xd6\xcbP\x12N\xc4f\x88\u01e6\x9a\u007f\xae\x8a\x01\n\xfc\x1a\xde;N\xd4\x00\x00\u07d4>\x91N0\x18\xac\x00D\x93A\u0

11d\xa7\x1d\x04\xdf\xee\xedb!\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4>\x94\x10\u04f9\xa8~\xd5\xe4Q\xa6\xb9\x1b\xb8\x92?\xe9\x0f\xb2\xb5\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4>\x94\xdfS\x13\xfaR\x05p\xef#+\xc31\x1d_b/\xf1\x83\x89lk\x93[\x8b\xbd@\x00\x00\u0794>\x9b4\xa5\u007f3u\xaeY\xc0\xa7^\x19\u0136A\"\x8d\x97\x00\x88\xf8i\x93)g~\x00\x00\u07d4>\xad\xa8\xc9/V\x06~\x1b\xb7<\xe3x\xdaV\xdc,\xdf\xd3e\x89w\xcd\xe9:\xeb\rH\x00\x00\xe0\x94>\xaf\by\xb5\xb6\xdb\x15\x9bX\x9f\x84W\x8bjt\xf6\xc1\x03W\x8a\x01\x898\xb6q\xfae\xa2\x80\x00\u07d4>\xaf1k\x87a]\x88\xf7\xad\xc7|X\xe7\x12\xedMw\x96k\x89\x05m\xbcL\xee$d\x80\x00\u07d4>\xb8\xb3;!\xd2<\u0686\xd8(\x88\x84\xabG\x0e\x16F\x91\xb5\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4>\xb9\xef\x06\xd0\xc2Y\x04\x03\x19\x94~\x8czh\x12\xaa\x02S\u0609\t\r\x97/22<\x00\x00\u07d4>\u030e\x16h\xdd\xe9\x95\xdcW\x0f\xe4\x14\xf4B\x11\xc54\xa6\x15\x89lk\x93[\x8b\xbd@\x00\x00\u07d4>\u03752\xe3\x97W\x96b\xb2\xa4aA\u73c25\x93j_\x89\x03\x9f\xba\xe8\xd0B\xdd\x00\x00\u07d4>\xeeo\x1e\x966\vv\x89\xb3\x06\x9a\xda\xf9\xaf\x8e\xb6\f\u404965\u026d\xc5\u07a0\x00\x00\xe0\x94?\b\u066d\x89O\x81>\x8e!H\xc1`\xd2K5:\x8et\xb0\x8a\f\xb4\x9bD\xba`-
\x80\x00\x00\u07d4?\f\x83\xaa\xc5qybsN\\\xea\xea\xec\u04db(\xad\x06\xbe\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94?\x10\x80\x02\x82\u0477\xdd\u01cf\xa9-
\x820\aN\x1b\xf6\xae\xae\x8a\x01\n\xfc\x1a\xde;N\xd4\x00\x00\u07d4?\x123qO
M\xe9\xdeN\xe9m\a;6\x8d\x81\x97\x98\x9f\x89\x02\x17\xc4\x10t\xe6\xbb\x00\x00\u07d4?\x17:\xa6\xed\xf4i\u0445\xe5\x9b\xd2j\xe4#k\x92\xb4\xd8\xe1\x89\x11X\xe4`\x91=\x00\x00\x00\u07d4?\x1b\xc4
\xc5<\x00,\x9e\x90\x03|D\xfej\x8e\xf4\xdd\xc9b\x89\t`\xdbwh\x1e\x94\x00\x00\u07d4?#a\b\xee\xc7\"\x89\xba\u00e6\\\u0483\xf9^\x04\x1d\x14L\x8964\xbf9\xab\x98x\x80\x00\u07d4?-
\xa0\x93\xbb\x16\xeb\x06O\x8b\xfa\x9e0\xb9)\xd1_\x8e\x1cL\x89lk\x93[\x8b\xbd@\x00\x00\u07d4?-\xd5]\xb7\xea\xb0\xeb\xeee\xb3>\xd8
,\x1e\x99.\x95\x8b\x89,s\xc97t,P\x00\x00\u07d4?/8\x14\x91y|\xc5\xc0\u0502\x96\xc1O\xd0\xcd\x00\xcd\xfa-
\x89+\x95\xbd\xcc9\xb6\x10\x00\x00\u07d4?0\u04fc\x9f`\"2\xbcrB\x88\xcaF\xcd\v\a\x88\xf7\x15\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4?<\x8ea\xe5`L\xef\x06\x05\xd46\xdd\"\xac\u0346\"\x17\xfc\x89Hz\x9a0E9D\x00\x00\u07d4??F\xb7\\\xab\xe3{\xfa\u0307`(\x1fCA\xca\u007fF=\x89
\xacD\x825\xfa\xe8\x80\x00\u07d4?G)c\x19x\x83\xbb\xdaZ\x9b}\xfc\xb2-
\xb1\x14@\xad1\x89\x1a\x19d<\xb1\xef\xf0\x80\x00\u07d4?L\xd19\x9f\x8a4\xed\u06da\x17\xa4q\xfc\x92+Xp\xaa\xfc\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4?U\x1b\xa9<\xd5F\x93\xc1\x83\xfb\x9a\xd6\re\xe1`\x96s\u0249lk\x93[\x8b\xbd@\x00\x00\xe0\x94?bzv\x9ej\x95\x0e\xb8p\x17\xa7\u035c\xa2\bq\x13h1\x8a\x02\ub3b1\xa1r\u0738\x00\x00\u07d4?m\xd3e\x0e\xe4(\u0737u\x95S\xb0\x17\xa9j\x94(j\u0249Hz\x9a0E9D\x00\x00\u07d4?tr7\x80o\xed?\x82\x8ahR\xeb\bg\xf7\x90'\xaf\x89\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4?u\xaea\xcc\x1d\x80Be;[\xae\xc4D>\x05\x1c^z\xbd\x89\x05-
T(\x04\xf1\xce\x00\x00\u07d4?\xb7\u0457\xb3\xbaO\xe0E\xef\xc2=P\xa1E\x85\xf5X\u0672\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94?\xbc\x1eE\x18\xd74\x00\xc6\xd0F5\x949\xfbh\xea\x1aI\xf4\x8a\x03y\v\xb8U\x13v@\x00\x00\u07d4?\xbe\xd6\xe7\xe0\u029c\x84\xfb\xe9\xeb\u03ddN\xf9\xbbIB\x81e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4?\u043bGy\x8c\xf4L\u07feM3=\xe67\xdfJ\x00\xe4\\\x89\x05lUy\xf7\"\x14\x00\x00\xe0\x94?\xe4\x0f\xbd\x91\x9a\xad(\x18\xdf\x01\xeeM\xf4lF\x84*\xc59\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4?\xe8\x01\xe6\x135\xc5\x14\r\xc7\xed\xa2\xefR\x04F

\nP\x120\x89lk\x93[\x8b\xbd@\x00\x00\u07d4?\xf86\xb6\xf5{\x90\x1bD\f0\xe4\xdb\xd0e\xcf7\xd3\u050c\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4?\xfc\xb8p\xd4\x02=%]Qg\u0625\a\xce\xfc6kh\xba\x89#4<CT\u04ac\x00\x00\u0794@\x13T\xa2\x97\x95/\xa9r\xad8<\xa0z\n(\x11\xd7Jq\x88\xc2I\xfd\xd3'x\x00\x00\u07d4@0\xa9%pk,\x10\x1c\x8c\\\xb9\xbd\x05\xfb\xb4\xf6u\x9b\x18\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94@1E\xcbJ\xe7H\x9f\u0310\u0358\\m\u01c2\xb3\xccND\x8a\x01E?\xf3\x87\xb2|\xac\x00\x00\u07d4@2
`\n6\xf7?$\xe1\x90\xd1\xed\xb2\xd6\x1b\xe3\xf4\x13T\x89\x10z\xd8\xf5V\xc6\xc0\x00\x00\u07d4@9\xbdP\xa2\xbd\xe1_\xfe7\x19\x1fA\x03\x90\x96*+\x88\x86\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4@<d\x89ju\xca\xd8\x16\xa9\x10^\x18\u062a[\xf8\x0f#\x8e\x895e\x9e\xf9?\x0f\xc4\x00\x00\xe0\x94@=S\xcfb\x0f\t\"\xb4\x17\x84\x8d\xee\x96\xc1\x90\xb5\xbc\x82q\x8a\x02\x15\xf85\xbcv\x9d\xa8\x00\x00\u07d4@A\x00\xdbL]\x0e\xecUx#\xb5\x83Cu\x8b\xcc,\x80\x83\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4@A7K\x0f\xee\xf4y.K3i\x1f\xb8h\x97\xa4\xffV\f\x89\x13\xc9d~%\xa9\x94\x00\x00\u07d4@F}\x80\xe7L5@{}\xb5\x17\x89#F\x15\xfe\xa6h\x18\x89\x15\b\x94\xe8I\xb3\x90\x00\x00\u07d4@XR\x00h:@9\x017)\x12\xa8\x984\xaa\u0735_\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4@X\x80\x88\x16\xfd\xaa:_\u024e\xd4|\xfa\xe6\xc1\x83\x15B.\x89\n\xd4\xc81j\v\f\x00\x00\u07d4@_Yk\x94\xb9G4L\x03<\xe2\u073f\xf1.%\xb7\x97\x84\x89lk\x93[\x8b\xbd@\x00\x00\u07d4@c\x00$\xbd,X\xd2H\xed\xd8FV\x17\xb2\xbf\x16G\xda\x0e\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94@e#`\xd6qm\xc5\\\xf9\xaa\xb2\x1f4\x82\xf8\x16\xcc,\xbd\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94@r\x95\xeb\xd9KH&\x9c-
V\x9c\x9b\x9a\xf9\xaa\x05\xe8>^\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4@s\xfaI\xb8q\x17\u02d0\x8c\xf1\xabQ-
\xa7T\xa92\xd4w\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4@\x8ai\xa4\a\x15\xe1\xb3\x13\xe15N`\b\x00\xa1\xe6\xdc\x02\xa5\x89\x01\u7e11\u0312T\x00\x00\u07d4@\x9b\xd7P\x85\x82\x1c\x1d\xe7\f\xdc;\x11\xff\xc3\xd9#\xc7@\x10\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4@\x9dZ\x96.\xde\uefa1x\x01\x8c\x0f8\xb9\u0372\x13\xf2\x89\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4@\xa31\x19[\x97s%\u00aa(\xfa/B\xcb%\xec<%<\x89lk\x93[\x8b\xbd@\x00\x00\u07d4@\xa7\xf7(g\xa7\u0706w\v\x16+uW\xa44\xedP\xcc\xe9\x8965\u026d\xc5\u07a0\x00\x00\u07d4@\xab\n>\x83\xd0\u022c\x93f\x91\x05 \xea\xb1w+\xac;\x1a\x894\xf1\f-
\xc0^|\x00\x00\u07d4@\xabf\xfe!>\xa5l:\xfb\x12\xc7[\xe3?\x8e2\xfd\b]\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94@\xadt\xbc\v\xce*E\xe5/6\xc3\u07bb\x1b:\xda\x1bv\x19\x8a\x01p\x16-
\xe1\t\xc6X\x00\x00\u07d4@\u03c9\x05\x91\xea\u484f\x81*)T\xcb)_c3'\xe6\x89\x02\x9b\xf76\xfcY\x1a\x00\x00\u07d4@\u03d0\xef[v\x8c]\xa5\x85\x00,\xcb\xe6avP\xd8\xe87\x8963\x03\"\xd5#\x8c\x00\x00\xe0\x94@\xd4]\x9dv%\xd1QV\xc92\xb7q\xca{\x05'\x13\tX\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4@\xdb\x1b\xa5\x85\xce4S\x1e\xde\xc5IHI9\x13\x81\xe6\xcc\u04c9a\t=|,m8\x00\x00\xe0\x94@\xdfI^\xcf?\x8bL\xef*l\x18\x99W$\x8f\u813c+\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4@\xe0\xdb\xf3\xef\uf404\xea\x1c\xd7\xe5\x03\xf4\v;J\x84C\xf6\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4@\xe2D\n\xe1B\u02006j\x12\xc6\xd4\x10/K\x844\xb6*\x8965\u026d\xc5\u07a0\x00\x00\u07d4@\xe3\u0083\xf7\xe2M\xe0A\f\x12\x1b\xee`\xa5`\u007f>)\xa6\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94@\xeaPD\xb2\x04\xb20v\xb1\xa5\x80;\xf1\xd3\f\x0f\x88\x87\x1a\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\xe0\x94@\xed\xdbD\x8di\x0e\xd7.\x05\xc2%\xd3O\xc85\x0f\xa1\xe4\u014a\x01{x\x83\xc0i\x16`\x00\x00\xe0\x94@\xf4\xf4\xc0ls,\xd3[\x11\x9b\x89;\x12~}\x9

d\aq\xe4\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4A\x01\x0f\u023a\xf8C}\x17\xa0Ci\x80\x9a\x16\x8a\x17\xcaV\xfb\x89\x05k\xc7^-
c\x10\x00\x00\u07d4A\x03)\x96q\xd4gc\x97\x8f\xa4\xaa\x19\xee4\xb1\xfc\x95'\x84\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4A\x03<\x1bm\x05\xe1\u0289\xb0\x94\x8f\xc6DS\xfb\xe8z\xb2^\x89Hz\x9a0E9D\x00\x00\u07d4A\t\x8a\x81E#\x17\xc1\x9e>\xef\v\xd1#\xbb\xe1x\xe9\xe9\u0289\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4A\x16\x10\xb1x\xd5a}\xfa\xb94\u0493\xf5\x12\xa9>\\\x10\xe1\x89\t79SM(h\x00\x00\u07d4A\x1c\x83\x1c\xc6\xf4O\x19e\xecWW\xabN[<\xa4\xcf\xfd\x1f\x89\x17\n\x0fP@\xe5\x04\x00\x00\xe0\x94A*h\xf6\xc6EU\x9c\xc9w\xfcId\x04z
\x1d\x1b\xb0\xe2\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4A?K\x02f\x9c\xcf\xf6\x80k\xc8&\xfc\xb7\xde\xca;\x0e\xa9\xbc\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4AE\x99\t.\x87\x9a\xe2Sr\xa8MsZ\xf5\xc4\xe5\x10\xcdm\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4AHV\x12\xd04F\xecL\x05\xe5$NV?\x1c\xba\xe0\xf1\x97\x894\x95tD\xb8@\xe8\x00\x00\u07d4A]\tj\xb0b\x93\x18?<\x03=%\xf6\xcfqx\xac;\u01c9\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4Af\xfc\b\u0285\xf7f\xfd\xe81F\x0e\x9d\xc9<\x0e!\xaal\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94Ag\x84\xaf`\x960\xb0p\u051a\x8b\xcd\x12#\\d(\xa4\b\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4Ag\xcdH\xe73A\x8e\x8f\x99\xff\xd14\x12\x1cJJ\xb2x\u0109\xc5S%\xcat\x15\xe0\x00\x00\u07d4Al\x86\xb7
\x83\xd1\xf8\x90}\x84\xef\xd2\xd2\u05c3\xdf\xfa>\xfb\x89lj\xccg\u05f1\xd4\x00\x00\u07d4AsA\x9d\\\x9fc)U\x1d\xc4\xd3\xd0\u03ac\x1bp\x1b\x86\x9e\x89\x04\xc5>\xcd\xc1\x8a`\x00\x00\u07d4At\xfa\x1b\xc1*;q\x83\u02eb\xb7z\vYU{\xa5\xf1\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4Axj\x10\xd4G\xf4\x84\xd32D\u0337\xfa\u034bB{[\x8c\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94Az<\u0454\x96S\nmB\x04\u00f5\xa1|\xe0\xf2\a\xb1\xa5\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4A~N&\x88\xb1\xfdf\xd8!R\x9eF\xedOB\xf8\xb3\xdb=\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94A\x9aq\xa3l\x11\xd1\x05\xe0\xf2\xae\xf5\xa3\xe5\x98\a\x8e\x85\xc8\v\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x94A\x9b\xdes\x16\xcc\x1e\u0495\u0205\xac\xe3B\u01db\xf7\xee3\xea\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4A\xa2\xf2\xe6\xec\xb8c\x94\xec\x0e3\x8c\x0f\xc9~\x9cU\x83\xde\u0489l\xee\x06\u077e\x15\xec\x00\x00\u07d4A\xa8\u0083\x00\x81\xb1\x02\xdfn\x011e|\a\xabc[T\u0389lj\xccg\u05f1\xd4\x00\x00\u07d4A\xa8\xe26\xa3\x0emc\xc1\xffdM\x13*\xa2\\\x89S~\x01\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4A\xa9\xa4\x04\xfc\x9f[\xfe\xe4\x8e\xc2e\xb1%#3\x8e)\xa8\xbf\x89\x15\b\x94\xe8I\xb3\x90\x00\x00\u07d4A\xad6\x9fu\x8f\xef8\xa1\x9a\xa3\x14\x93y\x83,\x81\x8e\xf2\xa0\x8966\x9e\xd7t}&\x00\x00\u07d4A\xb2\xd3O\xde\v\x10)&+Ar\xc8\x1c\x15\x90@[\x03\xae\x8965\u026d\xc5\u07a0\x00\x00\u07d4A\xb2\xdb\u05dd\u069b\x86Ojp0'T\x19\u00dd>\xfd;\x89\xadx\xeb\u016cb\x00\x00\x00\u07d4A\xc3\xc26u4\xd1;\xa2\xb3?\x18\\\xdb\xe6\xacC\xc2\xfa1\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4A\u02d8\x96D_p\xa1\n\x14!R\x96\xda\xf6\x14\xe3,\xf4\u0549g\x8a\x93
b\xe4\x18\x00\x00\u07d4A\xcey\x95\t5\xcf\xf5[\xf7\x8eL\xce\xc2\xfec\x17\x85\u06d5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4A\u04f71\xa3&\xe7hX\xba\xa5\xf4\xbd\x89\xb5{6\x93#C\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\xe0\x94A\xe4\xa2\x02u\xe3\x9b\xdc\xef\xebe\\\x03\"tKvQ@\u008a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4A\xed-
\x8ep\x81H,\x91\x9f\xc2=\x8f\x00\x91\xb3\xc8,F\x85\x89F:\x1ev[\u05ca\x00\x00\xe0\x94A\xf2~tK\u049d\xe2\xb0Y\x8f\x02\xa0\xbb\x9f\x98\xe6\x81\ua90a\x01\xa4\xab\xa2%\xc2\a@\x00\x00\u07d4A\xf4\x89\xa1\xect{\u009c>_\x9d\x8d\xb9xw\xd4\u0474\xe9\x89\a?u\u0460\x85\xba\x00\x00\u07

d4B\x0f\xb8n}+Q@\x1f\xc5\xe8\xc7
\x15\xde\xcbN\xf8\xfc.\x8965\u026d\xc5\u07a0\x00\x00\u07d4B\x16\x84\xba\xa9\xc0\xb4\xb5\xf5
S8\xe6\xf6\xe7\xc8\xe1F\xd4\x1c\xb7\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4B9\x96Y\xac\xa6
\xa5\xa8c\xea\"E\xc93\xfe\x9a5\xb7\x88\x0e\x89n\xce2\xc2l\x82p\x00\x00\xe0\x94B;\xcaG\xab\xc
0\fpW\xe3\xad4\xfc\xa6>7_\xbd\x8bJ\x8a\x03\xcf\xc8.7\xe9\xa7@\x00\x00\u07d4B<1\a\xf4\xba\xc
eANI\x9cd9\nQ\xf7F\x15\xca^\x89lk\x93[\x8b\xbd@\x00\x00\u07d4B<\xc4YL\xf4\xab\xb66\x8d\xe
5\x9f\u04b1#\a4a!C\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94BD\xf13\x11X\xb9\xce&\xbb\xe0\xb9#k\
x92\x03\xca5\x144\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u0794BQw\xebt\xad\n\x9d\x9aWR\
"\x81G\xeemcV\xa6\u6239\x8b\xc8)\xa6\xf9\x00\x00\u07d4BW%\xc0\xf0\x8f\b\x11\xf5\xf0\x06\xee
\xc9\x1c\\\\\x12k\x12\xae\x89\b!\xab\rD\x14\x98\x00\x00\xe0\x94BX\xfdf/\xc4\xce2\x95\xf0\xd4\xed
\x8f{\xb1D\x96\x00\xa0\xa9\x8a\x01lE.\xd6\b\x8a\xd8\x00\x00\xe0\x94B\\\x18\x16\x86\x8fww\xcc+
\xa6\xc6\u048c\x9e\x1eylR\xb3\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4B\\3\x8a\x13%\x
e3\xa1W\x8e\xfa)\x9eW\u0646\xebGO\x81\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94BbY\xb0\xa7Vp\
x1a\x8bf5(R!V\xc0(\x8f\x0f$\x8a\x02\x18\xae\x19k\x8dO0\x00\x00\u07d4Bm\x15\xf4\a\xa0\x115\x
b1:kr\xf8\xf2R\v51\xe3\x02\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4Box\xf7\r\xb2Y\xac\x854\x1
4[)4\xf4\xef\x10\x98\xb5\u0609\x13\x84\x00\xec\xa3d\xa0\x00\x00\u07d4Bs-
\x8e\xf4\x9f\xfd\xa0K\x19x\x0f\xd3\xc1\x84i\xfb7A\x06\x89\x17\v\x00\xe5\u4a7e\x00\x00\u07d4Bt\
x17\xbd\x16\xb1\xb3\xd2-\xbb\x90-
\x8f\x96W\x01o$\xa6\x1c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Btj\xee\xa1O'\xbe\xff\f\r\xa6BS\xf1\x
e7\x97\x18\x90\xa0\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4B{F*\xb8NP\x91\xf4\x8aF\xeb\f\u071
2\xdd\xcb&\xe0x\x89lk\x93[\x8b\xbd@\x00\x00\u07d4B~GQ\u00fa\xbex\xcf\xf8\x83\b\x86\xfe\xbc\
x10\xf9\x90\x8dt\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94B~\xc6h\xac\x94\x04\xe8\x95\u0306\x15\
x11\xd1b\nI\x12\xbe\x98\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4B\x80\xa5\x8f\x8b\xb1\v\x94@\
u0794\xf4+OY!
\x82\x01\x91\x89lk\x93[\x8b\xbd@\x00\x00\u07d4B\x8a\x1e\xe0\xed3\x1dyR\u033e\x1cyt\xb2\x85
+\u0453\x8a\x89w\xb7JN\x8d\xe5e\x00\x00\u0794B\x9c\x06\xb4\x87\xe8Tj\xbd\xfc\x95\x8a%\xa3
\xf0\xfb\xa5?o\x00\x88\xbbdJ\xf5B\x19\x80\x00\xe0\x94B\xa9\x8b\xf1`'\xceX\x9cN\xd2\xc9X1\xe2r
B\x05\x06N\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4B\xc6\xed\xc5\x15\xd3UW\x80\x8d\
x13\xcdD\xdc\xc4@\v%\x04\xe4\x89\n\xba\x14\u015b\xa72\x00\x00\u07d4B\xce\xcf\u0492\x10y\x
c2\xd7\xdf?\b\xb0z\xa3\xbe\xee^!\x9a\x8965\u026d\xc5\u07a0\x00\x00\u07d4B\u04669\x9b0\x16\
xa8Y\u007f\x8bd\t'\xb8\xaf\xbc\xe4\xb2\x15\x89\xa1\x8b\xce\xc3H\x88\x10\x00\x00\u07d4B\xd3I
@\xed\xd2\xe7\x00]F\xe2\x18\x8eL\xfe\u0383\x11\xd7M\x89\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u0
7d4B\u04e5\xa9\x01\xf2\xf6\xbd\x93V\xf1\x12\xa7\x01\x80\xe5\xa1U\v`\x892$\xf4'#\xd4T\x00\x00\
u07d4B\u05b2c\xd9\xe9\xf4\x11lA\x14$\xfc\x99Ux<v00\x89lk\x93[\x8b\xbd@\x00\x00\u07d4B\xdb
\v\x90%Y\xe0@\x87\xdd\\D\x1b\xc7a\x194\x18K\x89\x89m3\xb1}%:b\x00\x00\u07d4B\xdd\xd0\x1
4\xdcR\xbf\xbc\xc5U2Z@\xb5\x16\xf4\x86j\x1d\u04c9lk\x93[\x8b\xbd@\x00\x00\u07d4C\x19&?u
@,\vS%\xf2c\xbeJP\x80e\x10\x87\xf0\x895K\x0f\x14c\x1b\xab\x00\x00\u07d4C\x1f,\x19\xe3\x16\x
b0D\xa4\xb3\xe6\x1a\fo\xf8\xc1\x04\xa1\xa1/\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94C\"}e3Ni\x1
c\xf21\xb4\xa4\xe1\xd39\xb9]Y\x8a\xfb\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94C(\t\xa
29\x0f\a\xc6e\x92\x1f\xf3}T}\x12\xf1\u0256j\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4C)\xfc\t1\xcb\
xeb\x038\x80\xfeL\x93\x98\xcaE\xb0\xe2\xd1\x1a\x89lq qm3h\x00\x00\u07d4C-
\x88K\u059d\xb1\xac\xc0\u061cd\xad\xe4\xcbO\u00e8\x8bz\x89\x86\x9a\x8c\x10\x80\x8e\xec\x00

\x00\u07d4C1\xab7G\xd3W
\xa9\xd8\xca%\x16\\\u0485\xac\u053d\xa8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4C:;h\xe5k\r\xf1\x86+\x90Xk\xbd9\xc8@\xff\x196\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94C>;\xa1\xc5\x1b\x81\x0f\xc4g\u057aM\xeaB\xf7\xa9\x88^i\x8a\bxg\x83&\xea\xc9\x00\x00\x00\xe0\x94C>\xb9J3\x90\x86\xed\x12\u067d\xe9\xcd\x1dE\x86\x03\xc9}\u058a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4CI\"Zb\xf7\n\xeaH\n\x02\x99\x15\xa0\x1eSy\xe6O\xa5\x89\x8c\xd6~#4\xc0\xd8\x00\x00\u07d4CT\"\x1eb\xdc\t\xe6@d6\x16:\x18^\xf0m\x11J\x81\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94CTC\xb8\x1d\xfd\xb9\xbd\x8cg\x87\xbc%\x18\xe2\xd4~W\xc1_\x8a\x01C\x8d\x93\x97\x88\x1e\xf2\x00\x00\u07d4Ca\u0504o\xaf\xb3w\xb6\xc0\xeeI\xa5\x96\xa7\x8d\xdf5\x16\xa3\x89\xc2\x12z\xf8X\xdap\x00\x00\xe0\x94Cd0\x9a\x9f\xa0p\x95`\x0fy\xed\xc6Q
\xcd\xcd#\xdcd\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4Cg\xaeK\f\xe9d\xf4\xa5J\xfdK\\6\x84\x96\xdb\x16\x9e\x9a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Ct\x89(\xe8\xc3\xecD6\xa1\u0412\xfb\xe4:\xc7I\xbe\x12Q\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4Cv{\xf7\xfd*\xf9[r\xe91-
\xa9D<\xb1h\x8eCC\x89\x10CV\x1a\x88)0\x00\x00\xe0\x94Cy\x838\x8a\xb5\x9aO\xfc!_\x8e\x82iF\x10)\xc3\xf1\xc1\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4C\x89\x8cI\xa3MP\x9b\xfe\xd4\xf7`A\xee\x91\xca\xf3\xaaj\xa5\x89\x10CV\x1a\x88)0\x00\x00\u07d4C\x8c/T\xff\x8eb\x9b\xab6\xb1D+v\v\x12\xa8\x8f\x02\xae\x89lk\x93[\x8b\xbd@\x00\x00\u07d4C\x98b\x8e\xa6c-
9>\x92\x9c\xbd\x92\x84d\xc5h\xaaJ\f\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4C\x9d//Q\x10\xa4\u054b\x17W\x93P\x15@\x87@\xfe\xc7\xf8\x89\u03e5\xc5\x15\x0fL\x88\x80\x00\u07d4C\x9d\xee?vy\xff\x100s?\x93@\xc0\x96hkI9\v\x89lk\x93[\x8b\xbd@\x00\x00\u07d4C\xb0y\xba\xf0ry\x99\xe6k\xf7C\u057c\xbfwl;\t\"\x89lk\x93[\x8b\xbd@\x00\x00\u07d4C\xbc-
M\xdc\xd6X;\xe2\u01fc\tK(\xfbr\xe6+\xa8;\x89lk\x93[\x8b\xbd@\x00\x00\u07d4C\xc7\xeb\u0173\xe7\xaf\x16\xf4}\xc5az\xb1\x0e\x0f9\xb4\xaf\xbb\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4C\u02d6R\x81\x8coMg\x96\xb0\xe8\x94\t0ly\xdbcI\x89lk\x93[\x8b\xbd@\x00\x00\u07d4C\xcc\b\xd0s*\xa5\x8a\xde\xf7a\x9b\xedFU\x8a\xd7wAs\x89\xf0\xe7\u0730\x12*\x8f\x00\x00\xe0\x94C\u0567\x1c\xe8\xb8\xf8\xae\x02\xb2\xea\xf8\xea\xf2\xca(@\xb9?\xb6\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u0794C\xdb\u007f\xf9Z\bm(\ubff8/\xb8\xfb_#\n^\xbc\u0348\xdfn\xb0\xb2\xd3\xca\x00\x00\u07d4C\xe7\xec\x84cX\xd7\xd0\xf97\xad\x1c5\v\xa0i\u05ffr\xbf\x89\x06p\xaeb\x92\x14h\x00\x00\u07d4C\xf1o\x1eu\xc3\xc0j\x94x\xe8\u0157\xa4\n<\xb0\xbf\x04\u0309\x9d\xf7\u07e8\xf7`H\x00\x00\u07d4C\xf4p\xede\x9e)\x91\xc3u\x95~]\xde\u017d\x1d8\"1\x89\x05k\xc7^-
c\x10\x00\x00\u07d4C\xf7\xe8n8\x1e\xc5\x1e\u0110m\x14v\u02e9z=\xb5\x84\xe4\x8965\u026d\xc5\u07a0\x00\x00\u07d4C\xff8t>\xd0\xcdC0\x8c\x06e\t\u030e~r\xc8b\xaa\x89i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94C\xff\x88S\xe9\x8e\xd8@k\x95\x00\n\u0684\x83b\u05a09*\x8a\x04\xae\v\x1cM.\x84\xd0\x00\x00\u07d4D\t\x88f\xa6\x9bh\xc0\xb6\xbc\x16\x82)\xb9`5\x87\x05\x89g\x89\n1\x06+\xee\xedp\x00\x00\u07d4D\x19\xaca\x8d]\xea|\xdc`w
o\xb0}\xbd\xd7\x1c\x17\x02\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4D\x1aR\x00\x16a\xfa\xc7\x18\xb2\u05f3Q\xb7\xc6\xfbR\x1az\xfd\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94D\x1a\u0282c\x13$\xac\xbf\xa2F\x8b\xda2[\xbdxG{\xbf\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4D\x1f7\xe8\xa0)\xfd\x02H/(\x9cI\xb5\xd0m\x00\xe4\b\xa4\x89\x12\x11\xec\xb5m\x13H\x80\x00\u07d4D
\xaa5F[\xe6\x17\xad$\x98\xf3p\xde\n<\xc4\xd20\xaf\x89lk\x93[\x8b\xbd@\x00\x00\u07d4D#/\xf6m\xda\xd1\xfd\x84\x12f8\x006\xaf\xd7\xcf}\u007fB\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4D%\rGn\x06$\x84\xe9\b\n9g\xbf:Js*\xd7?\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07

d4D)\xa2\x9f\xee\x19\x84Pg,\f\x1d\a1b%\v\xecdt\x896*\xaf\x82\x02\xf2P\x00\x00\u07d4D5RS\xb2wH\xe3\xf3O\xe9\xca\xe1\xfbq\x8c\x8f$\x95)\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4D8\xe8\x80\xcb'f\xb0\xc1\u03ae\xc9\xd2A\x8f\u03b9R\xa0D\x89\a?\xa0s\x90?\b\x00\x00\u07d4DL\xafy\xb7\x138\ue6a7\xc73\xb0*\u02a7\xdc\x02YH\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4D\\\xb8\xde^=\xf5
\xb4\x99\xef\u0240\xf5+\xff@\xf5\\v\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94Dj\x809\xce\u03dd\xceHy\xcb\xca\xf3I;\xf5E\xa8\x86\x10\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4Dt)\x9d\x0e\xe0\x90\u0710x\x9a\x14\x86H\x9c=\rd^m\x8965\u026d\xc5\u07a0\x00\x00\u07d4D\x8b\xf4\x10\xad\x9b\xbc/\xec\xc4P\x8d\x87\xa7\xfc.K\x85a\xad\x89\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4D\x90\x1e\r\x0e\b\xac=^\x95\xb8\xec\x9d^\x0f\xf5\xf1.\x03\x93\x89\x16\xa1\xf9\xf5\xfd}\x96\x00\x00\xe0\x94D\x93\x12<\x02\x1e\xce;3\xb1\xa4R\xc9&\x8d\xe1@\a\xf9\u04ca\x01je\x02\xf1Z\x1eT\x00\x00\xe0\x94D\x9a\xc4\xfb\xe3\x83\xe3g8\x85^6JW\xf4q\xb2\xbf\xa11\x8a)\xb7d2\xb9DQ
\x00\x00\u07d4D\xa0\x1f\xb0J\xc0\xdb,\xce]\xbe(\x1e\x1cF\xe2\x8b9\xd8x\x89lj\xccg\u05f1\xd4\x00\x00\u07d4D\xa6=\x18BE\x87\xb9\xb3\a\xbf\xc3\xc3d\xae\x10\xcd\x04\xc7\x13\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94D\xa8\x98\x9e20\x81!\xf7$f\x97\x8d\xb3\x95\xd1\xf7I:K\x8a\x01\x88P)\x9fB\xb0j\x00\x00\u07d4D\xc1\x11\v\x18\x87\x0e\xc8\x11x\xd9=!X8\xc5Q\u050ed\x89\n\xd6\xf9\x85\x93\xbd\x8f\x00\x00\u07d4D\xc1Ge\x12|\xde\x11\xfa\xb4l],\xf4\u0532\x89\x00#\xfd\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94D\xc5N\xaa\x8a\xc9@\xf9\xe8\x0f\x1et\xe8/\xc1O\x16v\x85j\x8a\x01\xab,\xf7\xc9\xf8~ \x00\x00\u07d4D\xcdwSZ\x89?\xa7\xc4\xd5\xeb:$\x0ey\u0419\xa7--
\x89,s\xc97t,P\x00\x00\u07d4D\u07faP\xb8)\xbe\xcc_O\x14\u0470J\xab3
\xa2\x95\xe5\x8965\u026d\xc5\u07a0\x00\x00\u07d4D\xe2\xfd\xc6y\xe6\xbe\xe0\x1e\x93\xefJ:\xb1\xbc\xce\x01*\xbc|\x89\x16=\x19I\x00\xc5E\x80\x00\xe0\x94D\xf6/*\xaa\xbc)\xad:k\x04\xe1\xffo\x9c\xe4R\xd1\xc1@\x8a\x03\x99\x92d\x8a#\u0220\x00\x00\u07d4D\xff\xf3{\xe0\x1a8\x88\u04f8\xb8\u1200\xa7\xdd\xef\xee\xea\u04c9\x0e\f[\xfc}\xae\x9a\x80\x00\u07d4E\x06\xfe\x19\xfaK\x00k\xaa9\x84R\x9d\x85\x16\xdb++P\xab\x89lk\x93[\x8b\xbd@\x00\x00\u07d4E\x1b6\x99G[\xed]y\x05\xf8\x90Z\xa3Eo\x1e\u05c8\xfc\x89\x8a\xc7#\x04\x89\xe8\x00\x00\x00\u0794E\x1bpp%\x9b\u06e2q\x00\xe3n#B\x8aS\xdf\xe3\x04\u9239\x8b\xc8)\xa6\xf9\x00\x00\u07d4E'+\x8fb\xe9\xf9\xfa\x8c\xe0D
\u1ba3\xeb\xa9hn\xac\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94E+d\u06ce\xf7\xd6\u07c7\u01c8c\x9c\"\x90\xbe\x84\x82\xd5u\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4E>5\x9a3\x97\x94LZ'Z\xb1\xa2\xf7\n^Z?i\x89\x89\r\x02\xabHl\xed\xc0\x00\x00\u07d4EI\xb1Yy%_~e\xe9\x9b\rV\x04\u06d8\xdf\xca\u023f\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4EKa\xb3D\xc0\xef\x96Qy#\x81U\xf2w\u00c2\x9d\v8\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94EO\x01A\xd7!\xd3<\xbd\xc4\x10\x18\xbd\x01\x11\x9a\xa4xH\x18\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4ES3\x90\xe3@\xfe\r\xe3\xb3\xcf_\xb9\xfc\x8e\xa5R\xe2\x9eb\x89O%\x91\xf8\x96\xa6P\x00\x00\u07d4ES\x96\xa4\xbb\u067a\u8bdf\xb7\xc4\xd6MG\x1d\xb9\xc2E\x05\x89\b\xbaR\xe6\xfcE\xe4\x00\x00\u07d4E[\x92\x96\x92\x1at\xd1\xfcAa\u007fC\xb80>o>\xd7l\x89\u3bb5sr@\xa0\x00\x00\u07d4E\\\xb8\xee9\xff\xbcu#1\xe5\xae\xfcX\x8e\xf0\xeeY4T\x8965F:x\r\xef\x80\x00\u07d4Ej\u0b24\x8e\xbc\xfa\xe1f\x06\x02PR_c\x96^v\x0f\x89\x10CV\x1a\x88)0\x00\x00\u07d4Eo\x8dtf\x82\xb2$g\x93l\x06M\x1b6\x8c|\x05\xb1v\x89\u0213\u041c\x8fQP\x00\x00\u07d4Ep)\xc4i\xc4T\x8d\x16\x8c\xec>e\x87.D(\xd4+g\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Eq\xdeg+\x99\x04\xba\xd8t6\x92\xc2\x1cO\xdc\xeaL.\x01\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4Ex\x1b\xbew\x14\xa1\xc8\xf7;\x1cty!\xdfO\x84'\x8bp\x89lk\x93[\x8b\xbd@

\x00\x00\u07d4E{\xce\xf3}\xd3\xd6\v-
\xd0\x19\xe3\xfea\xd4k?\x1erR\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94E\x8e<\u025e\x94xD\
xa1\x8ejB\x91\x8f\xef~\u007f_^\xb3\x8a\a\xb5?y\xe8\x88\xda\xc0\x00\x00\u07d4E\x93\x93\xd6:\x
06>\xf3r\x1e\x16\xbd\x9f\xdeE\ue77dw\xfb\x89j\xba\u05a3\xc1S\x05\x00\x00\u07d4E\xa5p\xdc\xc
2\t\f\x86\xa6\xb3\xea)\xa6\bc\xdd\xe4\x1f\x13\xb5\x89\f\x9a\x95\xee)\x86R\x00\x00\u07d4E\xa8
\xa0g/\x17\xdct\xa0\x81\x12\xbcd?\xd1\x16w6\u00c9\n\xd6\xc4;(\x15\xed\x80\x00\u07d4E\xb4q\x
05\xfeB\xc4q-
\xcen*!\xc0[\xff\xd5\xeaG\xa9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4E\xbb\x82\x96R\u063f\xb5\x8b\
x85'\xf0\xec\xb6!\u009e!.\u00c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94E\xc0\u045f\v\x8e\x05O\x9e\x8
986\xd5\xec\xaey\x01\xaf(\x12\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4E\xc4\xec\xb4\xee\x89\x1e\xa9\x84\xa7\xc5\xce\xfd\x8d\xfb\x001\v(P\x89kV\x05\x
15\x82\xa9p\x00\x00\u07d4E\u028d\x95f\b\xf9\xe0\n/\x99t\x02\x86@\x88\x84ef\x8f\x89lk\x93[\x8b
\xbd@\x00\x00\xe0\x94E\u0298b\x00;N@\xa3\x17\x1f\xb5\xca\xfa\x90(\xca\xc8\xde\x19\x8a\x02\
ub3b1\xa1r\u0738\x00\x00\u07d4E\xd1\xc9\xee\xdf|\xabA\xa7y\x05{y9_T(\xd8\x05(\x89lk\x93[\x8b
\xbd@\x00\x00\u07d4E\u0535M7\xa8\xcfY\x98!#_\x06/\xa9\xd1p\xed\u8909\x11\x90g;_\u0690\x0
0\x00\xe0\x94E\xdb\x03\xbc\xcf\u05a5\xf4\xd0&k\x82\xa2*6\x87\x92\xc7}\x83\x8a\x01\xb1\xaeMn.
\xf5\x00\x00\x00\u07d4E\xe3\xa9>r\x14J\u0686\f\xbcV\xff\x85\x14Z\xda8\xc6\u0689WG=\x05\u06
ba\xe8\x00\x00\u07d4E\u6378\u06fa\xba_\xc2\xcb3|b\xbc\xd0\xd6\x1b\x05\x91\x89\x89lk\x93[\x8
b\xbd@\x00\x00\xe0\x94E\u6379L}\n\xb7\xacA\x85zq\xd6qG\x87\x0fNq\x8aT\xb4\v\x1f\x85+\xda\
x00\x00\x00\u07d4E\xf4\xfc`\xf0\x8e\xac\xa1\x05\x98\xf03c)\x80\x1e<\x92\xcbF\x89\n\u05ce\xbcZ
\xc6 \x00\x00\u07d4F\rSU\xb2\xce\xebnb\x10}\x81\xe5\x12p\xb2k\xf4V
\x89l\xb7\xe7Hg\xd5\xe6\x00\x00\xe0\x94F\"O2\xf4\xec\xe5\u0206p\x90\xd4@\x9dU\xe5\v\x18C-
\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4F'\xc6\x06\x84&q\xab\u0782\x95\xee]\xd9L\u007fT\x
954\xf4\x89\x0f\x89_\xbd\x872\xf4\x00\x00\u07d4F+g\x8bQ\xb5\x84\xf3\xedz\xda\a\v\\\u065c\v\xf
7\xb8\u007f\x89\x05k\xc7^-
c\x10\x00\x00\u07d4FM\x9c\x89\xcc\xe4\x84\xdf\x00\x02w\x19\x8e\xd8\a_\xa65r\u0449\x01\x15\x
8eF\t\x13\xd0\x00\x00\u07d4FPNj!Z\xc8;\xcc\xf9V\xbe\xfc\x82\xabZg\x93q\u0209\x1c!(\x05\u00b4
\xa5\x00\x00\xe0\x94FQ\xdcB\x0e\b\xc3);'\xd2Ix\x90\xebP\":\xe2\xf4\x8a\x04<3\xc1\x93ud\x80\x0
0\x00\u07d4FS\x1e\x8b\x1b\xde\t\u007f\u07c4\x9dm\x11\x98\x85`\x8a\x00\x8d\xf7\x89\n\u05ce\x
bcZ\xc6
\x00\x00\u07d4Fb\x92\xf0\xe8\rC\xa7\x87t'u\x90\xa9\xebE\x96\x12\x14\xf4\x894\x95tD\xb8@\xe8\
x00\x00\xe0\x94Fb\xa1v^\xe9!\x84-
\u0708\x89\x8d\x1d\xc8bu\x97\xbd~\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4Fe\xe4s\x9
6\xc7\u06d7\xeb*\x03\xd9\bc\xd5\u053a1\x9a\x94\x89
\x86\xac5\x10R`\x00\x00\u07d4Fo\xdak\x9bX\xc5S'P0j\x10\xa2\xa8\xc7h\x10;\a\x89\n\xd6\xee\xd
d\x17\xcf;\x80\x00\u07d4Fq$\xae\u007fE/&\xb3\xd5t\xf6\b\x88\x94\xfa]\x1c\xfb;\x89\x92^\x06\xee\
xc9r\xb0\x00\x00\u0794Fr*6\xa0\x1e\x84\x1d\x03\xf7\x80\x93^\x91}\x85\u0566z\xbd\x88\xce\xc7o
\x0eqR\x00\x00\u07d4Fw\x9aVV\xff\x00\xd7>\xac:\xd0\u00cbl\x850\x94\xfb@\x89\f\x82S\xc9lj\xf0
\x00\x00\u07d4Fw\xb0N\x03C\xa3!1\xfdj\xbb9\xb1\xb6\x15k\xba=[\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4F}Y\x88$\x9ahaG\x16e\x98@\xed\n\xe6\xf6\xf4W\xbc\x89\x15\x01\xa4\x8c\xef\xdf\
xde\x00\x00\u07d4F~\x0e\xd5O;v\xae\x066\x17n\aB\b\x15\xa0!sn\x89lk\x93[\x8b\xbd@\x00\x00\u
07d4F~\xa1\x04E\x82~\xf1\xe5\x02\xda\xf7k\x92\x8a

\x9e\r@2\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94F\u007f\xbfAD\x16\x00u\u007f\xe1X0\xc8\xcd_O\xfb\xbb\xd5`\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94F\x93Xp\x932\xc8+\x88~\xbc\xdd\xd0\"\x0f\x8e\u06e7\u040a\x03\xa9\u057a\xa4\xab\xf1\xd0\x00\x00\u07d4F\x97\xba\xaf\x9c\xcb`?\xd3\x040h\x9dCTE\xe9\u024b\xf5\x89\n\xd2\x01\xa6yO\xf8\x00\x00\u07d4F\xa3\v\x8a\x80\x891!tE\xc3\xf5\xa9>\x88,\x03E\xb4&\x89\r\x8d\xb5\xeb\u05f2c\x80\x00\u07d4F\xa40\xa2\u0528\x94\xa0\u062a?\xea\xc6\x156\x14\x15\xc3\xf8\x1f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4F\xaaP\x18pg~\u007f\nPHv\xb4\xe8\x80\x1a\n\xd0\x1cF\x89+^:\xf1k\x18\x80\x00\x00\u07d4F\xbf\u0172\a\xeb
\x13\xe2\xe6\x0fw_\xec\xd7\x18\x10\u0159\f\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4F\xc1\xaa\"D\xb9\u0229W\u028f\xacC\x1b\x05\x95\xa3\xb8h$\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4F\xd8\x061(B\x03\xf6(\x8e\xcdNWX\xbb\x9dA\xd0]\xbe\x89lk\x93[\x8b\xbd@\x00\x00\u07d4G\n\xc5\xd1\xf3\xef\xe2\x8f8\x02\xaf\x92[W\x1ec\x86\x8b9}\x89lk\x93[\x8b\xbd@\x00\x00\u07d4G\x10\x10\xdaI/@\x18\x83;\b\x8d\x98r\x90\x1e\x06\x12\x91t\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4G\x12T\x02e\xcb\xee\u00c4p\"\u015f\x1b1\x8dC@\n\x9e\x89\xbd\xbcA\xe04\x8b0\x00\x00\xe0\x94G\x14\u03e4\xf4k\u05bdps}u\x87\x81\x97\xe0\x8f\x88\xe61\x8a\x02\u007f>\u07f3Nn@\x00\x00\u07d4G
H\xcc`\x9a\xeb$!e\uaa87\x05\x85\f\xf3\x12]\xe0\x8965\u026d\xc5\u07a0\x00\x00\u07d4G!\x92)\xe8\xcdVe\x9ae\u00a9C\xe2\u075a\x8fK\xfd\x89\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d4G7\xd0B\xdcj\xe7>\xc7:\xe2Qz\u03a2\xfd\xd9d\x87\u014965\u026d\xc5\u07a0\x00\x00\u07d4GAX\xa1\xa9\xdci<\x13?e\xe4{\\:\xe2\xf7s\xa8o\x89\n\xdaUGK\x814\x00\x00\u07d4GE\xab\x18\x1a6\xaa\x8c\xbf\"\x89\xd0\xc4Qe\xbc~\xbe#\x81\x89\x02\"\xc8\xeb?\xf6d\x00\x00\u07d4GPf\xf9\xad&eQ\x96\xd5SS'\xbb\xeb\x9by)\xcb\x04\x89\xa4\xccy\x95c\u00c0\x00\x00\xe0\x94GR!\x8eT\xdeB?\x86\xc0P\x193\x91z\xea\b\xc8\xfe\u054a\x04<3\xc1\x93ud\x80\x00\x00\u07d4GZa\x93W-JNY\u05fe\t\u02d6\r\u074cS\x0e/\x89$,\xf7\x8c\xdf\a\xff\x80\x00\u07d4Gd\x8b\xed\x01\xf3\xcd2I\bNc]\x14\u06a9\xe7\xec<\x8a\x89\n\x84Jt$\xd9\xc8\x00\x00\u07d4Gh\x84\x10\xff%\xd6T\xd7.\xb2\xbc\x06\xe4\xad$\xf83\xb0\x94\x89\b\xb2\x8da\xf3\u04ec\x00\x00\u07d4GkU\x99\b\x9a?\xb6\xf2\x9clr\xe4\x9b.G@\ua00d\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4Gs\x0f_\x8e\xbf\x89\xacr\xef\x80\xe4l\x12\x19P8\xec\xdcI\x89\xabM\xcf9\x9a:`\x00\x00\xe0\x94G{$\xee\u80deO\u045d\x12P\xbd\vfEyJa\u028a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4G\x81\xa1\nM\xf5\uef02\xf4\xcf\xe1\a\xba\x1d\x8av@\xbdf\x89a\t=|,m8\x00\x00\u07d4G\x88Z\xba\xbe\xdfM\x92\x8e\x1c<q\xd7\xca@\xd5c\xedY_\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4G\x8d\xc0\x9a\x13\x117|\t?\x9c\u022et\x11\x1fe\xf8/9\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94G\x8eRN\xf2\xa3\x81\xd7\f\x82X\x8a\x93\xcaz_\xa9\xd5\x1c\xbf\x8a5\xfa\x97\"o\x88\x99p\x00\x00\xe0\x94G\x92\x98\xa9\xde\x14~c\xa1\xc7\xd6\xd2\xfc\xe0\x89\xc7\xe6@\x83\xbd\x8a\x02\x1e\x19\xdd<<\ry\x80\x00\u07d4G\x9a\xbf-\xa4\u0547\x16\xfd\x97:\r\x13\xa7_S\x01P&\n\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4G\xa2\x81\xdf\xf6Ag\x19xU\xbfnp^\xb9\xf2\xce\xf62\xea\x8966\xc9yd6t\x00\x00\u07d4G\xbe\xb2\x0fu\x91\x00T*\xa9=A\x11\x8b2\x11\xd6d\x92\x0e\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94G\xc2G\xf5;\x9f\xbe\xb1{\xba\a\x03\xa0\f\x00\x9f\xdb\x0fn\xae\x8a\x04<3\xc1\x93ud\x80\x00\x00\u0794G\xc7\xe5\ufd0b:\xedK|n\x82KC_5}\xf4\xc7#\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4G\u03dc\xda\xf9/\u0259\xcc^\xfb\xb7
<a\xe4\xf1\xcd\xd4\u00c9\a\x1f\x8a\x93\xd0\x1eT\x00\x00\u07d4G\xd2\x0ej\xe4\xca\xd3\xf8)\xea\xc0~Z\xc9{f\xfd\xd5l\xf5\x8965\u026d\xc5\u07a0\x00\x00\u07d4G\u05d2\xa7Vw\x9a\xed\xf14>\x8

8\x83\xa6a\x9cl(\x11\x84\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94G\xe2]\xf8\x82%8\xa8Yk(\xc67\x89kM\x14<5\x1d\x8a\x11\v\xe9\xeb$\xb8\x81P\x00\x00\u07d4G\xf4ik\xd4b\xb2\r\xa0\x9f\xb8>\xd2\x03\x98\x18\xd7v%\xb3\x89\b\x13\xcaV\x90m4\x00\x00\u07d4G\xfe\xf5\x85\x84FRH\xa0\x81\r`F>\xe9>Zn\xe8\u04c9\x0fX\xcd>\x12i\x16\x00\x00\u07d4G\xffo\xebC!
`\xbb\x15\x03\u05e3\x97\xfc\b\xf4\xe7\x03R\x89lk\x93[\x8b\xbd@\x00\x00\u07d4G\xff\xf4,g\x85Q\xd1A\xebu\xa6\xee9\x81\x17\xdf>J\x8d\x89\x05k\xea\xe5\x1f\xd2\xd1\x00\x00\u07d4H\x01\x0e\xf3\xb8\xe9^?0\x8f0\xa8\xcb\u007fN\xb4\xbf`\xd9e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4H\n\xf5v\x00\x9c\xa77\x81\xb7\x0eC\xb9Y\x16\xa6\"\x03\xab\x892\x19r\xf4\b=\x87\x80\x00\u07d4H\x0f1\xb9\x891\x1eA$\u01a7F_ZD\tM6\xf9\u04097\x90\xbb\x85Q7d\x00\x00\xe0\x94H\x11\x15)j\xb7\xdbRI/\xf7\xb6G\xd63)\xfb\\\xbck\x8a\x03h\xc8b:\x8bM\x10\x00\x00\u07d4H\x1e:\x91\xbf\xdc/\x1c\x84(\xa0\x11\x9d\x03\xa4\x16\x01A~\x1c\x8965\u026d\xc5\u07a0\x00\x00\u07d4H(\xe4\xcb\xe3N\x15\x10\xaf\xb7,+\ueb0aE\x13\xea\xeb\u0649\u0556{\xe4\xfc?\x10\x00\x00\xe0\x94H)\x82\xac\x1f\x1cm\x17!\xfe\xec\u0679\xc9l\xd9I\x80PU\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4H0,1\x1e\xf8\xe5\xdcfAX\xddX<\x81\x19Mn\rX\x89\xb6gl\xe0\xbc\xcb\\\x00\x00\u07d4H;\xa9\x904\xe9\x00\xe3\xae\xdfaI\x9d;+\xce9\xbe\xb7\xaa\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4HT\x8bK\xa6+\xcb/\r4\xa8\x8d\u019ah\x0eS\x9c\xf0F\x89\x05l\xf1\u02fbt2\x00\x00\u07d4Hc\x84\x979&Zc\xb0\xa2\xbf#jY\x13\xe6\xf9Y\xce\x15\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d4He\x9d\x8f\x8c\x9a/\xd4Oh\u06a5]#\xa6\b\xfb\xe5\x00\u0709lk\x93[\x8b\xbd@\x00\x00\xe0\x94Hf\x9e\xb5\xa8\x01\u0637_\xb6\xaaX\xc3E\x1bpX\xc2C\xbf\x8a\x06\x8dB\xc18\u06b9\xf0\x00\x00\u07d4Hjl\x85\x83\xa8D\x84\xe3\xdfC\xa1#\x83\u007f\x8c~#\x17\u0409\x11\x87\xc5q\xab\x80E\x00\x00\u07d4Hz\xdf}p\xa6t\x0f\x8dQ\xcb\xddh\xbb?\x91\u0125\xceh\x89\x03\x9f\xba\xe8\xd0B\xdd\x00\x00\u07d4H~\x10\x85\x02\xb0\xb1\x89\uf70cm\xa4\xd0\xdbba\xee\xc6\xc0\x89g\x8a\x93
b\xe4\x18\x00\x00\xe0\x94H\x88\xfb%\xcdP\u06f9\xe0H\xf4\x1c\xa4}x\xb7\x8a'\xc7\u064a\x03\xa9\u057a\xa4\xab\xf1\xd0\x00\x00\u0794H\x934\u00b6\x95\xc8\xee\a\x94\xbd\x86B\x17\xfb\x9f\xd8\xf8\xb15\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4H\xa3\r\xe1\xc9\x19\xd3\xfd1\x80\xe9}_+*\x9d\xbd\x96M-
\x89\x02b\x9ff\xe0\xc50\x00\x00\u07d4H\xbf\x14\u05f1\xfc\x84\xeb\xf3\xc9k\xe1/{\xce\x01\xaai\xb0>\x89\x06\x81U\xa46v\xe0\x00\x00\u07d4H\xc2\ue465\aV\xd8\u039a\xbe\xebu\x89\xd2,o\xee]\xfb\x89\xae\x8ez\v\xb5u\xd0\x00\x00\u07d4H\xc5\u0197\v\x91a\xbb\x1c{z\xdf\xed\x9c\xde\u078a\x1b\xa8d\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94H\xd2CKz}\xbb\xff\b\";c\x87\xb0]\xa2\xe5\t1&\x8a\x03\xcf\xc8.7\xe9\xa7@\x00\x00\u07d4H\xd4\xf2F\x8f\x96?\u05da\x00a\x98\xbbg\x89]-
Z\xa4\u04c9K\xe4\xe7&{j\xe0\x00\x00\u07d4H\xe0\xcb\xd6\u007f\x18\xac\xdbzb\x91\xe1%M\xb3.\trs\u007f\x89\x05k\xe0<\xa3\xe4}\x80\x00\u07d4H\xf6\n5HO\xe7y+\u030a{c\x93\xd0\u0761\xf6\xb7\x17\x89\xc3(\t>a\xee@\x00\x00\u07d4H\xf8\x83\xe5g\xb46\xa2{\xb5\xa3\x12M\xbc\x84\xde\xc7u\xa8\x00\x89)\xd7n\x86\x9d\u0340\x00\x00\xe0\x94I\x01E\xaf\xa8\xb5E\"\xbb!\xf3R\xf0m\xa5\xa7\x88\xfa\x8f\x1d\x8a\x01\xf4lb\x90\x1a\x03\xfb\x00\x00\u07d4I\t\xb3\x19\x98\xea\xd4\x14\xb8\xfb\x0e\x84k\xd5\xcb\xde995\xbe\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4I\x12\xd9\x02\x93\x16v\xff9\xfc4\xfe<<\xc8\xfb!\x82\xfaz\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4I\x13o\xe6\xe2\x8btS\xfc\xb1kk\xbb\u9aac\xba\x837\xfd\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94I\x15a\u06cbo\xaf\xb9\x00~b\xd0P\u0082\xe9,Kk\u020a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4I\x18]\xd7\xc262\xf4lu\x94s\ubb96`\b\xcd5\x98\x89\r\xc5_\xdb\x17d{\x00\x00\u07d4I,\xb5\xf8a\xb1\x87\xf9\xdf!\xcdD\x85\xbe\xd9\vP\xff\xe2-\x89\x1b\x19\xe5\vD\x97|\x00\x00\u07d4I-

\xe4j\xaf\x8f\x1dp\x8dY\u05da\xf1\xd0:\xd2\xcb`\x90/\x89lk\x93[\x8b\xbd@\x00\x00\u07d4I.p\xf0M\x18@\x8c\xb4\x1e%`70Pk5\xa2\x87k\x89\x02\"\xc8\xeb?\xf6d\x00\x00\u07d4I:g\xfe#\xde\xccc\xb1\r\xdau\xf3(v\x95\xa8\x1b\u056b\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4I=H\xbd\xa0\x15\xa9\xbf\xcf\x16\x03\x93n\xabh\x02L\xe5Q\xe0\x89\x018\xa3\x88\xa4<\x00\x00\x00\xe0\x94IBV\xe9\x9b\x0f\x9c\xd6\xe5\xeb\xca8\x99\x862R\x90\x01e\u020a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4IM\xecM^\xe8\x8a'q\xa8\x15\xf1\xeerd\x94/\xb5\x8b(\x89lk\x93[\x8b\xbd@\x00\x00\u07d4I[d\x1b\x1c\u07a3b\u00f4\u02fd\x0f\\\xc5\v\x1e\x17k\x9c\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94Ih\xa2\xce\xdbEuU\xa19)Z\xea(wnT\x00<\x87\x8a\x02#\x1a\xef\u0266b\x8f\x00\x00\u07d4Im6U4S\n_\xc1W|\nRA\u02c8\xc4\xdapr\x89a\t=|,m8\x00\x00\xe0\x94In1\x95\x92\xb3A\xea\xcc\xd7x\u0767\xc8\x19mT\xca\xc7u\x8a\x01\xf5q\x89\x87fKH\x00\x00\u07d4IoXC\xf6\xd2L\u064d%^L#\xd1\xe1\xf0#\"uE\x89_\x17\x9f\u0526\xee\t\x80\x00\xe0\x94Ip\u04ec\xf7+[\x1f2\xa7\x00<\xf1\x02\xc6N\xe0TyA\x8a\x1d\xa5jK\b5\xbf\x80\x00\x00\u07d4Iw\xa7\x93\x9d\t9h\x94U\xce&9\xd0\xeeZL\xd9\x10\xed\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4Iy\x19N\xc9\xe9}\xb9\xbe\xe84;|w\xd9\xd7\xf3\xf1\u071f\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4Iy4c\xe1h\x10\x83\u05ab\xd6\xe7%\u057b\xa7E\xdc\xcd\xe8\x89\x1d\x98\xe9LNG\x1f\x00\x00\u07d4I\x81\xc5\xfff\xccN\x96\x80%\x1f\xc4\xcd/\xf9\a\xcb2xe\x89(\xa8WBTf\xf8\x00\x00\u07d4I\x89\u007f\xe92\xbb\xb3\x15L\x95\u04fc\xe6\xd9;ms)\x04\u0749\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4I\x89\xe1\xab^|\xd0\aF\xb3\x93\x8e\xf0\xf0\xd0d\xa2\x02[\xa5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4I\x8a\xbd\xeb\x14\xc2k{r4\xd7\x0f\u03ae\xf3a\xa7m\xffr\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4I\xa6E\xe0f}\xfd{2\xd0u\xcc$g\u074ch\t\a\u0109\a\x06\x01\x95\x8f\u02dc\x00\x00\xe0\x94I\xb7N\x16\x92e\xf0\x1a\x89\xecL\x90r\u0164\xcdr\xe4\xe85\x8a\x03h\xc8b:\x8bM\x10\x00\x00\u07d4I\xbd\xbc{\xa5\xab\xeb\xb68\x9e\x91\xa3(R
\xd3E\x1b\xd2S\x8965\u026d\xc5\u07a0\x00\x00\u07d4I\xc9A\xe0\xe5\x01\x87&\xb7)\x0f\xc4s\xb4q\xd4\x1d\xae\x80\u0449\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94I\xc9w\x1f\xca\x19\u0579\xd2E\u0211\xf8\x15\x8f\xe4\x9fG\xa0b\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4I\xcf\x1eT\xbe61\x06\xb9 r\x9d-
\v\xa4o\bg\x98\x9a\x89\x0e\x87?D\x13<\xb0\x00\x00\u07d4I\xd2\u008e\xe9\xbcT^\xaa\xf7\xfd\x14\xc2|@s\xb4\xbb_\x1a\x89O\xe9\xb8\x06\xb4\r\xaf\x00\x00\u07d4I\xdd\xee\x90.\x1d\f\x99\u0471\x1a\xf3\u030a\x96\xf7\x8eM\xcf\x1a\x89\n\u03a5\xe4\xc1\x8cS\x00\x00\u07d4I\xf0(9[Z\x86\xc9\xe0\u007fwxc\x0eL.=7:w\x89\x06\xa7JP8\u06d1\x80\x00\xe0\x94J\x195\xe2a\x9b$\xb0p\x9dVY\x0e\x91\x83\xcc\xf2\xc1\u064a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4J@S\xb3\x1d\x0e\xe5\u06ef\xb1\xd0k\u05ec\u007f\xf3\",G\u0589K\xe4\xe7&{j\xe0\x00\x00\u07d4JC\x01p\x15-
\xe5\x17&3\u0742b\xd1\a\xa0\xaf\xd9j\x0f\x89\xabM\xcf9\x9a:`\x00\x00\u07d4JG\xfc>\x17\u007fVz\x1e8\x93\xe0\x00\xe3k\xba#R\n\xb8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4JR\xba\xd2\x03W\"\x8f\xaa\x1e\x99k\xedy\f\x93gK\xa7\u0409Hz\x9a0E9D\x00\x00\u07d4JS\xdc\xdbV\xceL\xdc\xe9\xf8.\xc0\xeb\x13\xd6sR\xe7\u020b\x89\u3bb5sr@\xa0\x00\x00\u07d4J_\xae;\x03r\xc20\xc1%\xd6\xd4p\x14\x037\xab\x91VV\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4Jq\x90a\xf5(T\x95\xb3{\x9d~\xf8\xa5\x1b\a\xd6\u6b2c\x89\n\xd4\xc81j\v\f\x00\x00\u07d4Js8\x92\x98\x03\x1b\x88\x16\u0329FB\x1c\x19\x9e\x18\xb3C\u0589\"8h\xb8y\x14o\x00\x00\u07d4Js]\"G\x927m3\x13g\xc0\x93\xd3\x1c\x87\x944\x15\x82\x89f\xff\xcb\xfd^Z0\x00\x00\u07d4Jt\x94\xcc\xe4HU\u0300X(B\xbe\x95\x8a\r\x1c\x00r\ue242\x1a\xb0\xd4AI\x80\x00\x00\u07d4Ju\xc3\xd4\xfao\u033d]\u0567\x03\xc1Sy\xa1\xe7\x83\u9dc9b\xa9\x92\xe5:\n\xf0\x00\x00\xe0\x94J\x81\xab\xe4\x98L|k\xefc\u0598 \xe5WC\xc6\x1f

\x1c\x8a\x03d\x01\x00N\x9a\xa3G\x00\x00\u07d4J\x82iO\xa2\x9d\x9e!2\x02\xa1\xa2\t(]\xf6\xe7E\xc2\t\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4J\x83\\%\x82LG\xec\xbf\u01d49\xbf?\\4\x81\xaau\u0349K\xe4\xe7&{j\xe0\x00\x00\u07d4J\x91\x802C\x91Y\xbb1[g%\xb6\x83\r\xc86\x97s\x9f\x89\x12\xa3.\xf6x3L\x00\x00\u07d4J\x97\xe8\xfc\xf4c^\xa7\xfc^\x96\xeeQu.\u00c8qk`\x89\x1d\x99E\xab+\x03H\x00\x00\u07d4J\x9a&\xfd\n\x8b\xa1\x0f\x97}\xa4\xf7|1\x90\x8d\xabJ\x80\x16\x89a\t=|,m8\x00\x00\u07d4J\xa1H\xc2\xc34\x01\xe6j+Xnew\u0132\x92\xd3\xf2@\x89\v\xb8`\xb2\x85\xf7t\x00\x00\u07d4J\xa6\x93\xb1\"\xf3\x14H*G\xb1\x1c\xc7|h\xa4\x97\x87ab\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4J\xb2\xd3O\x04\x83O\xbftyd\x9c\xab\x92=,G%\xc5S\x89\xbe\xd1\xd0&=\x9f\x00\x00\x00\u07d4J\xc0vs\xe4/d\xc1\xa2^\xc2\xfa-
\x86\xe5\xaa+4\xe09\x89lk\x93[\x8b\xbd@\x00\x00\u07d4J\u016c\xad\x00\v\x88w!L\xb1\xae\x00\xea\u0263}Y\xa0\xfd\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4J\u0250ZL\xb6\xab\x1c\xfdbTn\xe5\x91s\x00\xb8|O\u07897\b\xba\xed=h\x90\x00\x00\u07d4J\u03e9\xd9N\xdaf%\xc9\u07e5\xf9\xf4\xf5\xd1\a\xc4\x03\x1f\u07c9\x02\"\xc8\xeb?\xf6d\x00\x00\u07d4J\xd0G\xfa\xe6~\xf1b\xfeh\xfe\xdb\xc2};e\xca\xf1\f6\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4J\xd9]\x18\x8dddp\x9a\xdd%U\xfbM\x97\xfe\x1e\xbf1\x1f\x89\x12\xc1\xb6\xee\xd0=(\x00\x00\u07d4J\xdb\xf4\xaa\xe0\xe3\xefD\xf7\xddM\x89\x85\u03ef\tn\u010e\x98\x89\b!\xab\rD\x14\x98\x00\x00\u07d4J\xe2\xa0M9\t\xefENTL\xcf\xd6\x14\xbf\xef\xa7\x10\x89\xae\x89\x18\x01\x15\x9d\xf1\xee\xf8\x00\x00\xe0\x94J\xe90\x82\xe4Q\x87\xc2a`\xe6g\x92\xf5\u007f\xad5Q\xc7:\x8a\x04\x96\x15
\xda\xff\x82(\x00\x00\u07d4J\xf0\xdb\a{\xb9\xba^D>!\xe1H\xe5\x9f7\x91\x05\u0152\x89
\x86\xac5\x10R`\x00\x00\u07d4K\x06\x19\xd9\u062a1:\x951\xac}\xbe\x04\xca\rjZ\u0476\x89lk\x93[\x8b\xbd@\x00\x00\u07d4K\v\u062c\xfc\xbcS\xa6\x01\v@\xd4\u040d\xdd-\x9dib-
\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4K\x19\xeb\f5K\xc199`\xeb\x06\x06;\x83\x92o\rg\xb2\x89\x01\x92t\xb2Y\xf6T\x00\x00\u07d4K)C|\x97\xb4\xa8D\xbeq\u0323\xb6H\xd4\xca\x0f\u075b\xa4\x89\b$q\x984\u03ec\x00\x00\u07d4K1\xbfA\xab\xc7\\\x9a\xe2\u034f\u007f5\x16;n+tPT\x89\x14\xb5P\xa0\x13\xc78\x00\x00\u07d4K:|\u00e7\u05f0\x0e\xd5(\"!\xa6\x02Y\xf2[\xf6S\x8a\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94K:\xab3^\xbb\xfa\xa8p\xccM`^}.t\xc6h6\x9f\x8a\f\xb4\x9bD\xba`-
\x80\x00\x00\u07d4K<s\x88\xccv\xda=b\xd4\x00g\u06bc\xcd~\xf0C=#\x89\x05l\xd5_\xc6M\xfe\x00\x00\u07d4K=\xfb\xdbEK\xe5'\x9a;\x8a\xdd\xfd\x0e\xd1\xcd7\xa9B\r\x89lk\x93[\x8b\xbd@\x00\x00\u07d4KG\x0f{\xa00\xbc|\xfc\xf38\u053f\x042\xa9\x1e.\xa5\xff\x89lk\x93[\x8b\xbd@\x00\x00\u07d4KS\xaeY\u01c4\xb6\xb5\xc46\x16\xb9\xa0\x80\x95X\xe6\x84\xe1\f\x89A\rXj
\xa4\xc0\x00\x00\u07d4KX\x10\x1fD\xf7\xe3\x89\xe1-
G\x1d\x165\xb7\x16\x14\xfd\xd6\x05\x89\b\xacr0H\x9e\x80\x00\x00\u07d4K\\\xdb\x1eB\x8c\x91\xdd|\xb5Jj\xedEq\xda\x05K\xfeR\x89\x04\xc5>\xcd\xc1\x8a`\x00\x00\u07d4K`\xa3\xe2S\xbf8\xc8\xd5f
\x10\xbb\x93\xa4s\xc9e\xc3\xe5\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4Kt\xf5\xe5\x8e.\xdfv\xda\xf7\x01Q\x96J\v\x8f\x1d\xe0f<\x89\x11\x90\xaeID\xba\x12\x00\x00\u07d4Kv!f\xdd\x11\x18\xe8Ci\xf8\x04\xc7_\x9c\xd6W\xbfs\f\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4Ky.)h>\xb5\x86\u353b3Rl`\x01\xb3\x97\x99\x9e\x89 \x86\xac5\x10R`\x00\x00\u07d4K\x90N\x93K\xd0\u030b
p_\x87\x9e\x90[\x93\xea\f\xcc0\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94K\x92\x06\xbakT\x9a\x1a\u007f\x96\x9e\x1d]\xba\x86u9\xd1\xfag\x8a\x01\xab,\xf7\xc9\xf8~
\x00\x00\u07d4K\x98N\xf2lWn\x81Z.\xae\xd2\xf5\x17\u007f\a\u06f1\xc4v\x89T\x91YV\xc4\t`\x00\x00\u07d4K\x9e\x06\x8f\xc4h\tv\xe6\x15\x04\x91)\x85\xfd\\\xe9K\xab\r\x89$=M\x18\"\x9c\xa2\x00\x

00\u07d4K\xa0\xd9\xe8\x96\x01w+IhG\xa2\xbbC@\x18g\x87\xd2e\x8965\u026d\xc5\u07a0\x00\x
00\u07d4K\xa5:\xb5I\xe2\x01m\xfa\"<\x9e\u0563\x8f\xad\x91(\x8d\a\x89K\xe4\xe7&{j\xe0\x00\x00\
xe0\x94K\xa8\xe0\x11\u007f\xc0\xb6\xa3\xe5k$\xa3\xa5\x8f\xe6\xce\xf4B\xff\x98\x8a\x011\xbe\xb
9%\xff\xd3
\x00\x00\u07d4K\xac\x84j\xf4\x16\x9f\x1d\x95C\x1b4\x1d\x88\x00\xb2!\x80\xaf\x1a\x89\x01\x15\x
8eF\t\x13\xd0\x00\x00\u07d4K\xb6\xd8k\x83\x14\xc2-\x8d7\xeaQm\x00\x19\xf1V\xaa\xe1-
\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94K\xb9e\\\xfb*6\xea|cz{\x85\x9bJ1T\xe2n\xbe\x8a\x03c\\\
x9a\xdc]\xea\x00\x00\x00\xe0\x94K\xbc\xbf8\xb3\xc9\x01c\xa8K\x1c\u04a9;X\xb2\xa34\x8d\x87\x
8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94K\xd6\xdd\f\xff#@\x0e\x170\xba{\x89E\x04W}\x14\xe7
J\x8a+\xa0\xcc\xdd\xd0\xdfs\xb0\x00\x00\u07d4K\xe8b\x8a\x81T\x87N\x04\x8d\x80\xc1B\x18\x10
\"\xb1\x80\xbc\xc1\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4K\xe9\rA!)\u0564\xd0BCa\xd6d\x9dNG
\xa6#\x16\x897\b\xba\xed=h\x90\x00\x00\xe0\x94K\xea(\x8e\xeaB\u0115^\xb9\xfa\xad*\x9f\xafG\x
83\xcb\u076c\x8a\x06\x18\xbe\x16c\u012fI\x00\x00\u07d4K\xf4G\x97\x99\xef\x82\xee\xa2\tC7OV\
xa1\xbfT\x00\x1e^\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4K\xf8\xbf\x1d5\xa211Wd\xfc\x80\x01\x
80\x9a\x94\x92\x94\xfcI\x89\x03\x9f\xba\xe8\xd0B\xdd\x00\x00\u07d4K\xf8\xe2oL'\x90\xdae3\xa2\
xac\x9a\xba\xc3\u019a\x19\x943\x89\n\u05ce\xbcZ\xc6
\x00\x00\u0794L\n\xcaP\x8b<\xaf^\xe0(\xbcp}\xd1\xe8\x00\xb88\xf4S\x88\xfc\x93c\x92\x80\x1c\x0
0\x00\xe0\x94L\v\x15\x15\xdf\xce\u05e1>\x13\xee\x12\xc0\xf5#\xaePO\x03+\x8a\n\x96\x81c\xf0\x
a5{@\x00\x00\u07d4L\x13\x98\f2\xdc\xf3\x92\vx\xa4\xa7\x903\x12\x90|\x1b\x12?\x89\x03A\x00\x
15\xfa\xae\f\x00\x00\u07d4L\x15y\xaf3\x12\xe4\xf8\x8a\xe9<h\xe9D\x9c.\x9ah\xd9\u0109lk\x93[\x
8b\xbd@\x00\x00\xe0\x94L#\xb3p\xfc\x99+\xb6|\xec\x06\xe2g\x15\xb6/\v:J\u00ca\x02\x1e\x19\xe
0\u027a\xb2@\x00\x00\u07d4L$\xb7\x8b\xaf+\xaf\xc7\xfc\u0190\x16Bk\xe9s\xe2\nP\xb2\x89\xa2\
xa1]\tQ\x9b\xe0\x00\x00\xe0\x94L/\x1a\xfe\xf7\u0146\x8cD\x83/\xc7|\xb0;U\xf8\x9emn\x8a\x04<3\
xc1\x93ud\x80\x00\x00\u07d4L7{\xb0:\xb5,L\xb7\x9b\xef\xa1\xdd\x11I\x82\x92LJ\xe9\x89c\x16\x0
3\xcc\u04cd\xd7\x00\x00\u07d4L>\x95\xcc9W\xd2R\xce\v\xf0\xc8}[O\"4g.p\x89\x87\x86x2n\xac\x9
0\x00\x00\u07d4LB<v\x93\r\a\xf9<G\xa5\xccOaWE\xc4Z\x9dr\x89\x05k\xc7^-
c\x10\x00\x00\u07d4LE\xd4\u0267%\xd1\x11\x12\xbf\xcb\xca\x00\xbf1\x18l\u02ad\xb7\x89\x15\xa
f\x1dx\xb5\x8c@\x00\x00\u07d4LNo\x13\xfb^?p\xc3v\x02b\xa0>1y\x82i\x1d\x10\x89\x05k\xc7^-
c\x10\x00\x00\u07d4LZ\xfe@\xf1\x8f\xfcH\u04e1\xae\xc4\x1f\u009d\xe1y\xf4\u0497\x89lk\x93[\x8
b\xbd@\x00\x00\u07d4L[=\xc0\xe2\xb96\x0f\x91(\x9b\x1f\xe1<\xe1,\x0f\xbd\xa3\xe1\x89lk\x93[\x8
b\xbd@\x00\x00\u07d4Lfk\x86\xf1\xc5\ue324\x12\x85\xf5\xbd\xe4\xf7\x90R\b\x14\x06\x89\x1b\x1
a\xe4\xd6\xe2\xefP\x00\x00\u07d4Lik\xe9\x9f:i\x04@\xc3CjY\xa7\xd7\xe97\u05ba\r\x89\xbb\x91%
T\"c\x90\x00\x00\u07d4Lj$\x8f\xc9}p]\xefI\\\xa2\aY\x16\x9e\xf0\xd3dq\x89)3\x1eeX\xf0\xe0\x00\x0
0\u07d4Lj\x9d\xc2\u02b1\n\xbb.|\x13p\x06\xf0\x8f\ucd77y\xe1\x89\x1b\r\x04
/G\xec\x00\x00\u07d4Lk\x93\xa3\xbe\xc1cIT\f\xbf\xca\xe9l\x96!\xd6dP\x10\x89lk\x93[\x8b\xbd@\x
00\x00\u07d4Lu\x98\x13\xad\x13\x86\xbe\xd2\u007f\xfa\xe9\xe4\x81^60\u0323\x12\x89lk\x93[\x8
b\xbd@\x00\x00\xe0\x94Lv\f\xd9\xe1\x95\xeeO-k\xce%\x00\xff\x96\xda|C\ue44a\f\xb4\x9bD\xba`-
\x80\x00\x00\u07d4Lv{e\xfd\x91\x16\x1fO\xbd\xccji\xe2\xf6\xadq\x1b\xb9\x18\x89'\b\x01\xd9F\xc9
@\x00\x00\u07d4L~.+w\xad\f\xd6\xf4J\xcb(a\xf0\xfb\x8b(u\x0e\xf9\x89\x01\x15\x8eF\t\x13\xd0\x0
0\x00\u07d4L\x85\xed6/$\xf6\xb9\xf0L\xdf\xcc\xd0\"\xaeSQG\u02f9\x89QP\xae\x84\xa8\xcd\xf0\x0
0\x00\u07d4L\x93[\xb2Pw\x8b<L\u007f~\a\xfc%\x1f\xa601J\xab\x89QP\xae\x84\xa8\xcd\xf0\x00\x
00\u07d4L\x99y\x92\x03l[C:\xc3=%\xa8\xea\x1d\xc3\xd4\xe4\xe6\u0609\x01\x95;=J\xb1h\x00\x00\

u07d4L\x99\xda\xe9d\x81\xe8\a\xc1\xf9\x9f\x8b\u007f\xbd\xe2\x9buG\u017f\x89\b!\xab\rD\x14\x9
8\x00\x00\u07d4L\x9a\x86*\xd1\x15\xd6\xc8'N\u0439D\xbd\u05a5P\x05\x10\xa7\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94L\xa7\x83\xb5V\xe5\xbfS\xaa\x13\xc8\x11f\x13\xd6W\x82\u0276B\x8a\x05V
\x18@\xb4\xad\x83\xc0\x00\x00\u07d4L\xa7\xb7\x17\u067c\x87\x93\xb0N\x05\x1a\x8d#\xe1d\x0f[
\xa5\xe3\x89C\xb5\x14T\x9e\xcfb\x00\x00\u07d4L\xa8\xdbJ^\xfe\xfc\x80\xf4\u035b\xbc\u03302e\x
93\x132\xb6\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4L\xac\x91\xfb\x83\xa1G\xd2\xf7l2g\x98K\x91\ny\x933H\x89uy*\x8a\xbd\xef|\x00\x0
0\u07d4L\xad\xf5s\xceL\xee\u01cb\x8e\x1b!\xb0\xedx\xeb\x11;,\x0e\x89lk\x93[\x8b\xbd@\x00\x00
\u07d4L\xb5\xc6\xcdq<\xa4G\xb8H\xae/V\xb7a\xca\x14\u05edW\x89\x0e~\xeb\xa3A\vt\x00\x00\u
07d4L\xc2,\x9b\u026d\x05\xd8u\xa3\x97\xdb\xe8G\xed\"\x1c\x92\fg\x89lk\x93[\x8b\xbd@\x00\x00\
u07d4L\u0430\xa6CcbY\\\xea\xde\x05.\xbc\x9b\x92\x9f\xb6\xc6\xc0\x89lk\x93[\x8b\xbd@\x00\x00
\u07d4L\xdaA\xddS9\x91)\a\x94\xe2*\xe3$\x14>0\x9b==\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\u07
d4L\xee\x90\x1bJ\u0231V\xc5\xe2\xf8\xa6\xf1\xbe\xf5r\xa7\xdc\xeb~\x8965\u026d\xc5\u07a0\x00\
x00\u07d4L\xef\xbe#\x98\xe4}R\u73743L\x8bivu\U00053b89\xd9o\u0390\u03eb\xcc\x00\x00\u07d
4L\xf5S{\x85\x84/\x89\xcf\xee5\x9e\xaeP\x0f\xc4I\xd2\x11\x8f\x8965\u026d\xc5\u07a0\x00\x00\xe
0\x94M\bG\x1dh\x00z\xff*\xe2y\xbc^?\xe4\x15o\xbb\xe3\u078a\bxg\x83&\xea\xc9\x00\x00\x00\u0
7d4M
\x01\x10\x12@\b\xd5ov\x98\x12VB\f\x94jo\xf4\\\x89\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4M$\xb7\
xacG\xd2\xf2}\xe9\tt\xba=\xe5\xea\xd2\x03TK\u0349\x05k\xc7^-
c\x10\x00\x00\u0794M)\xfcR:,\x16)S!!\u0699\x98\u9d6b\x9d\x1bE\x88\xdbD\xe0I\xbb,\x00\x00\u0
7d4M8\xd9\x0f\x83\xf4Q\\\x03\xccx2j\x15M5\x8b\u0602\xb7\x89\n\ad\a\xd3\xf7D\x00\x00\u07d4M
L\xf5\x80t)a^0\xcd\xfa\xce\x1eZ\xaeM\xad0U\xe6\x89
\x86\xac5\x10R`\x00\x00\u07d4MW\xe7\x16\x87l\f\x95\xef^\xae\xbd5\xc8\xf4\x1b\x06\x9bk\xfe\x8
9lk\x93[\x8b\xbd@\x00\x00\xe0\x94Mg\U000ab159\xfe\xf5\xfcA9\x99\xaa\x01\xfd\u007f\xcep\xb4=
\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4Mn\x8f\xe1\t\xcc\xd2\x15\x8eM\xb1\x14\x13/\xe
7_\xec\u023e[\x89\x01[5W\xf1\x93\u007f\x80\x00\xe0\x94Mq\xa6\xeb=\u007f2~\x184'\x8e(\v\x03\
x9e\xdd\xd3\x1c/\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4M|\xfa\xa8L\xb31\x06\x80\n\x8c\x80
/\xb8\xaaF8\x96\u0159\x89a\t=|,m8\x00\x00\u07d4M\x80\x10\x93\xc1\x9c\xa9\xb8\xf3B\xe3<\xc9\
xc7{\xbdL\x83\x12\u03c9\x12\xb3\xe7\xfb\x95\u0364\x80\x00\u07d4M\x82\x88\x94u/o%\x17]\xaf!
w\tD\x87\x95Ko\x9f\x89O!+\xc2\u011c\x83\x80\x00\xe0\x94M\x82\xd7p\f\x12;\xb9\x19A\x9b\xba\x
f0Fy\x9ck\x0e,f\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4M\x83m\x9d;\x0e,\xbdM\xe0PYo\xaaI\f\xf
f\xb6\r]\x89\x10CV\x1a\x88)0\x00\x00\u07d4M\x86\x97\xaf\x0f\xbf,\xa3n\x87h\xf4\xaf\"\x135phZ`\x
89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4M\x92y\x96
)\xa8\xbdEc\x977\xe9\x8bQ\x1e\xff\aL!\x89Hz\x9a0E9D\x00\x00\u07d4M\x93io\xa2HY\xf5\u0493\
x9a\xeb\xfaT\xb4\xb5\x1a\xe1\xdc\u0309\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\u07d4M\x9cw\xd0u\f^
o\xbc$\u007f/\u05d2thl\xb3S\u0589\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4M\xa5\xed\u0188\xb0\x
cbb\xe1@=\x17\x00\xd9\u0739\x9f\xfe?\u04c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94M\xa8\x03\ai\x84
K\xc3A\x86\xb8\\\xd4\xc74\x88I\xffI\xe9\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4M\xb1\x
c4:\x0f\x83M}\x04x\xb8\x96\ag\xec\x1a\xc4L\x9a\xeb\x89/Q\x810V'7\x00\x00\u07d4M\xb2\x12\x8
4\xbc\xd4\xf7\x87\xa7Ue\x00\xd6\xd7\xd8\xf3f#\xcf5\x89i(7Ow\xa3c\x00\x00\u07d4M\xc3\xda\x13\
xb2\xb4\xaf\xd4O]\r1\x89\xf4D\xd4\xdd\xf9\x1b\x1b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4M\u013f^
u\x89\xc4{(7\x8du\x03\u03d6H\x80a\u06fd\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4M\xc9\u057b

K\x19\xce\u0354\xf1\x9e\xc2]
\x0e\xa7/%\xd7\xed\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94M\xcd\x11\x81X\x18\xae)\xb8]\x016sI\xa8\xa7\xfb\x12\xd0k\x8a\x01\xacB\x86\x10\x01\x91\xf0\x00\x00\u07d4M\xcfb\xa3\xde?\x06\x1d\xb9\x14\x98\xfda\x06\x0f\x1fc\x98\xffs\x89lj\xccg\u05f1\xd4\x00\x00\u07d4M\xd11\xc7J\x06\x8a7\xc9\n\xde\xd4\xf3\t\xc2@\x9fdx\u04c9\x15\xaf9\u4ab2t\x00\x00\xe0\x94M\u0767Xk\"7\xb0S\xa7\xf3(\x9c\xf4`\xdcW\xd3z\t\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4M\xe3\xfe4\xa6\xfb\xf64\xc0Q\x99\u007fG\xcc\u007fHy\x1fX$\x89l]\xb2\xa4\xd8\x15\xdc\x00\x00\u07d4M\xf1@\xbaye\x85\xddT\x891[\xcaK\xbah\n\u06f8\x18\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4N\x02\ay\xb5\xdd\xd3\xdf\"\x8a\x00\xcbH\xc2\xfc\x97\x9d\xa6\xae8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4N\v\xd3$s\xc4\xc5\x1b\xf2VT\xde\xf6\x9fy|k)\xa22\x89V\xc9]\xe8\xe8\xca\x1d\x00\x00\u07d4N\"%\xa1\xbbY\xbc\x88\xa21ft\xd33\xb9\xb0\xaf\xcafU\x89\bg\x0e\x9e\xc6Y\x8c\x00\x00\u07d4N#\x10\x19\x1e\xad\x8d;\xc6H\x98s\xa5\xf0\xc2\xeck\x87\u1f8965\u026d\xc5\u07a0\x00\x00\u07d4N#-
S\xb3\u6f8f\x89Sa\xd3\x1c4\xd4v+\x12\xc8.\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4N+\xfaJFo\x82g\x1b\x80\x0e\xeeBj\xd0\f\a\x1b\xa1p\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4N>\xda\u0506M\xabd\xca\xe4\xc5Azvw@S\xdcd2\x89
\b\xfbG\x8c\xbf\xa9\x80\x00\u07d4NC\x18\xf5\xe1>\x82JT\xed\xfe0\xa7\xedO&\xcd=\xa5\x04\x89lk\x93[\x8b\xbd@\x00\x00\u07d4N[w\xf9\x06aY\xe6\x15\x93?-
\xdatw\xfaNG\xd6H\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94Nf\x00\x80b\x89EJ\u03630\xa2\xa3U`\x10\u07ec\xad\xe6\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4Ns\xcf#y\xf1$\x86\x0fs\xd6\xd9\x1b\xf5\x9a\xcc\\\xfc\x84[\x89\x02,\xa3X|\xf4\xeb\x00\x00\xe0\x94Nz\xa6~\x12\x18>\xf9\xd7F\x8e\xa2\x8a\xd29\xc2\xee\xf7\x1bv\x8a\x01\n\xfc\x1a\xde;N\xd4\x00\x00\xe0\x94N{TGM\x01\xfe\xfd8\x8d\xfc\xd5;\x9ff&$A\x8a\x05\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94N\x89.\x80\x81\xbf6\xe4\x88\xfd\xdb;&0\xf3\xf1\xe8\xda0\u048a\x02\x8a\xba0u$Q\xfc\x00\x00\xe0\x94N\x8amcH\x9c\xcc\x10\xa5\u007f\x88_\x96\xeb\x04\xec\xbbT`$\x8a\x03\xea\xe3\x13\x0e\u0316\x90\x00\x00\u07d4N\x8eG\xae;\x1e\xf5\f\x9dT\xa3\x8e\x14
\x8c\x1a\xbd6\x03\u0089y(\xdb\x12vf\f\x00\x00\u0794N\x90\u03312X\xac\xaa\x9fO\xeb\xc0\xa3B\x92\xf9Y\x91\xe20\x88\xdbD\xe0I\xbb,\x00\x00\u07d4N\xa5n\x11\x12d\x1c\x03\x8d\x05e\xa9\u0096\xc4c\xaf\xef\xc1~\x89\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\xe0\x94N\xa7\x0f\x041?\xaee\xc3\xff\"J\x05\\=-
\xab(\xdd\u07ca\x04<0\xfb\b\x84\xa9l\x00\x00\u07d4N\xb1EKW8\x05\u022c\xa3~\xde\xc7\x14\x9aA\xf6\x12\x02\xf4\x89\x10CV\x1a\x88)0\x00\x00\u07d4N\xb8{\xa8x\x8e\xba\r\xf8~[\x9b\xd5\n\x8eE6\x80\x91\xc1\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4N\xbcV)\xf9\xa6\xa6k,\xf36:\u0109\\\x03H\u8fc7\x8967\tlK\xcci\x00\x00\u07d4N\xc7h)^\xea\xba\xfcB\x95\x84\x15\xe2+\xe2\x16\xcd\xe7v\x18\x89\x03;\x1d\xbc9\xc5H\x00\x00\u07d4N\xcc\x19\x94\x8d\xd9\u0347\xb4\xc7
\x1a\xb4\x8eu\x8f(\xe7\xccv\x89\x1b\x1d\xaba\u04ead\x00\x00\u07d4N\xd1M\x81\xb6\v#\xfb%\x05M\x89%\u07e5s\u072eah\x89\x12nr\xa6\x9aP\xd0\x00\x00\xe0\x94N\xe1<\rA
\vF\u045d\xee\\K\xce\xc7\x1d\x82\xbb\x8e8\x8a\x01\xab\xee\x13\u033e\ufbc0\x00\u07d4N\xea\xd4\n\xad\x8cs\xef\b\xfc\x84\xbc\n\x92\xc9\t/j6\xbf\x89\x01s\x17\x90SM\xf2\x00\x00\u07d4N\xeb\xe8\f\xb6\xf3\xaeY\x04\xf6\xf4\xb2\x8d\x90\u007f\x90q\x89\xfc\xab\x89lj\xccg\u05f1\xd4\x00\x00\u07d4N\xeb\xf1
]\f\xc2\f\xeel\u007f\x8f\xf3\x11_V\u050f\xba&\x89\x01\r:\xa56\xe2\x94\x00\x00\u07d4N\xf1\xc2\x14c:\xd9\xc0p;N#t\xa2\xe3>>B\x92\x91\x89Hz\x9a0E9D\x00\x00\u07d4N\xfc\xd9\u01df\xb43L\xa6${

\n3\xbd\x9c\xc32\b\xe2r\x89Hz\x9a0E9D\x00\x00\xe0\x94O\x06$k\x8dK\u0496a\xf4>\x93v\"\x01\u0486\x93Z\xb1\x8a\x01\x059O\xfcF6\x11\x00\x00\u07d4O\x15+/\xb8e\x9dCwn\xbb\x1e\x81g:\xa8Ai\xbe\x96\x89lk\x93[\x8b\xbd@\x00\x00\u07d4O\x17\u007f\x9dV\x95=\xedq\xa5a\x1f93\"\xc3\x02y\x89\\\x89\rU\uf422\xda\x18\x00\x00\u07d4O\x1a-
\xa5JLm\xa1\x9d\x14$\x12\xe5n\x81WA\xdb#%\x89\x05k\xc7^-
c\x10\x00\x00\u07d4O#\xb6\xb8\x17\xff\xa5\xc6d\xac\xda\u05db\xb7\xb7&\xd3\n\xf0\xf9\x89_h\xe8\x13\x1e\u03c0\x00\x00\xe0\x94O&i\f\x99+z1*\xb1.\x13\x85\xd9J\xcdX(\x8e{\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4O+G\xe2wZ\x1f\xa7\x17\x8d\xad\x92\x98Z[\xbeI;\xa6\u0589\n\u05ce\xbcZ\xc6
\x00\x00\u07d4O:HT\x91\x11E\xea\x01\xc6D\x04K\xdb.Z\x96\n\x98/\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4O?,g0i\xac\x97\xc2\x026\a\x15)\x81\xf5\xcd`c\xa0\x89
\x86\xac5\x10R`\x00\x00\xe0\x94OJ\x9b\xe1\f\xd5\xd3\xfb]\xe4\x8c\x17\xbe)o\x89V\x90d[\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4OR\xadap\xd2[*.\x85\x0e\xad\xbbRA?\xf20>\u007f\x89\xa4\xccy\x95c\u00c0\x00\x00\u07d4OX\x01\xb1\xeb0\xb7\x12\u0620WZ\x9aq\xff\x96]O4\xeb\x89\x10CV\x1a\x88)0\x00\x00\u07d4O]\xf5\xb9CW\u0794\x86\x04\xc5\x1bx\x93\xcd\xdf`v\xba\xad\x89\xcb\xd4{n\xaa\x8c\xc0\x00\x00\u07d4Od\xa8^\x8e\x9a@I\x8c\fu\xfc\xeb\x037\xfbI\b>^\x8965\u026d\xc5\u07a0\x00\x00\u07d4Og9m%S\xf9\x98x_pN\a\xa69\x19}\u0454\x8d\x89\x10DrR\x1b\xa78\x00\x00\u07d4OmG7\u05e9@8$\x87&H\x86i|\xf7c\u007f\x80\x15\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4Os0\toy\xed&N\xe0\x12\u007f]0\xd2\xf7<R\xb3\u0609\x1b\x1azB\v\xa0\r\x00\x00\u07d4Ov{\xc8yJ\xef\x9a\n8\xfe\xa5\xc8\x1f\x14iO\xf2\x1a\x13\x89\x1b\xc43\xf2?\x83\x14\x00\x00\u07d4O\x85\xbc\x1f\xc5\xcb\xc9\xc0\x01\xe8\xf17.\aPSp\xd8\xc7\x1f\x892\xf5\x1e\u06ea\xa30\x00\x00\u07d4O\x88\xdf\xd0\x10\x91\xa4Z\x9e&v\x02\x1ed(l\xd3k\x8d4\x8965\u026d\xc5\u07a0\x00\x00\u07d4O\x89r\x83\x8fp\xc9\x03\u0276\xc6\xc4ab\xe9\x9db\x16\xd4Q\x89\xf9\xe8\x9a\x0f,V\xc8\x00\x00\u07d4O\x8a\xe8\x028\xe6\x00\bUpu\xabj\xfe\n\u007f.t\xd7)\x89\x05k\xc7^-
c\x10\x00\x00\u07d4O\x8e\x8d'O\xb2*?\xd3jG\xfer\x98\x04qTK44\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94O\x9c\u2bdb\x8c^B\u0180\x8a8p\xecWo15E\u044a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4O\xa3\xf3.\xf4\bdH\xb3D\xd5\xf0\xa9\x89\r\x1c\xe4\xd6\x17\u00c9QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4O\xa5T\xab\x95\\$\x92\x178jM2c\xbb\xf7(\x95CN\x89\x01\x15NS!}\xdb\x00\x00\u07d4O\xa9\x83\xbb^0s\xa8\xed\xb5W\xef\xfe\xb4\xf9\xfb\x1d`\uf189V\xb9\xafW\xe5u\xec\x00\x00\u07d4O\xaf\x90\xb7n\u03f9c\x1b\xf9\x02!v\x03-\x8b,
p\t\x8966;]\x9awp\x00\x00\u07d4O\xc4l9ngHi\xad\x94\x81c\x8f\x00\x13c\f\x87\u02ac\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94O\xcc\x19\xea\x9fLW\u073c\xe8\x93\x19<\xfb\x16j\xa9\x14\xed\u014a\x01{\x8b\xaa\u007f\x19Tj\x00\x00\xe0\x94O\u0384)\xbaI\u02a06\x9d\x1eIM\xb5~\x89\uacad9\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4O\xda\xc1\xaaQp\a\xe0\b\x940\xb31j\x1b\xad\xd1,\x01\u01c9\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4O\xe5j\xb3\xba\u1c24D3E\x833\u0130Z$\x8f\x82A\x89v-\x93\xd1\xddo\x90\x00\x00\u07d4O\xeb\x84k\xe40A\xfdk4
(\x97\x94>?!\xcb\u007f\x04\x89\x04\x82\xfe&\f\xbc\xa9\x00\x00\u07d4O\xeeP\xc5\xf9\x88
k\t\xa5sF\x9f\xb1\u0434.\xbbm\u0389l\xee\x06\u077e\x15\xec\x00\x00\u07d4O\xf6v\xe2\u007fh\x1a\x98-
\x8f\xd9\xd2\x0ed\x8b=\xce\x05\xe9E\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4O\xf6\u007f\xb8\u007fn\xfb\xa9'\x990\u03fd\x1bz4<y\xfa\u0789\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4P\x06\xfeL\"\x179\x80\xf0\ft4+9\xcd#\x1ce1)\x89lk\x93[\x8b\xbd@\x00\x00\u07d4P\f\x165.\x90\x1dH\xba\x8

d\x04\xe2\xc7g\x12\x17ry\v\x02\x89\x01\xa3\xa6\x82Is\t\x80\x00\u07d4P\f\x90)X\xf6B\x15\x94\u0476\xde\xd7\x12I\rR\xedID\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94P\x0e4\xcd\u5f5e+q\xbb\x92\xd7\xcfU\xee\xe1\x88\xd5\xfa\f\x8a\x01!\xeah\xc1\x14\xe5\x10\x00\x00\u07d4P2\xe4\xbc\xf7\x93+I\xfd\xba7{o\x14\x99ce\x13\xcf\u00c9\x05k\xc7^-
c\x10\x00\x00\u07d4P7\x8a\xf7\xefT\x04?\x89*\xb7\u0397\xd6Gy5\x11\xb1\b\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4P;\xdb\u063cB\x1c2\xa4C\x03-
\xeb.>L\u057a\x8bN\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94PFf\u03891\x17^\x11\xa5\xed\x11\xc1\u072a\x06\xe5\u007fNf\x8a\x02\u007f>\u07f3Nn@\x00\x00\u0794PXM\x92\x06\xa4l\xe1\\0\x11\x17\xee(\xf1\\0\xe6\x0eu\x88\xb9\xf6]\x00\xf6<\x00\x00\xe0\x94PZ3\xa1\x864\xddH\x00i<g\xf4\x8a\x1di=H3\xf8\x8a\x01\x89!\xb7\x99A\xdc\xd0\x00\x00\u0794P^O|'U\x88\xc53\xa2\x0e\xbd*\xc1;@\x9b\xbd\xea<\x88\xf4?\xc2\xc0N\xe0\x00\x00\u07d4Pb\xe5\x13La/\x12iM\xbd\x0e\x13\x1dL\xe1\x97\u0476\xa4\x8965\u026d\xc5\u07a0\x00\x00\u07d4Pd\x11\xfdy\x004\x80\xf6\xf2\xb6\xaa\xc2k{\xa7\x92\U00014c89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4Pg\xf4T\x9a\xfb\xfe\x88LY\xcb\xc1+\x96\x93I#\xd4]\xb0\x8965\u026d\xc5\u07a0\x00\x00\u07d4Pv:\u0746\x8f\xd76\x11x4/\xc0U\ua8b9_hF\x89\x03\x9f\x90F\xe0\x89\x8f\x00\x00\u07d4P\x8c\xf1\x91\x19\xdbp\xaa\x86EBS\xdavJ,\xb1\xb2\xbe\x1a\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94P\x99\x82\xf5b7\xeeE\x89Q\x04~\n\"0\xf8\x04\xe2\u854a\x03\xb4\xadIa\x06\xb7\xf0\x00\x00\u07d4P\x9a
\xbcH\xe7+\xe1\u036f\x95i\xc7\x11\xe8d\x8d\x95s4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4P\x9c\x86h\x03m\x14?\xb8\xaep\xb1\x19\x95c\x1f=\xfc\xad\x87\x8965\u026d\xc5\u07a0\x00\x00\u07d4P\xad\x18z\xb2\x11g\u00b6\xe7\x8b\xe0\x15?DPJ\a\x94^\x89\x05l\xd5_\xc6M\xfe\x00\x00\u07d4P\xb9\xfe\xf0\xa12\x9b\x02\xd1e\x06%_Z-\xb7\x1e\xc9-
\x1f\x89G\u0682\x15d\b\\\x00\x00\u07d4P\xbbg\u0238\u063d\x0fc\xc4v\t\x04\xf2\xd33\xf4\x00\xaa\u0389lk\x93[\x8b\xbd@\x00\x00\u07d4P\xbe\xf2ubH\xf9\xa7\xa3\x80\xf9\x1b\x05\x1b\xa3\xbe(\xa6I\xed\x89li\xf7>)\x13N\x00\x00\u07d4P\u0286\xb5\xeb\x1d\x01\x87M\xf8\xe5\xf3IE\u051cl\x1a\xb8H\x8965\u026d\xc5\u07a0\x00\x00\u07d4P\u0357\xe97\x8b\\\xf1\x8f\x179c#l\x99Q\xeft8\xa5\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4P\u073c'\xbc\xad\x98@\x93\xa2\x12\xa9\xb4\x17\x8e\xab\xe9\x01ua\x89\a\xe3by\v\\\xa4\x00\x00\u07d4P\xe10#\xbd\x9c\xa9j\xd4\xc5?\xdf\xd4\x10\xcbk\x1fB\v\u07c9\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94P\xe1\xc8\xec\x98A[\xefD&\x18p\x87\x99C{\x86\xe6\xc2\x05\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4P\xf8\xfaK\xb9\xe2g|\x99\nN\xe8\xcep\xdd\x15#%\x1eO\x89\x01i=#\x16Ok\x00\x00\u07d4P\xfb6\xc2q\a\xee,\xa9\xa3#n'F\u0321\x9a\xcekI\x89lk\x93[\x8b\xbd@\x00\x00\u07d4P\xfe\xf2\x96\x95U\x88\u02aet\xc6.\xc3*#\xa4T\xe0\x9a\xb8\x89A\x1d\xff\xab\xc5\a8\x00\x00\u07d4Q\x02\xa4\xa4
w\xe1\x1cX\xdfGs\u3b14F#\xa6m\x9f\x89lp\x15\xfdR\xed@\x80\x00\u07d4Q\x03\x93w\xee\xd0\xc5s\xf9\x86\xc5\xe8\xa9_\xb9\x9aY\xe93\x0f\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4Q\x03\xbc\t\x93>\x99!\xfdS\xdcSo\x11\xf0]\rG\x10}\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94Q\x04\xec\xc0\xe30\xdd\x1f\x81\xb5\x8a\xc9\u06f1\xa9\xfb\xf8\x8a<\x85\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4Q\r\x81Y\u0314Wh\xc7E\a\x90\xba\a>\xc0\xd9\xf8\x9e0\x89\x8a\xc7#\x04\x89\xe8\x00\x00\x00\u07d4Q\x0e\xdaV\x01I\x9a\r^\x1a\x00k\xff\xfd\x836r\xf2\xe2g\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Q\x12dF\xab=\x802U~\x8e\xbaeY}u\xfa\u0701\\\x89\x11t\xa5\xcd\xf8\x8b\xc8\x00\x00\xe0\x94Q\x18U}`\r\x05\xc2\xfc\xbf8\x06\xff\xbd\x93\xd0
%\xd70\x8a\x02g\u04ebd#\xf5\x80\x00\x00\u07d4Q\x1e\x0e\xfb\x04\xacN?\xf2\xe6U\x0eI\x82\x95

\xbf\xcdV\xff\u0549$=M\x18\"\x9c\xa2\x00\x00\u07d4Q!\x16\x81{\xa9\xaa\xf8C\xd1P|e\xa5\xead\n{\x9e\xec\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4Q&F\ri,q\u026fo\x05WM\x93\x99\x83h\xa27\x99\x89\x02\u0465\x1c~\x00P\x00\x00\u07d4Q'\u007f\xe7\xc8\x1e\xeb\xd2R\xa0=\xf6\x9ak\x9f2n'\"\a\x89\x03@.y\u02b4L\x80\x00\u07d4Q)oPD'\r\x17pvF\x12\x9c\x86\xaa\xd1d^\xad\xc1\x89H|r\xb3\x10\xd4d\x80\x00\xe0\x94Q+\x91\xbb\xfa\xa9\xe5\x81\xefh?\xc9\r\x9d\xb2*\x8fI\xf4\x8b\x8aA\xa5\"8m\x9b\x95\xc0\x00\x00\u07d4Q5\xfb\x87W`\f\xf4tTbR\xf7M\xc0tm\x06&,\x89lk\x93[\x8b\xbd@\x00\x00\u07d4QF2\xef\xbdd,\x04\xdel\xa3B1]@\u0750\xa2\u06e6\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4QKu\x12\u026e^\xa6<\xbf\x11q[c\xf2\x1e\x18\u0496\xc1\x89lj\xccg\u05f1\xd4\x00\x00\u07d4QS\xa0\xc3\u0211(\x81\xbf\x1c5\x01\xbfd\xb4Vl\xe4\x82\"\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94QVQ\xd6\xdbO\xaf\x9e\xcd\x10:\x92\x1b\xbb\xbej\xe9p\xfd\u050a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94Q_0\xbc\x90\xcd\xf4W~\xe4}e\u05c5\xfb\xe2\xe87\u01bc\x8a\x02'\x1b^\x01\x8b\xa0X\x00\x00\u07d4Q`\xeda.\x1bH\xe7??\xc1[\xc42\x1b\x8f#\xb8\xa2K\x89\x1e\x82kB(e\xd8\x00\x00\u07d4Qa\xfdI\xe8G\xf6tU\xf1\u023bz\xbb6\xe9\x85&\r\x03\x89A\rXj
\xa4\xc0\x00\x00\u07d4QiT\x02_\xca&\b\xf4}\xa8\x1c!^\xed\xfd\x84J\t\xff\x89\x14\xb5P\xa0\x13\xc78\x00\x00\u07d4Qi\xc6\n\xeeL\xee\u0444\x9a\xb3mfL\xff\x97\x06\x1e\x8e\xa8\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4Q|uC\r\xe4\x01\xc3A\x03&\x86\x11'\x90\xf4mM6\x9e\x89\x15\b\x94\xe8I\xb3\x90\x00\x00\u07d4Q|\xd7`\x8e]\r\x83\xa2kq\u007f6\x03\xda\xc2'}\u00e4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Q\x86]\xb1H\x88\x19Q\xf5\x12Qq\x0e\x82\xb9\xbe\r~\xad\xb2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Q\x89\x1b,\xcd\xd2\xf5\xa4K*\x8b\u011a]\x9b\xcadw%\x1c\x89\x10\xce\x1d=\x8c\xb3\x18\x00\x00\u07d4Q\x8c\xef'\xb1\x05\x82\xb6\xd1OiH=\u06a0\xdd<\x87\xbb\\\x89
\x86\xac5\x10R`\x00\x00\u07d4Q\xa6\xd6'\xf6j\x89#\u060d`\x94\xc4qS\x80\xd3\x05|\xb6\x89>s\xd2z5\x94\x1e\x00\x00\u07d4Q\xa8\xc2\x166\x02\xa3.\xe2L\xf4\xaa\x97\xfd\x9e\xa4\x14QiA\x89\x03h\xf7\xe6\xb8g,\x00\x00\u07d4Q\xb4u\x8e\x9e\x14P\xe7\xafBh\xc3\u01f1\xe7\xbdo\\uP\x8965\u026d\xc5\u07a0\x00\x00\u07d4Q\u028b\xd4\xdcdO\xacG\xafgUc\u0540J\r\xa2\x1e\xeb\x89*\xb7\xb2`\xff?\xd0\x00\x00\u07d4Q\xd2K\xc3so\x88\xddc\xb7\"
&\x88f0\xb6\ub1cd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Q\u05cb\x17\x8dp~9n\x87\x10\x96\\OA\xb1\xa1\xd9\x17\x9d\x89\x05\xfe\xe2\"\x04\x1e4\x00\x00\u07d4Q\xe3/\x14\xf4\xca^(|\xda\xc0W\xa7y^\xa9\xe0C\x99S\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4Q\xe4?\xe0\xd2\\x(`\xaf\x81\xea\x89\xddy<\x13\xf0\u02f1\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4Q\xe7\xb5\\/\x98
\xee\xd78\x846\x1bPf\xa5\x9boE\u0189lk\x93[\x8b\xbd@\x00\x00\xe0\x94Q\xea\x1c\t4\xe3\xd0@\"\ud715\xa0\x87\xa1P\xefp^\x81\x8a\x01Tp\x81\xe7\"M
\x00\x00\u07d4Q\xee\f\xca;\xcb\x10\xcd>\x987\"\xce\xd8I=\x92l\bf\x8965f3\xeb\xd8\xea\x00\x00\xe0\x94Q\xf4f:\xb4O\xf7\x93E\xf4'\xa0\xf6\xf8\xa6\u0225?\xf24\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4Q\xf5^\xf4~dV\xa4\x18\xab2\xb9\"\x1e\xd2}\xbaf\b\xee\x89\u3bb5sr@\xa0\x00\x00\xe0\x94Q\xf9\xc42\xa4\xe5\x9a\xc8b\x82\u05ad\xabL.\xb8\x91\x91`\xeb\x8ap;[\x89\u00e6\xe7@\x00\x00\u07d4R\x0ff\xa0\xe2e\u007f\xf0\xacA\x95\xf2\xf0d\xcf/\xa4\xb2BP\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4R\x10#T\xa6\xac\xa9]\x8a.\x86\xd5\u07bd\xa6\xdei4`v\x89lk\x93[\x8b\xbd@\x00\x00\u07d4R\x13\xf4Y\xe0x\xad:\xb9Z\t
#\x9f\xcf\x163\xdc\x04\u0289\x8c\xf2\x18|*\xfb\x18\x80\x00\u07d4R\x15\x18;\x8f\x80\xa9\xbc\x03\xd2l\xe9\x12\a\x83*\r9\xe6
\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94R!Cx\xb5@\x04\x05j|\xc0\x8c\x89\x13'y\x8a\u01b2H\x8a\x037\xfe_\xea\xf2\u0440\x00\x00\xe0\x94R##\xaa\xd7\x1d\xbc\x96\xd8Z\xf9\x0f\bK\x99\xc3\xf0\

x9d\ucdca\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4R>\x14\r\xc8\x11\xb1\x86\xde\xe5\xd6\u020b\xf6\x8e\x90\xb8\xe0\x96\xfd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4R?mdi\x0f\xda\u0354(SY\x1b\xb0\xff
\xd3em\x95\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4RO\xb2\x10R,^#\xbbg\u07ff\x8c&\xaaam\xa4\x99U\x8965b\xa6m4#\x80\x00\u07d4RU\xdci\x15ZE\xb9p\xc6\x04\xd3\x00G\xe2\xf50i\x0e\u007f\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4R`\xdcQ\xee\a\xbd\u06ab\xab\xb9\xeetK9<\u007fG\x93\xa6\x89\x01\xd8f_\xa5\xfaL\x00\x00\u07d4Rg\xf4\xd4\x12\x92\xf3p\x86<\x90\u05d3)i\x03\x846%\u01c9K\xe4\xe7&{j\xe0\x00\x00\u07d4Rk\xb53\xb7n
\xc8\xee\x1e\xbf\x12?\x1e\x9f\xf4\x14\x8e@\xbe\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4Rl\xb0\x9c\u3b63g.\xec\x1d\xebF
[\xe8\x9aKV>\x89\x85\xcaa[\xf9\xc0\x10\x00\x00\u07d4Rs\x8c\x90\xd8`\xe0L\xb1/l\x8d\x96\xfd\xb5\xbf6\xfc4\x0e\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\u07d4Rz\x8c\xa1&\x863\xa6\xc99\xc5\xde\x1b\x92\x9a\ue4ae\xac\x8d\x890\xca\x02O\x98{\x90\x00\x00\u07d4R\x81\x01\xceF\xb7
\xa2!M\u036ef\x18\xa51w\xff\xa3w\x89\x1b\x96\x12\xb9\xdc\x01\xae\x00\x00\xe0\x94R\x81s4s\xe0\r\x87\xf1\x1e\x99U\u5275\x9fJ\u008ez\x8a\x8b\xd6/\xf4\xee\xc5Y
\x00\x00\u07d4R\x98\xab\x18*\x195\x9f\xfc\xec\xaf\xd7\u0475\xfa!-
\xed\xe6\u0749\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4R\x9a\xa0\x02\u0196*:\x85E\x02\u007f\u0630_\"\xb5\xbf\x95d\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4R\x9e\x82O\xa0rX+@2h:\xc7\xee\xcc\x1c\x04\xb4\xca\xc1\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94R\xa5\xe4\xdeC\x93\xee\xcc\xf0X\x1a\xc1\x1bR\u0183\xc7n\xa1]\x8a\x04<0\xfb\b\x84\xa9l\x00\x00\u07d4R\xb4%|\xf4\x1bn(\x87\x8dP\xd5{\x99\x91O\xfa\x89\x87:\x89\xd5\r\u026a,Aw\x00\x00\u07d4R\xb8\xa9Y&4\xf70\v|\\Y\xa34[\x83_\x01\xb9\\\x89lk\x93[\x8b\xbd@\x00\x00\u07d4R\xbd\u066fYx\x85\v\xc2A\x10q\x8b7#u\x9bC~Y\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4R\xcd
@;\xa7\xed\xa6\xbc0z=c\xb5\x91\x1b\x81|\x12c\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u0794R\u04c0Q\x1d\xf1\x9d^\u0080{\xbc\xb6vX\x1bg\xfd7\xa3\x88\xb9\xf6]\x00\xf6<\x00\x00\xe0\x94R\xe1s\x13P\xf9\x83\xcc,A\x89\x84/\xde\x06\x13\xfa\xd5\f\xe1\x8a\x02w\x01s8\xa3\n\xe0\x00\x00\u07d4R\xe4g\x832\x9av\x93\x01\xb1u\x00\x9d4gh\xf4\xc8~\xe4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4R\xf0X\xd4aG\xe9\x00m)\xbf,\t0J\xd1\xcd\xddn\x15\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4R\xf1T#2<$\xf1\x9a\xe2\xabg7\x17\"\x9d?t}\x9b\x897\xa04\xcb\xe8\xe3\xf3\x80\x00\u07d4R\xf8\xb5\t\xfe\xe1\xa8t\xabo\x9d\x876\u007f\xbe\xaf\x15\xac\x13\u007f\x8965\u026d\xc5\u07a0\x00\x00\u07d4R\xfbF\xac]\x00\xc3Q\x8b,:\x1c\x17}D/\x81eU_\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4S\x00w\xc9\xf7\xb9\a\xff\x9c\xec\fw\xa4\x1ap\xe9\x02\x9a\xddJ\x89lk\x93[\x8b\xbd@\x00\x00\u07d4S\x03\x19\xdb\n\x8f\x93\xe5\xbb}M\xbfH\x161O\xbe\xd86\x1b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4S\x04}\u022c\x90\x83\xd9\x06r\xe8\xb3G<\x10\f\xcd'\x83#\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4S\va\xe4/9Bm$\b\xd4\bR\xb9\xe3J\xb5\xeb\xeb\u0149\x0e~\xeb\xa3A\vt\x00\x00\u07d4S\x0f\xfa\u00fc4\x12\xe2\xec\x0e\xa4{y\x81\xc7p\xf5\xbb/5\x89\a?u\u0460\x85\xba\x00\x00\u07d4S\x17\xec\xb0#\x05,\xa7\xf5e+\xe2\xfa\x85L\xfeEc\xdfM\x89\x1b\x1a\xb3\x19\xf5\xecu\x00\x00\u07d4S\x19M\x8a\xfa>\x885\x02v~\xdb\xc3\x05\x86\xaf3\xb1\x14\u04c9lk\x93[\x8b\xbd@\x00\x00\u07d4S*}\xa0\xa5\xadt\aF\x8d;\xe8\xe0~i\xc7\xddd\xe8a\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4S-2\xb0\x0f0[\xcc$\xdc\xefV\x81}b/4\xfb,$\x89a\x94\x04\x9f0\xf7
\x00\x00\u07d4S4DX@\x82\xeb\xa6T\xe1\xad0\xe1Is\\o{\xa9\"\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4S8\xefp\xea\xc9\u075a\xf5\xa0P;^\xfa\xd1\x03\x9eg\xe7%\x89\x90\xf54`\x8ar\x88\x00\x00\x

e0\x94S9oJ&\u00b4`D\x960lTB\xe7\xfc\xba'.6\x8a\x04?/\b\xd4\x0eZ\xfc\x00\x00\xe0\x94S:s\xa4\xa2\"\x8e\xee\x05\xc4\xff\xd7\x18\xbb\xf3\xf9\xc1\xb1)\xa7\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4S<\x06\x92\x8f\x19\u0429V\xcc(\x86k\xf6\xc8\xd8\xf4\x19\x1a\x94\x89\x0f\xd8\xc1C8\xe60\x00\x00\u07d4S@e6\x1c\xb8T\xfa\xc4+\xfb\\\x9f\xcd\xe0`J\xc9\x19\u0689lk\x93[\x8b\xbd@\x00\x00\u07d4SC\u007f\xec\xf3J\xb9\xd45\xf4\u07b8\xca\x18\x15\x19\xe2Y
5\x89\n1\x06+\xee\xedp\x00\x00\u07d4SR\x01\xa0\xa1\xd74\"\x80\x1fU\xde\xd4\u07ee\xe4\xfb\xaan;\x89\x02&!\x1fy\x15B\x80\x00\xe0\x94S`\x81\x05\xceK\x9e\x11\xf8k\xf4\x97\xff\xca;x\x96{_\x96\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4SnM\x80)\xb7?Uy\u0723>p\xb2N\xba\x89\xe1\x1d~\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4Sp\rS%MC\x0f\"x\x1aJv\xa4c\x93;]k\b\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94S\u007f\x9dM1\xefp\x83\x9d\x84\xb0\xd9\u0377+\x9a\xfe\xdb\xdf5\x8a\x0e\u04b5%\x84\x1a\xdf\xc0\x00\x00\xe0\x94S\x81D\x85\x03\xc0\xc7\x02T+\x1d\xe7\xcc_\xb5\xf6\xab\x1c\xf6\xa5\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94S\x94.yI\xd6x\x8b\xb7\x80\xa7\xe8\xa0y'\x81\xb1aK\x84\x8a\x03]\xebFhO\x10\xc8\x00\x00\u07d4S\x95\xa4E]\x95\xd1x\xb4S*\xa4r[\x19?\xfeQ)a\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94S\x98\x9e\xd30V?\xd5}\xfe\u027d4<7`\xb0y\x93\x90\x8a\x01P\x89N\x84\x9b9\x00\x00\x00\u07d4S\xa2Dg(\x95H\x0fJ+\x1c\xdf}\xa5\xe5\xa2B\xecM\xbc\x8965\u026d\xc5\u07a0\x00\x00\u07d4S\xa7\x14\xf9\x9f\xa0\x0f\xefu\x8e#\xa2\xe7F2m\xad$|\xa7\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4S\xaf2\xc2/\uf640?\x17\x8c\xf9\v\x80/\xb5q\xc6\x1c\xb9\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4S\xc0\xbb\u007f\u020e\xa4\"\xd2\xef~T\x0e-\x8f(\xb1\xbb\x81\x83\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94S\xc5\xfe\x01\x19\xe1\xe8Hd\f\xee0\xad\ua594\x0f*]\x8b\x8a\x04\x9a\xda_\xa8\xc1\f\x88\x00\x00\u07d4S\xc9\xec\xa4\ts\xf6;\xb5\x92{\xe0\xbcj\x8a\x8b\xe1\x95\x1ft\x89lk\x93[\x8b\xbd@\x00\x00\u07d4S\u0388\xe6lZ\xf2\U0009bf4fY*V\xa3\xd1_
l2\x89\a\xa2\x8c1\xcc6\x04\x00\x00\u07d4S\xce\xc6\u0200\x92\xf7V\xef\xe5o}\xb1\x12(\xa2\xdbE\xb1\"\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4S\xe3[\x12#\x1f\x19\xc3\xfdwL\x88\xfe\xc8\xcb\xee\xdf\x14\b\xb2\x89\x1b\xc1mgN\xc8\x00\x00\x00\u07d4S\xe4\xd9im\xcb?M{?p\u072aN\xec\xb7\x17\x82\xff\\\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4S\xfa\xf1e\xbe\x03\x1e\xc1\x830\xd9\xfc\xe5\xbd\x12\x81\xa1\xaf\b\u06c9\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4T\n\x18\x19\xbd|5\x86\x1ey\x18\x04\xe5\xfb\xb3\xbc\x97\u026b\xb1\x89N\xd7\xda\xc6B0
\x00\x00\xe0\x94T\f\a(\x02\x01N\xf0\xd5a4Z\xecH\x1e\x8e\x11\xcb5p\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94T\f\xf2=\xd9\\MU\x8a'\x9dw\x8d+75\xb3\x16A\x91\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4T\x10`\xfcX\xc7P\xc4\x05\x12\xf83i\xc0\xa63@\xc1\"\xb6\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4T\x13\xc9\u007f\xfaJn*{\xba\x89a\u071f\u03850\xa7\x87\u05c965\u026d\xc5\u07a0\x00\x00\u07d4T\x1d\xb2\n\x80\xcf;\x17\xf1b\x1f\x1b?\xf7\x9b\x88/P\xde\xf3\x8965\u026d\xc5\u07a0\x00\x00\u07d4T.\x80\x96\xba\xfb\x88\x16&\x06\x00.\x8c\x8a>\u0458\x14\xae\xac\x89lk\x93[\x8b\xbd@\x00\x00\u07d4T1\v:\xa8\x87\x03\xa7%\u07e5}\xe6\xe6F\x93Qd\x80,\x89g\x8a\x93b\xe4\x18\x00\x00\u07d4T1\xb1\u0447Q\xb9\x8f\xc9\u220a\xc7u\x9f\x155\xa2\xdbG\x89lk\x93[\x8b\xbd@\x00\x00\u07d4T1\xcaB~ae\xa6D\xba\xe3&\xbd\tu\n\x17\x8ce\r\x89lk\x93[\x8b\xbd@\x00\x00\u07d4T5\xc6\xc1y3\x17\xd3,\xe1;\xbaLO\xfe\xb9s\xb7\x8a\u0709\r\x8ek\x1c\x12\x85\xef\x00\x00\xe0\x94T6)\xc9\\\xde\xf4(\xad7\xd4S\u02958\xa9\xf9\t\x00\xac\x8a\t(\x96R\x9b\xad\u0708\x00\x00\u07d4T9\x1bM\x17mGl\xea\x16N_\xb55\u0197\x00\xcb%5\x89\x05l\xd5_\xc6M\xfe\x00\x00\xe0\x94T:\x8c\x0e\xfb\x8b\xcd\x15\xc5C\u29a4\xf8\aYv1\xad\xef\x8a\x01?\x80\xe7\xe1O-

D\x00\x00\u07d4T?\x8cgN$b\xd8\xd5\u06a0\xe8\x01\x95\xa8p\x8e\x11\xa2\x9e\x89\x03wX\x83;:z
\x00\x00\xe0\x94TK[5\x1d\x1b\xc8.\x92\x97C\x99H\xcfHa\xda\u026e\x11\x8a\x04\xa8\x9fT\xef\x0
1!\xc0\x00\x00\u07d4TM\xdaB\x1d\xc1\xebs\xbb$\xe3\xe5j$\x80\x13\xb8|\x0fD\x89j\xcb=\xf2~\x1f\
x88\x00\x00\u07d4TW\\1\x14u\x1e<c\x19q\xdaj*\x02\xfd?\xfb\xfc\u0209i*\xe8\x89p\x81\xd0\x00\x
00\u07d4T[\xb0p\xe7\x81\x17.\xb1`\x8a\xf7\xfc(\x95\xd6\u02c7\x19~\x89y\xa5\xc1~\xc7H\x90\x00
\x00\xe0\x94Tu\xd7\xf1t\xbd\xb1\xf7\x89\x01||\x17\x05\x98\x96F\a\x9dI\x8a\x01\xfd\x934\x94\xaa
_\xe0\x00\x00\u07d4T\x85X\u040c\xfc\xb1\x01\x18\x1d\xac\x1e\xb6\tKN\x1a\x89o\xa6\x89lj\xccg\
u05f1\xd4\x00\x00\u07d4T\x93\x9f\xf0\x89!\xb4g\xcf)Fu\x1d\x85cx)lc\xed\x8965\u026d\xc5\u07a0\
x00\x00\u07d4T\x9bGd\x9c\xfa\u0653\xe4\x06M&6\xa4\xba\xa0b3\x05\u0309
\x9d\x92/RY\xc5\x00\x00\u07d4T\x9dQ\xaf)\xf7$\xc9g\xf5\x94#\xb8[&\x81\xe7\xb1Q6\x89\xcb\xd4
{n\xaa\x8c\xc0\x00\x00\u07d4T\xa17\x01\x16\xfe\"\t\x9e\x01]\a\xcd&i\xdd)\x1c\xc9\u0449\x01\x15\
x8eF\t\x13\xd0\x00\x00\xe0\x94T\xa6+\xf9#>\x14o\xfe\u00c7nE\xf2\x0e\xe8AJ\u07ba\x8a\x02\x1e
\x19\xe0\u027a\xb2@\x00\x00\u07d4T\xb4B\x9b\x18/\x03w\xbe~bi9\xc5\xdbd@\xf7]z\x89j\xcb=\xf
2~\x1f\x88\x00\x00\u07d4T\xbc\xb8\xe7\xf7<\xda=s\xf4\u04cb-
\bG\xe6\x00\xba\r\xf8\x89:pAX\x82\xdf\x18\x00\x00\u07d4T\xc9>\x03\xa9\xb2\xe8\xe4\xc3g(5\xa9
\xeev\xf9a[\xc1N\x89\x01\r:\xa56\xe2\x94\x00\x00\u07d4T\u0388'YV\xde\xf5\xf9E\x8e;\x95\xde\xc
a\xcdH@!\xa0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4T\xdb^\x06\xb4\x81]1\xcbV\xa8q\x9b\xa3:\xf2\x
d7>rR\x89$R\x1e*0\x17\xb8\x00\x00\xe0\x94T\xe0\x12\x83\u030b8E8\xdddgp\xb3W\xc9`\xd6\xca
\u034a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4T\xecs\x00\xb8\x1a\xc8C3\xed\x1b\x03<\xd5\u05e39r\xe24\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4T\xfe\xbc\xce
\xfez\x90\x98\xa7U\xbd\x90\x98\x86\x02\xa4\x8c\b\x9e\x89\"\xb1\xc8\xc1\"z\x00\x00\x00\u07d4U\
n\xad\xae\x12!\xb0z\xfe\xa3\x9f\xba.\xd6.\x05\u5df5\xf9\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d
4U\f0o\x81\xef]\x95\x80\xc0l\xb1\xab
\x1b\x95\xc7H\xa6\x91\x89$\x17\xd4\xc4p\xbf\x14\x00\x00\xe0\x94U\x19\x99\xdd\xd2\x05V3'\xb9\
xb50xZ\xcf\xf9\xbcs\xa4\xba\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4U\x1ew\x84w\x8e\xf8\xe
0H\xe4\x95\xdfI\xf2aO\x84\xa4\xf1\u0709
\x86\xac5\x10R`\x00\x00\xe0\x94U)\x83\na\xc1\xf1<\x19~U\v\xed\xdf\u05bd\x19\\\x9d\x02\x8a\x0
2\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4U)\x87\xf0e\x1b\x91[.\x1eS(\xc1!\x96\rK\xddj\xf4\x89a\t
=|,m8\x00\x00\u07d4U;k\x1cW\x05\x0e\x88\xcf\f1\x06{\x8dL\xd1\xff\x80\xcb\t\x89\x15\xaf\x1dx\xb
5\x8c@\x00\x00\u07d4U?7\xd9$fU\x0e\x9f\xd7u\xaet6-
\xf00\x17\x912\x89lk\x93[\x8b\xbd@\x00\x00\u07d4UC6\xeeN\xa1U\xf9\xf2O\x87\xbc\xa9\xcar\xe
2S\xe1,\u0489\x05k\xc7^-
c\x10\x00\x00\u0794UC\xddm\x16\x9e\xec\x8a!;\xbfz\x8a\xf9\xff\xd1]O\xf7Y\x88\xfc\x93c\x92\x80\
x1c\x00\x00\u07d4UG\xfd\xb4\xae\x11\x95>\x01)+x\a\xfa\x92#\xd0\xe4`j\x89\x05]\x11}\xcb\x1d&\
x00\x00\u07d4UR\xf4\xb3\xed>\x1d\xa7\x9a/x\xbb\x13\xe8\xaeZh\xa9\xdf;\x8965\u026d\xc5\u07a
0\x00\x00\u07d4U\\\xa9\xf0\\\xc14\xabT\xae\x9b\xea\x1c?\xf8z\xa8Q\x98\u0289\x05k\xc7^-
c\x10\x00\x00\xe0\x94U]\x8d<\xe1y\x8a\u0290'T\xf1d\xb8\xbe*\x022\x9cl\x8a\x02\x1e\x19\xe0\u0
27a\xb2@\x00\x00\u07d4U]\xf1\x93\x90\xc1m\x01)\x87r\xba\xe8\xbc:\x11R\x19\x9c\xbd\x89\n\u0
5ce\xbcZ\xc6
\x00\x00\u07d4U^\xbe\x84\u06a4+\xa2V\xeax\x91\x05\xce\u0136\x93\xf1/\x18\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94U\u007f^e\xe0\xda3\x99\x82\x19\xadN\x99W\x05E\xb2\xa9\xd5\x11\x8a\x0

2U\x9c\xbb\x98XB@\x00\x00\u07d4U\x83`
h\x83\xdd\x1bmJYc\x9eV)\xd0\xf0\xc6u\u0409lk\x93[\x8b\xbd@\x00\x00\u07d4U\x84B0P\xe3\xc2
\x05\x1f\v\xbd\x8fD\xbdm\xbc'\xec\xb6,\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4U\x85)CI)p\xf8\xd6
)\xa1Sf\xcd\xda\x06\xa9OE\x13\x89lk\x93[\x8b\xbd@\x00\x00\u0794U\x86d\x86\xec\x16\x8fy\xdb\
xe0\u1af1\x88d\u0649\x91\xae,\x88\xdfn\xb0\xb2\xd3\xca\x00\x00\u07d4U\x8cTd\x9a\x8an\x94r+\
xd6\xd2\x1d\x14qOqx\x054\x89lk\x93[\x8b\xbd@\x00\x00\u07d4U\x91\x940O\x14\xb1\xb9:\xfeDO
\x06$\xe0S\xc2:\x00\t\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4U\x93\xc9\u0536ds\x0f\xd9<\xa6
\x01Q\xc2\\.\xae\xd9<;\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4U\x97\x06\xc32\xd2\ay\xc4_\x8am\x04ji\x91Y\xb7I!\x89\x14\x9bD.\x85\xa3\u03c0\
x00\u07d4U\x98\xb3\xa7\x9aH\xf3+\x1f_\xc9\x15\xb8{d]\x80]\x1a\xfe\x89\x1b\x1a\xe4\xd6\xe2\xef
P\x00\x00\u07d4U\xa3\xdfW\xb7\xaa\xec\x16\xa1b\xfdS\x16\xf3[\xec\b(!\u03c9j\xcb=\xf2~\x1f\x88\
x00\x00\u07d4U\xa4\xca\xc0\u02cbX-
\x9f\xef8\xc5\xc9\xff\xf9\xbdS\t=\x1f\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4U\xa6\x1b\x10\x94\x80\
xb5\xb2\xc4\xfc\xfd\xef\x92\xd9\x05\x84\x16\f\r5\x89\x02lVM+S\xf6\x00\x00\u07d4U\xaa]1>\xbb\b
M\xa0\xe7\x80\x10\x91\u2792\xc5\xde\u00ea\x89lk\x93[\x8b\xbd@\x00\x00\u07d4U\xab\x99\xb0\
xe0\xe5]{\xb8t\xb7\xcf\xe84\xdec\x1c\x97\xec#\x897\xe9\x8c\xe3h\x99\xe4\x00\x00\u07d4U\xaf\t/\
x94\xbajy\x91\x8b\f\xf99\xea\xb3\xf0\x1b?Q\u01c9\b
\xd5\xe3\x95v\x12\x00\x00\u07d4U\xc5dfAf\xa1\xed\xf3\x91>\x01i\xf1\xcdE\x1f\xdb]\f\x89\x82\x17
\xeaIP\x8el\x00\x00\xe0\x94U\xcaj\xbey\xea$\x97\xf4o\u06f804`\x10\xfeF\x9c\xbe\x8a\x016\x9f\xb
9a(\xacH\x00\x00\u07d4U\xca\xffK\xba\x04\xd2
\u0265\xd2\x01\x86r\xec\x85\xe3\x1e\xf8>\x89lk\x93[\x8b\xbd@\x00\x00\u07d4U\xd0W\xbc\xc0K\
xd0\xf4\xaf\x96BQ:\xa5\t\v\xb3\xff\x93\xfe\x89;\xfeE,\x8e\xddL\x00\x00\u07d4U\xd4.\xb4\x95\xbfF\
xa64\x99{_.\xa3b\x81I\x18\u2c09\x05\xc0\xd2e\xb5\xb2\xa8\x00\x00\u07d4U\u069d\xcd\xcaa\xcb\
xfe\x1f\x13<{\xce\xfc\x86{\x9c\x81\"\xf9\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4U\xe2
\x87bb\xc2\x18\xafOVxG\x98\xc7\xe5]\xa0\x9e\x91\x89\a=\x99\xc1VE\xd3\x00\x00\u07d4U\xfd\b\
u0440d\xbd
,\x0e\xc3\xd2\xcc\xe0\xce\v\x9d\x16\x9cM\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4V\x00s\nU\xf6\xb
2\x0e\xbd$\x81\x1f\xaa=\xe9m\x16b\xab\xab\x89e\xea=\xb7UF`\x00\x00\u07d4V\x03$\x1e\xb8\xf
0\x8fr\x1e4\x8c\x9d\x9a\xd9/H\u342a$\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4V\x056yJ\x9e+\x00I\xd1\x023\xc4\x1a\xdc_A\x8a&J\x8965\u026d\xc5\u07a0\x00\x
00\u07d4V\aY\x00Y\xa9\xfe\xc1\x88\x11I\xa4K6\x94\x9a\xef\x85\xd5`\x89lk\x93[\x8b\xbd@\x00\x
00\u07d4V\v\xec\xdfR\xb7\x1f=\x88'\xd9'a\x0f\x1a\x98\x0f3qo\x89\x17GMp_V\u0400\x00\xe0\x94
V\r\xa3~\x95m\x86/\x81\xa7_\u0540\xa7\x13\\\x1b$cR\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\
xe0\x94V\x0f\xc0\x8d\a\x9f\x04~\xd8\xd7\xdfuU\x1a\xa55\x01\xf5p\x13\x8a\x01\x9b\xff/\xf5yh\xc0\
x00\x00\u07d4V\x1b\xe9)\x9b>k>c\xb7\x9b\t\x16\x9d\x1a\x94\x8a\xe6\xdb\x01\x89\x1b\x1a\xe4\x
d6\xe2\xefP\x00\x00\xe0\x94V \xe3\xedy-
/\x185\xfe_UA}Q\x11F\fj\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4V
\xf4m\x14Q\xc25=bC\xa5\u0534'\x13\v\xe2\xd4\a\x89\x03@\xaa\xd2\x1b;p\x00\x00\xe0\x94V!\x05
\xe8+\t\x975\xdeI\xf6&\x92\u0307\xcd8\xa8\xed\u034a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94
V*\x8d\u02fe\xee\xf7\xb3`h]'0;\u059e\tJ\xcc\xf6\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4
V+\xce\u04ca\xb2\xabl\b\x0f;\x05A\xb8Enp\x82K?\x89\"\xca5\x87\xcfN\xb0\x00\x00\xe0\x94V+\xe
9Z\xba\x17\xc57\x1f\u2e82\x87\x99\xb1\xf5]!w\u058a\b\x16\xd3~\x87\xb9\xd1\xe0\x00\x00\u07d4

V/\x16\u05da\xbf\xce\u00d4>4\xb2\x0f\x05\xf9{\xdf\u0366\x05\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4V7=\xaa\xb4c\x16\xfd~\x15v\xc6\x1ej\xff\xcbeY\xdd\u05c9\v\xacq]\x14l\x9e\x00\x00\u07d4V9v8\xbb<\xeb\xf1\xf6
byK^\xb9B\xf9\x16\x17\x1d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4V:\x03\xab\x9cV\xb6\x00\xf6\xd2[f\f!\xe1c5Qzu\x8965\u026d\xc5\u07a0\x00\x00\u07d4V<\xb8\x80<\x1d2\xa2['\xb6A\x14\x85+\xd0M\x9c
\u0349\v\x14\x9e\xad\n\xd9\xd8\x00\x00\u07d4VXc\x91\x04\fW\xee\xc6\xf5\xaf\xfd\x8c\u052b\xde\x10\xb5\n\u0309\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4Vl\x10\xd68\u8e0bG\xd6\xe6\xa4\x14Iz\xfd\xd0\x06\x00\u0509\x05k9Bc\xa4\f\x00\x00\u07d4Vl(\xe3L8\b\xd9vo\xe8B\x1e\xbfO+\x1cO}w\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4V\x8d\xf3\x18Vi\x9b\xb5\xac\xfc\x1f\xe1\u0580\u07d9`\xcaCY\x89J\xcfUR\xf3\xb2I\x80\x00\u07d4V\x91\xdd/gE\xf2\x0e\"\xd2\xe1\u0479U\xaa)\x03\xd6VV\x89j\xc5\xc6-
\x94\x86\a\x00\x00\u07d4V\xa1\xd6\r@\xf5\u007f0\x8e\xeb\xf0\x87\xde\xe3\xb3\u007f\x1e|,\xba\x89>\u072e\xc8-\x06\xf8\x00\x00\u07d4V\xac
\xd6;\xd8\x03Y\\\xec\x03m\xa7\xed\x1d\xc6n\n\x9e\a\x89\x03w*S\xcc\xdce\x80\x00\u07d4V\xb6\xc2=\xd2\uc434r\x8f;\xb2\xe7d\xc3\xc5\f\x85\xf1D\x8965\u026d\xc5\u07a0\x00\x00\u07d4V\xdf\x05\xba\xd4l?\x00\xaeGn\xcf\x01{\xb8\xc8w8?\xf1\x89\n\xb1]\xaa\xefp@\x00\x00\u07d4V\xee\x19\u007fK\xbf\x9f\x1b\x06b\xe4\x1c+\xbd\x9a\xa1\xf7\x99\xe8F\x8965\u026d\xc5\u07a0\x00\x00\u07d4V\xf4\x93\xa3\xd1\b\xaa\xa2\u044d\x98\x92/\x8e\xfe\x16b\u03f7=\x89m\x81!\xa1\x94\xd1\x10\x00\x00\u07d4V\xfc\x1a{\xad@G#|\xe1\x16\x14b\x96#\x8e\a\x8f\x93\xad\x89\t\xa6?\b\xeac\x88\x00\x00\u07d4V\xfe\xbf\x9e\x10\x03\xaf\x15\xb1\xbdI\a\xec\b\x9aJ\x1b\x91\xd2h\x89\n\u05ce\xbcZ\xc6\x00\x00\u07d4W\x17\u0313\x01Q\x1dJ\x81\xb9\xf5\x83\x14\x8b\xee\xd3\xd3\u0303\t\x89\x8c\xf2?\x90\x9c\x0f\xa0\x00\x00\u07d4W\x17\xf2\xd8\xf1\x8f\xfc\xc0\xe5\xfe$}:B\x19\x03|:d\x9c\x89\u063beI\xb0+\xb8\x00\x00\u07d4W\x19P\xea,\x90\xc1B}\x93\x9da\xb4\xf2\xdeL\xf1\u03ff\xb0\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4W\x19\xf4\x9br\r\xa6\x88V\xf4\xb9\xe7\b\xf2VE\xbd\xbcKA\x89\"\xb1\xc8\xc1\"z\x00\x00\x00\u07d4W*\xc1\xab\xa0\xde#\xaeA\xa7\xca\xe1\xdc\bB\u062b\xfc\x10;\x89g\x8a\x93 b\xe4\x18\x00\x00\xe0\x94W-\xd8\xcd?\xe3\x99\xd1\xd0\xec(\x121\xb7\xce\xfc
\xb9\u4eca\x023\xc8\xfeBp>\x80\x00\x00\xe0\x94WI!\x83\x8c\xc7}l\x98\xb1}\x90::\xe0\xee\r\xa9[\u040a\vS(\x17\x8a\xd0\xf2\xa0\x00\x00\u07d4WJ\xd95S\x90\u421e\xf4*\xcd\x13\x8b*'\xe7\x8c\x00\xae\x89Tg\xb72\xa9\x134\x00\x00\u07d4WM\xe1\xb3\xf3\x8d\x91XF\xae7\x18VJZ\xda
\xc2\xf3\xed\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94W\\\x00\u0081\x82\x10\u0085U\xa0\xff)\x01\x02\x89\xd3\xf8#\t\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94Ws\xb6\x02g!\xa1\xdd\x04\xb7\x82\x8c\xd6+Y\x1b\xfb4SL\x8a\x05\xb7\xacES\xdez\xe0\x00\x00\xe0\x94WwD\x1c\x83\xe0?\v\xe8\xdd4\v\xdechP\x84|b\v\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4Wx\xff\u071b\x94\u0165\x9e\"N\xb9e\xb6\u0790\xf2\"\xd1p\x89\x12-
\u007f\xf3f\x03\xfc\x00\x00\u07d4Wz\xee\xe8\u053c\b\xfc\x97\xab\x15n\xd5\u007f\xb9p\x92Sf\xbe\x89\x12\r\xf1\x14rX\xbf\x00\x00\u07d4W{-
\a<Y\fP0o[\x11\x95\xa4\xb2\xba\x9e\u0366%\x89\x14@\xbd\u0515\x15\xf0\x00\x00\u07d4W{\xfed\xe3\xa1\xe3\x80\x0e\x94\xdb\x1cl\x18M\x8d\u022a\xfcf\x89Q4\xed\x17A\u007f(\x00\x00\u07d4W\x82Z\xeb\t\al\xaaGx\x87\xfb\u026e7\xe8\xb2|\xc9b\x89\x05k\xc7^-
c\x10\x00\x00\u07d4W\x880\x10\xb4\xac\x85\u007f\xed\xac\x03\xea\xb2U\x17#\xa8D\u007f\xfb\x8965\u026d\xc5\u07a0\x00\x00\u07d4W\x89\xd0\x1d\xb1,\x81j\xc2h\xe9\xaf\x19\xdc\r\xd6\u065f\x

15\u07c9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4W\x92\x81OY\xa3:\x18C\xfa\xa0\x1b\xaa\b\x9e\xb0/\xfb\\\xf1\x89\x1b\x1a\xb3\x19\
xf5\xecu\x00\x00\u07d4W\x93\xab\xe6\xf1S3\x11\xfdQSh\x91x;?\x96%\xef\x1c\x89,\u0626V\xf2?\
xda\x00\x00\xe0\x94W\x97\xb6\x0f\u0489J\xb3\xc2\xf4\xae\u0786\xda\xf2\xe7\x88\xd7E\xad\x8a\
x01EB\xba\x12\xa37\xc0\x00\x00\u07d4W\xa8R\xfd\xb9\xb1@[\xf5<\u03d5\b\xf82\x99\xd3
lR\x89lk\x93[\x8b\xbd@\x00\x00\u07d4W\xb2=j\x1a\xdc\x06\xc6R\xa7y\u01a7\xfbk\x95\xb9\xfe\xa
df\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4W\xbc
\xe2\xd6+=\x19f<\xdbL0\x9d[O/\xc2\u06cf\x89\x05k\xc7^-
c\x10\x00\x00\u07d4W\xbd\xdf\a\x884\x00\x9c\x89\u060eb\x82u\x9d\xc4S5\xb4p\x89tq|\xfbh\x83\
x10\x00\x00\u07d4W\xbe\xeaql\xbd\x81p\ns\xd6\u007f\x9f\xf09R\x9c-
\x90%\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4W\xd02\xa4=\x16Nq\xaa.\xf3\xff\xd8I\x1b\nN\xf1\xea[\x89lk\x93[\x8b\xbd@\x00\x0
0\u07d4W\xd3\u07c0O+\xee\xe6\xefS\xab\x94\xcb>\xe9\xcfRJ\x18\u04c9\x15Vak\x96\x06g\x00\x
00\u07d4W\xd5\xfd\x0e=0I3\x0f\xfc\xdc\xd0
Ei\x17e{\xa2\u0689k\xf2\x01\x95\xf5T\xd4\x00\x00\u07d4W\u0754q\xcb\xfa&'\t\xf5\U00106f37t\xc
5\xf5'\xb8\xf8\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4W\xdf#\xbe\xbd\xc6^\xb7_\ub732\xfa\xd1\
xc0si++\xaf\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4X\x00\u03410\x83\x9e\x94I]-
\x84\x15\xa8\xea,\x90\xe0\xc5\u02c9\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94X\x03\xe6\x8b4\xda\x12\x1a\xef\b\xb6\x02\xba\u06ef\xb4\xd1$\x81\u028a\x03\xc
f\xc8.7\xe9\xa7@\x00\x00\xe0\x94X\x16\xc2hww\xb6\xd7\u04a2C-
Y\xa4\x1f\xa0Y\xe3\xa4\x06\x8a\x1cO\xe4:\xdb\n^\x90\x00\x00\u07d4X\x1a:\xf2\x97\xef\xa4Cj)\xa
f\x00r\x92\x9a\xbf\x98&\xf5\x8b\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94X\x1b\x9f\xd6\xea\xe3r\xf3P
\x1fB\xeb\x96\x19\xee\xc8
\xb7\x8a\x84\x8a\x04+\xe2\xc0\f\xa5;\x8d\x80\x00\u07d4X\x1b\xdf\x1b\xb2v\xdb\u0746\xae\xdc\x
db9z\x01\xef\xc0\xe0\f[\x8965\u026d\xc5\u07a0\x00\x00\u07d4X\x1f4\xb5#\xe5\xb4\x1c\t\xc8|)\x8
e)\x9c\xbc\x0e)\xd0f\x89=X3\xaa\xfd9u\x80\x00\xe0\x94X$\xa7\xe2(8'q40\x8c_KP\u06b6^C\xbb1\
x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4X+pf\x9c\x97\xaa\xb7\u0581H\xd8\xd4\xe9\x04\x11\x
e2\x81\rV\x8965f3\xeb\xd8\xea\x00\x00\u07d4X.|\xc4o\x1d{Nn\x9d\x95\x86\x8b\xfd7\x05s\x17\x8f
L\x89lk\x93[\x8b\xbd@\x00\x00\u07d4X>\x83\xbaU\xe6~\x13\xe0\xe7o\x83\x92\xd8s\xcd!\xfb\xf7\
x98\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4Xi\xfb\x86}q\xf18\u007f\x86;i\x8d\t\xfd\xfb\x87\u01
1b\\\x89\u01bb\xf8X\xb3\x16\b\x00\x00\u07d4X}hI\xb1h\xf6\xc33+z\xba\xe7\xeblB\xc3\u007fH\xbf\
x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4X\x87\xdcj3\xdf\xedZ\xc1\xed\xef\xe3^\xf9\x1a!b1\xac\x96\x89\r
\x8drkqw\xa8\x00\x00\xe0\x94X\x8e\u0650\xa2\xaf\xf4J\x94\x10]X\xc3\x05%w5\xc8h\xac\x8a\x03
h\xc8b:\x8bM\x10\x00\x00\u07d4X\xae-
\xdc_L\x8a\u0697\xe0l\x00\x86\x17\x17g\xc4#\xf5\u05c9WG=\x05\u06ba\xe8\x00\x00\u07d4X\xa
e\xd6gJ\xff\xd9\xf6B3'*W\x8d\xd98k\x99\xc2c\x89\xb8Pz\x82\a(
\x00\x00\xe0\x94X\xb8\b\xa6[Q\xe63\x89i\xaf\xb9^\xc7\a5\xe4Q\xd5&\x8a\bxK\xc1\xb9\x83z8\x00
\x00\u07d4X\xb8\xae\x8fc\xef5\xed\ab\xf0\xb6#=J\xc1Nd\xb6M\x89lk\x93[\x8b\xbd@\x00\x00\u07
d4X\xba\x15ie\x0e[\xbb\xb2\x1d5\xd3\xe1u\xc0\u05b0\xc6Q\xa9\x89\x1b\x1a\xe4\xd6\xe2\xefP\x0
0\x00\u07d4X\xc5U\xbc)<\xdb\x16\xc66.\xd9z\xe9U\v\x92\xea\x18\x0e\x89\x01\x15\x8eF\t\x13\xd
0\x00\x00\u07d4X\xc6P\xce\xd4\v\xb6VA\xb8\xe8\xa9$\xa09\xde\xf4hT\u07c9\x01\x00\xbd3\xfb\x
98\xba\x00\x00\u07d4X\xc9\aT\xd2\xf2\n\x1c\xb1\xdd3\x06%\xe0KE\xfaa\x9d\\\x89lk\x93[\x8b\xbd

@\x00\x00\xe0\x94X\xe2\xf1\x12#\xfc\x827\xf6\x9d\x99\xc6(\x9c\x14\x8c\x06\x04\xf7B\x8a\x05\x15\n\xe8J\x8c\xdf\x00\x00\x00\u07d4X\xe5T\xaf=\x87b\x96
\xdaa\xd58\xc7\xf5\xb4\xb5LJ\xfe\x89FP\x9diE4r\x80\x00\u07d4X\xe5\xc9\xe3D\xc8\x06e\r\xac\xfc\x90M3\xed\xbaQ\a\xb0\u0789\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\u07d4X\xe6a\u043as\xd6\xcf$\t\x9aUb\xb8\b\xf7\xb3g;h\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94X\xf0[&%`P<\xa7a\xc6\x18\x90\xa4\x03_Lsr\x80\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4X\xfb\x94sd\xe7iWe6\x1e\xbb\x1e\x80\x1f\xfb\x8b\x95\xe6\u0409\n\u05ce\xbcZ\xc6
\x00\x00\u07d4Y\x01\x81\xd4E\x00{\u0407Z\xaf\x06\x1c\x8dQ\x159\x00\x83j\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4Y\x02\xe4J\xf7i\xa8rF\xa2\x1e\a\x9c\b\xbf6\xb0n\xfe\xb3\x8965\u026d\xc5\u07a0\x00\x00\u07d4Y\n\xcb\xda7)\f\r>\xc8O\xc2\x00\rv\x97\xf9\xa4\xb1]\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94Y\f\xcbY\x11\xcfx\xf6\xf6\"\xf55\xc4t7_J\x12\xcf\u03ca\x04<3\xc1\x93ud\x80\x00\x00\u07d4Y\x10\x10m\xeb\u0491\xa1\u0340\xb0\xfb\xbb\x8d\x8d\x9e\x93\xa7\xcc\x1e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Y\x16\x17I\xfe\xdc\xf1\xc7!\xf2 -
\x13\xad\xe2\xab\xcfF\v=\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94Y\x1b\xef1q\xd1\u0155w\x17\xa4\xe9\x8d\x17\xeb\x14,!NV\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4Y
<\xc3u\x99\xb6H1*|\xc9\xe0m\xac\xb5\x89\xa9\xaej\x89\b\x0fyq\xb6@\x0e\x80\x00\u07d4Y&\x81q\xb83\xe0\xaa\x13\xc5KR\xcc\xc0B.O\xa0:\ub262\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94Y'w&\x1e;\xd8R\u010e\u0295\xb3\xa4L[\u007f-
B,\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4Y0Dg\x0f\xae\xff\x00\xa5[Z\xe0Q\xeb{\xe8p\xb1\x16\x94\x89\a?u\u0460\x85\xba\x00\x00\xe0\x94Y;E\xa1\x86J\xc5\xc7\xe8\xf0\u02ae\xba\r\x87<\xd5\xd1\x13\xb2\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4Y<H\x93[\xea\xff\x0f\xde\x19\xb0M0\x9c\xd50\xa2\x8eR\u0389\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4YG<\xd3\x00\xff\xfa\xe2@\xf5xV&\xc6]\xfe\u01d2\xb9\xaf\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4YH\xbc6P\xedQ\x9b\xf8\x91\xa5rg\x9f\u0652\xf8x\fW\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4YJv\xf0i58\x8d\xde^#F\x96\xa0f\x8b\xc2\r-
\u0709\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4YV\x9a!\u048f\xbaK\xda7u4\x05\xa0\x81\xf2\x06=\xa1P\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4YV\xb2\x8e\u01c9\vv\xfc\x06\x1a\x1f\xebR\xd8*\xe8\x1f\xb65\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Y^#\u05c8\xa2\u053b\x85\xa1]\xf7\x13m&JcU\x11\xb3\x89\u0556{\xe4\xfc?\x10\x00\x00\xe0\x94Yp8\xff\x91\xa0\x90\f\xbb\xabH\x8a\xf4\x83\u01d0\xe6\xec\x00\xa0\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94Yp\xfb\x1b\x14M\xd7Q\xe4\xce.\xca|\xaa
\xe3c\xdcM\xa3\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4Yu\xb9R\x8f#\xaf\x1f\x0e.\xc0\x8a\xc8\xeb\xaaxj,\xb8\xe0\x89\x12\xbfPP:\xe3\x03\x80\x00\u07d4Yu\u05cd\x97N\u5edeML\xa2\xaew\xc8K\x9c;K\x82\x89JD\x91\xbdm\xcd(\x00\x00\u07d4Y\x85\u015aD\x9d\xfc]\xa7\x87\xd8$Ntlmp\u02a5_\x89\x05k\xc7^-
c\x10\x00\x00\u07d4Y\x8a\xaa\xba\xe9\xed\x83={\xc2\"\xe9\x1f\u02a0d{wX\v\x89a\x94\x04\x9f0\xf7
\x00\x00\u07d4Y\x92bLT\xcd\xec`\xa5\xae\x93\x803\xaf\x8b\xe0\xc5\f\xbb\n\x89\xc4T\xe0\xf8\x87\x0f+\x00\x00\xe0\x94Y\x97(\xa7\x86\x18\u0461{\x9e4\xe0\xfe\xd8\xe8W\xd5\xc4\x06\"\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4Y\x97\xff\xef\xb3\xc1\xd9\xd1\x0f\x1a\u2b0a\xc3\xc8\xe2\xd2)'\x83\x8965\u026d\xc5\u07a0\x00\x00\u07d4Y\xa0\x87\xb95\x1c\xa4/X\xf3n\x02\x19'\xa2)\x88(O8\x89\x01\x00\xbd3\xfb\x98\xba\x00\x00\u07d4Y\xa1-

\xf2\xe3\xef\x85z\xce\xff\x93\x06\xb3\t\xf6\xa5\x00\xf7\x014\x8965\u026d\xc5\u07a0\x00\x00\u07d
4Y\xb9m\ub1c4\x88]\x8d;J\x16aC\xccC]%U\xa1\x89Hz\x9a0E9D\x00\x00\u07d4Y\xb9\xe73\u02e
4\xbe\x00B\x9bK\xd9\u07e6G2\x05:}U\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94Y\xc5\xd0k\x1
7\x0e\xe4\xd2n\xb0\xa0\xebF\xcb}\x90\xc1\xc9\x10\x19\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x0
0\u07d4Y\xc7\xf7\x85\xc91`\xe5\x80~\xd3N^SK\xc6\x18\x86G\xa7\x89\"\xb1\xc8\xc1\"z\x00\x00\x
00\u07d4Y\xd19\xe2\xe4\f{\x97#\x9d#\u07ec\xa38X\xf6\x02\xd2+\x89lk\x93[\x8b\xbd@\x00\x00\u
07d4Y\xf6${\rX*\xaa%\xe5\x11Ge\xe4\xbf<wOC\u0089\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d
4Y\xfe\x00im\xbd\x87\xb7\x97k)\xd1\x15l\x88B\xa2\xe1y\x14\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x9
4Z\r`\x9a\xae#2\xb17\xab;/&aZ\x80\x8f7\xe43\x8a!\xe1\x9e\f\x9b\xab$\x00\x00\x00\u07d4Z\x19+\
x96J\xfd\x80w>_^\xdajV\xf1N%\xe0\xc6\xf3\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4Z\x1a3ib
\xd6\xe0\xc601\u0303\u01a5\u01a6\xf4G\x8e\u02c965\u026d\xc5\u07a0\x00\x00\u07d4Z\x1d--
\x1dR\x03\x04\xb6
\x88IW\x047\xeb0\x91\xbb\x9f\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4Z&s1\xfa\xcb&-
\xaa\xec\xd9\xddc\xa9p\f_RY\u07c9\x05k\xc7^-
c\x10\x00\x00\xe0\x94Z(WU9\x1e\x91NX\x02_\xaaH\xcch_O\xd4\xf5\xb8\x8a\x05\x81v{\xa6\x18\x
9c@\x00\x00\u07d4Z)\x16\xb8\xd2\xe8\xcc\x12\xe2\a\xabFMC>#p\xd8#\u0649lk\x93[\x8b\xbd@\x
00\x00\u07d4Z+\x1c\x85:\xeb(\xc4U9\xafv\xa0\n\xc2\u0628$(\x96\x89\x01Z\xf1\u05cbX\xc4\x00\x
00\u07d4Z-
\xaa\xb2\\1\xa6\x1a\x92\xa4\xc8,\x99%\xa1\xd2\xefXX^\x89\f8\r\xa9\u01d5\f\x00\x00\u07d4Z0\xfe\
xac7\xac\x9fr\u05f4\xaf\x0f+\xc79R\xc7O\xd5\u00c9lk\x93[\x8b\xbd@\x00\x00\u07d4ZTh\xfa\\\xa2
&\xc7S.\xcf\x06\xe1\xbc\x1cE\"]~\u0249g\x8a\x93
b\xe4\x18\x00\x00\u07d4ZVR\x857JI\xee\xddPL\x95}Q\bt\xd0\x04U\xbc\x89\x05k\xc7^-
c\x10\x00\x00\u07d4Z^\xe8\xe9\xbb\x0e\x8a\xb2\xfe\xcbK3\u0494x\xbeP\xbb\xd4K\x89*\x11)\u04
13g
\x00\x00\xe0\x94Z_\x85\b\xda\x0e\xbe\xbb\x90\xbe\x903\xbdM\x9e'A\x05\xae\x00\x8a\x01je\x02\
xf1Z\x1eT\x00\x00\u07d4Z`q\xbc\xeb\xfc\xbaJ\xb5\u007fM\xb9o\u01e6\x8b\xec\xe2\xba[\x89lk\x9
3[\x8b\xbd@\x00\x00\u07d4Z`\xc9$\x16(s\xfc~\xa4\xda\u007f\x97.5\x01g7`1\x89\x04\x87\xf2w\xa
8\x85y\x80\x00\u07d4Zf\x86\xb0\xf1~\a\xed\xfcY\xb7Y\xc7}[\xef\x16M8y\x89P\xc5\xe7a\xa4D\b\x
00\x00\u07d4Zp\x10o
\xd6?\x87Re\xe4\x8e\r5\xf0\x0e\x17\xd0+\u0249\x01\x15\x8eF\t\x13\xd0\x00\x00\u0794Zt\xbab\x
e7\xc8\x1a4t\xe2}\x89O\xed3\xdd$\xad\x95\xfe\x88\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94Zw5\x0
0}p\xb0hD\u0699\x01\xcd\xfa\xdb\x11\xa2X,/\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4Z\x82\xf
9l\u0537\xe2\xd9=\x10\xf3\x18]\xc8\xf4=Ku\xaai\x89lc?\xba\xb9\x8c\x04\x00\x00\u07d4Z\x87\xf04
\xe6\xf6\x8fNt\xff\xe6\fd\x81\x946\x03l\xf7\u05c9\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94Z\x89\x1
1U\xf5\x0eB\aCt\xc79\xba\xad\xf7\xdf&Q\x15:\x8a\x01\x02\xdao\xd0\xf7:<\x00\x00\u07d4Z\x9c\x8
bi\xfcaMiVI\x99\xb0\r\xcbB\xdbg\xf9~\x90\x89\xb9\xe6\x15\xab\xad:w\x80\x00\xe0\x94Z\xaf\x1c1
%Jn\x00_\xba\u007fZ\xb0\xecy\xd7\xfc+c\x0e\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d4Z\xb1\xa5
aSH\x00\x1c|w]\xc7WHf\x9b\x8b\xe4\xde\x14\x89%jr\xfb)\xe6\x9c\x00\x00\xe1\x94Z\xbf\xec%\xf7
L\u06047c\x1aw1\x90i2wcV\xf9\x8b\t\xd8<\xc0\u07e1\x11w\xff\x80\x00\u07d4Z\u0090\x8b\x0f9\x
8c\r\xf5\xba\xc2\xcb\x13\xcas\x14\xfb\xa8\xfa=\x89\n\xd4\xc81j\v\f\x00\x00\xe0\x94Z\u025a\u05c1
j\xe9\x02\x0f\xf8\xad\xf7\x9f\xa9\x86\x9b|\xeaf\x01\x8a\x04ri\x8bA;C
\x00\x00\u07d4Z\xd1,^\xd4\xfa\x82~!P\u03e0\u058c\n\xa3{\x17i\xb8\x89+^:\xf1k\x18\x80\x00\x00\

xe0\x94Z\xd5\xe4
uV\x13\x88o5\xaaV\xac@>\xeb\xdf\xe4\xb0\u040a\x10\xf0\xcf\x06M\u0552\x00\x00\x00\u07d4Z\xdew\xfd\x81\xc2\\\n\xf7\x13\xb1\a\x02v\x8c\x1e\xb2\xf9u\xe7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4Z\xe6N\x85;\xa0\xa5\x12\x82\u02cd\xb5.Aa^|\x9fs?\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Z\xed\x0el\xfe\x95\xf9\u0580\xc7dr\xa8\x1a+h\n\u007f\x93\xe2\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4Z\xef\x16\xa2&\xddh\a\x1f$\x83\xe1\xdaBY\x83\x19\xf6\x9b,\x89lk\x93[\x8b\xbd@\x00\x00\u07d4Z\xf4j%\xac\t\xcbsakS\xb1O\xb4/\xf0\xa5\x1c\u0772\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4Z\xf7\xc0r\xb2\u016c\xd7\x1cv\xad\xdc\xceS\\\xf7\xf8\xf95\x85\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94Z\xfd\xa9@\\\x8e\x976QEt\u0692\x8d\xe6tV\x01\t\x18\x8a\x01E\xb8\xb0#\x9aF\x92\x00\x00\u07d4[\x06\xd1\xe6\x93\f\x10Ti+y\xe3\xdb\xe6\xec\xceS\x96d
\x89\v\"\u007fc\xbe\x81<\x00\x00\u07d4[%\xca\xe8m\xca\xfa*`\xe7r61\xfc_\xa4\x9c\x1a\xd8}\x89\x87\fXQ\x0e\x85
\x00\x00\u07d4[(|~sB\x99\xe7'bo\x93\xfb\x11\x87\xa6\rPW\xfe\x89\x05|\xd94\xa9\x14\xcb\x00\x00\u07d4[)\f\x01\x96|\x81.M\xc4\xc9\v\x17L\x1b@\x15\xba\xe7\x1e\x89\b
\xeb4\x8dR\xb9\x00\x00\u07d4[+d\xe9\xc0X\u30a8\xb2\x99\"N\xec\xaa\x16\xe0\x9c\x8d\x92\x89\b\xbaR\xe6\xfcE\xe4\x00\x00\xe0\x94[./\x16\x18U.\xab\r\xb9\x8a\xddUc|)Q\xf1\xfb\x19\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4[0`\x8cg\x8e\x1a\xc4d\xa8\x99L;3\xe5\xcd\xf3Iq\x12\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4[36\x96\xe0L\xca\x16\x92\xe7\x19\x86W\x9c\x92\rk)\x16\xf9\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94[C\rw\x96\x96\xa3e?\xc6\x0et\xfb\u02ec\xf6\xb9\u00ba\xf1\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4[Cse\xae:\x9a/\xf9|h\xe6\xf9\nv
\x18\x8c}\x19\x89l\x87T\xc8\xf3\f\b\x00\x00\u07d4[I\xaf\xcduDx8\xf6\xe7\xce\u068d!w}O\xc1\xc3\xc0\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4[L\f`\xf1\x0e\u0489K\xdbB\xd9\xdd\x1d!\x05\x87\x81\n\r\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4[N\xa1m\xb6\x80\x9b\x03R\u0536\xe8\x1c9\x13\xf7jQ\xbb2\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4[[\xe0\xd8\xc6rv\xba\xab\xd8\xed\xb3\rH\xeaud\v\x8b)\x89,\xb1\xf5_\xb7\xbe\x10\x00\x00\u07d4[]Qp)2\x15b\x11\x1bC\bm\v\x045\x91\x10\x9ap\x89\x8c\xf2?\x90\x9c\x0f\xa0\x00\x00\xe0\x94[]\x8c\x8e\xedl\x85\xac!Va\xde\x02fv\x82?\xaa\n\f\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4[mU\xf6q)g@\\e\x91)\xf4\xb1\xde\t\xac\xf2\xcb{\x89\x0e~\xeb\xa3A\vt\x00\x00\u07d4[p\u011c\u024b=\xf3\xfb\xe2\xb1Y\u007f\\\x1bcG\xa3\x88\xb7\x894\x95tD\xb8@\xe8\x00\x00\u07d4[sn\xb1\x83Sb\x9b\u0796v\xda\xdd\x16P4\xce^\xcch\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4[u\x9f\xa1\x10\xa3\x1c\x88F\x9fT\xd4K\xa3\x03\xd5}\xd3\xe1\x0f\x89[F\xdd/\x0e\xa3\xb8\x00\x00\u07d4[w\x84\xca\xea\x01y\x9c\xa3\x02'\x82vg\xce
|\\\xbcv\x89lk\x93[\x8b\xbd@\x00\x00\u07d4[x\xec\xa2\u007f\xbd\xeao&\xbe\xfb\xa8\x97+)^x\x146K\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94[\x80\v\xfd\x1b>\u0525}\x87Z\xed&\xd4/\x1aw\b\xd7*\x8a\x01Z\x82\xd1\u057b\x88\xe0\x00\x00\u07d4[\x85\xe6\x0e*\xf0TO/\x01\xc6N
2\x90\x0e\xbd8\xa3\u01c9lk\x93[\x8b\xbd@\x00\x00\u07d4[\xa2\xc6\xc3]\xfa\xec)h&Y\x19\x04\xd5DFJ\xea\xbd^\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94[\xafmt\x96
\x80>\x83H\xaf7\x10\xe5\xc4\xfb\xf2\x0f\u0214\x8a\x01\x0f@\x02a]\xfe\x90\x00\x00\u07d4[\xc1\xf9U\a\xb1\x01\x86B\xe4\\\xd9\xc0\xe2'3\xb9\xb1\xa3&\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94[\xd25GG\u007fm\t\u05f2\xa0\x05\xc5\xeee\fQ\fV\u05ca\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4[\xd2J\xac6\x12\xb2\f`\x9e\xb4gy\xbf\x95i\x84\a\xc5|\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4[\u0586-Q}M\xe4U\x9dN\xec\n\x06\xca\xd0^/\x94n\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4[\xe0EQ*\x02n?\x1c\xeb\xfdZ~\xc0\xcf\xc3o-

\xc1k\x89\x06\x81U\xa46v\xe0\x00\x00\xe0\x94[\xf9\xf2\"nZ\xea\xcf\x1d\x80\xae\nY\xc6\xe3\x808\xbc\x8d\xb5\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4[\xfa\xfe\x97\xb1\xdd\x1dq+\xe8mA\xdfy\x89SE\x87Z\x87\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\\\x0f.Q7\x8fk\r{\xabas1X\vn9\xad<\xa5\x8a\x02\bj\xc3Q\x05&\x00\x00\x00\u07d4\\)\xf9\xe9\xa5#\xc1\xf8f\x94H\xb5\\H\xcb\xd4|%\xe6\x10\x894F\xa0\xda\xd0L\xb0\x00\x00\xe0\x94\\0\x8b\xacHW\xd3;\xae\xa0t\xf3\x95m6!\xd9\xfa(\xe1\x8a\x01\x0f\b\xed\xa8\xe5U\t\x80\x00\u07d4\\1*V\u01c4\xb1\"\t\x9bvM\x05\x9c!\xec\xe9^\x84\u0289\x05&c\u032b\x1e\x1c\x00\x00\u07d4\\1\x99m\xca\xc0\x15\xf9\xbe\x98[a\x1fF\x870\xef$M\x90\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\\24W\xe1\x87v\x1a\x82v\xe3Y\xb7\xb7\xaf?;n=\xf6\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\\<\x1cd[\x91uC\x11;>l\x1c\x05M\xa1\xfet+\x9a\x89+^:\xf1k\x18\x80\x00\x00\u0794\\=\x19D\x1d\x19l\xb4Cf
\xfc\xad\u007f\xbby\xb2\x9ex\x88\xc6s\xce<@\x16\x00\x00\u07d4\\?V\u007f\xaf\xf7\xba\u0475\x12\x00\"\xe8\xcb\u02a8+l\x17\xb3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\\Ch\x91\x8a\xced\t\u01de\u0280\u036a\xe49\x1d+bN\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\\FA\x97y\x1c\x8a=\xa3\xc9%Co'z\xb1;\xf2\xfa\xa2\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\\H\x81\x16\\\xb4+\xb8.\x979l\x8e\xf4J\xdb\xf1s\xfb\x99\x89\x05\xfe\xe2\"\x04\x1e4\x00\x00\xe0\x94\\H\x92\x90z\a
\xdfo\xd3A>c\xffv}k9\x80#\x8a\x02\xcb\x00\x9f\u04f5y\x0f\x80\x00\u07d4\\O$\xe9\x94\ud3c5\x0e\xa7\x81\x8fG\x1c\x8f\xac;\xcf\x04R\x89]\x80h\x8d\x9e1\xc0\x00\x00\u07d4\\T\x19V\\:\xadNqN\a92\x8e5!\u024f\x05\u0309\x1c\x9fx\u0489>@\x00\x00\u07d4\\a6\xe2\x18\xde\na\xa17\xb2\xb3\x96
-
*a\x12\xb8\t\u05c9\x0f\xf3\u06f6_\xf4\x86\x80\x00\xe0\x94\\a\xaby\xb4\b\xdd2)\xf6bY7\x05\xd7/\x1e\x14{\xb8\x8a\x04\xd0$=4\x98\u0344\x00\x00\u07d4\\m\x04\x1d\xa7\xafD\x87\xb9\xdcH\xe8\xe1\xf6\af\u0425m\xbc\x89O\a\n\x00>\x9ct\x00\x00\u07d4\\o6\xaf\x90\xab\x1aeln\xc8\xc7\xd5!Q'b\xbb\xa3\xe1\x89lh\xcc\u041b\x02,\x00\x00\u07d4\\{\x9e\u01e2C\x8d\x1e<v\x98\xb5E\xb9\xc3\xfdw\xb7\xcdU\x8965\u026d\xc5\u07a0\x00\x00\u07d4\\\x93o;\x9d\"\xc4\x03\xdb^s\x0f\xf1w\xd7N\xefB\u06ff\x89\x04\x10\u0546\xa2\nL\x00\x00\u07d4\\\xb71\x16\r.\x89eg\v\u0792]\x9d\xe5Q\t54}\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\\\xb9S\xa0\xe4/P0\x81\"&!\u007f\xff\xc3\xce#\x04W\xe4\x89\x05k\xc7^-
c\x10\x00\x00\u0794\\\xbd\x8d\xaf'\xdd\xf7\x04\xcd\xd0\xd9\t\xa7\x89\xba6\xedO7\xb2\x88\xb9\xf6]\x00\xf6<\x00\x00\xe0\x94\\\xc4\u02e6!\xf2
cwB\x05\u007f`U\xb8\r\xff\xd7~\x13\x8a\bxG{}%;f\x00\x00\xe0\x94\\\xc7\xd3\x06mE\xd2v!\xf7\x8b\xb4\xb39G>D*\x86\x0f\x8a\x02\x1e\x18\x99\xf07z\xea\x00\x00\u07d4\\\xcc\xf1P\x8b\xfd5\xc2\x050\xaad%\x00\xc1\r\xeee\xea\xed\x89.\x14\x1e\xa0\x81\xca\b\x00\x00\u07d4\\\xcer\xd0h\xc7\xc3\xf5[\x1d(\x19T^w1|\xae\x82@\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\\\xd0\xe4u\xb5D!\xbd\xfc\f\x12\xea\x8e\b+\u05e5\xaf\nj\x89\x032\xca\x1bg\x94\f\x00\x00\u07d4\\\u0548\xa1N\xc6H\xcc\xf6G)\xf9\x16z\xa7\xbf\x8b\xe6\xeb=\x8965\u026d\xc5\u07a0\x00\x00\u07d4\\\u062f`\xdee\xf2M\xc3\xceW0\xba\x92e0\"\xdcYc\x89a\t=|,m8\x00\x00\u07d4\\\xdcG\b\xf1O@\xdc\xc1Zy_}\xc8\xcb\v\u007f\xaa\x9en\x89\x1d\x1c_>\xda
\xc4\x00\x00\u07d4\\\u0d86,\u0391b\xe8~\bI\xe3\x87\xcb]\xf4\xf9\x11\x8c\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\xe0\x94\\\xe2\xe7\u03aa\xa1\x8a\xf0\xf8\xaa\xfa\u007f\xba\xd7L\u021e<\xd46\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\\\xe4@h\xb8\xf4\xa3\xfey\x9ej\x83\x11\xdb\xfd\xed\xa2\x9d\xee\x0e\x89lk\x93[\x8b\xbd@\x00\x00\u0794\\\xeb\xe3\v*\x95\xf4\xae\xfd\xa6ee\x1d\xc0\xcf~\xf5u\x

81\x99\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\\\xf1\x8f\xa7\u0227\xc0\xa2\xb3\xd5\xef\u0459\x0fd\xdd\xc5i$,\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\\\xf4N\x10T\reqd#\xb1\xbc\xb5B\xd2\x1f\xf8:\x94\u034a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\\\xf8\xc0>\xb3\xe8r\xe5\x0f|\xfd\f/\x8d;?,\xb5\x18:\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\\\xfa\x8dV\x85ue\x8c\xa4\xc1\xa5\x93\xacL]\x0eD\xc6\aE\x89\x0f\xc6o\xae7F\xac\x00\x00\u07d4\\\xfa\x98w\xf7\x19\u01dd\x9eIJ\b\xd1\xe4\x1c\xf1\x03\xfc\x87\u0249\n\u05ce\xbcZ\xc6
\x00\x00\u07d4]\x1d\xc38{G\xb8E\x1eU\x10l\f\xc6}m\xc7+\u007f\v\x89lk\x93[\x8b\xbd@\x00\x00\u07d4]#\x1ap\xc1\xdf\xeb6\n\xbd\x97\xf6\x16\xe2\xd1\r9\xf3\u02b5\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4]$\xbd\xbc\x1cG\xf0\xeb\x83\xd1(\xca\xe4\x8a\xc3<H\x17\xe9\x1f\x8965\u026d\xc5\u07a0\x00\x00\u07d4](\x19\xe8\xd5x!\x92.\xe4Ee\f\xca\xec}@TJ\x8d\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4]/\u007f\v\x04\xbaK\xe1a\u1736\xf1\x12\xcez^}\u007f\xe4\x89\x01\xe8\u007f\x85\x80\x9d\xc0\x00\x00\u07d4]2\xf6\xf8nx\u007f\xf7\x8ec\u05cb\x0e\xf9_\xe6\a\x18R\xb8\x89\x15\xbeat\xe1\x91.\x00\x00\u07d4]9\uf7a6\xbd\xff\xf1]\x11\xfe\x91\xf5a\xa6\xf9\xe3\x1f]\xa5\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94]?;\x1fq0\xb0\xbb!\xa0\xfd29b9\x17\x9a%e\u007f\x8a\r:\xb8\xea^\x8f\xd9\xe8\x00\x00\u07d4]WQ\x81\x9bO=&\xed\f\x1a\xc5qU'5'\x1d\xbe\xfa\x8965\u026d\xc5\u07a0\x00\x00\u07d4]\\,\x10\x99\xbb\xee\xfb&~t\xb5\x88\x80\xb4D\xd9DI\xe0\x89\r\xbf\v\u0441\xe2\xe7\x00\x00\u07d4]\\\xdb\xe2[*\x04K{\x9b\u30fc\xaaX\a\xb0m<k\x89lk\x93[\x8b\xbd@\x00\x00\u07d4]]n\x82\x1cn\uf581\f\x83\u0111F\x85`\xefp\xbf\xb5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4]h2K\xcbwm?\xfd\v\xf9\xfe\xa9\x1d\x9f\x03\u007f\u05ab\x0f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4]j\xe8\xcb\u05b39<\"\xd1bT\x10\r\x028\xe8\b\x14|\x89'\a\xe5mQ\xa3\f\x00\x00\u07d4]l\\r\rf\xa6\xab\u0283\x97\x14.c\xd2h\x18\xea\xabT\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4]l\u03c0g8\t\x10B\xad\x97\xa6\xe0\x95\xfe\x8c6\xaay\u0149\n1\x06+\xee\xedp\x00\x00\u07d4]qy\x9c\x8d\xf3\xbc\xcb~\xe4F\xdfP\xb81+\xc4\xebq\u0149\n\u05ce\xbcZ\xc6 \x00\x00\u07d4]\x82-
\x9b>\xf4\xb5\x02bt\a\xda'/g\x81Jk\xec\u0509\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4]\x83\xb2\x1b\xd2q#`Ckg\xa5\x97\xee3x\xdb>z\xe4\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94]\x87+\x12.\x99N\xf2|q\xd7\u07b4W\xbfeB\x9e\xcal\x8a\x01\xb1\xad\xed\x81\u04d4\x10\x80\x00\xe0\x94]\x8d1\xfa\xa8d\xe2!Y\xcdoQu\xcc\xec\xc5?\xa5Mr\x8a\x05\xb6\x96\xb7\r\xd5g\x10\x00\x00\xe0\x94]\x95\x8a\x9b\u0449\u0098_\x86\u014a\x8ci\xa7\xa7\x88\x06\xe8\u068a\x02(\xf1o\x86\x15x`\x00\x00\u07d4]\xa2\xa9\xa4\xc2\xc0\xa4\xa9$\xcb\xe0\xa5:\xb9\xd0\xc6'\xa1\u03e0\x89'\xbf8\xc6TM\xf5\x00\x00\u07d4]\xa4\u0288\x93\\'\xf5\\1\x10H\x84\x0eX\x9e\x04\xa8\xa0I\x89\x04V9\x18$O@\x00\x00\u07d4]\xa5G\x85\u027d0W\\\x89\u07b5\x9d
A\xd2\n9\xe1{\x89j\xa2\t\xf0\xb9\x1de\x80\x00\xe0\x94]\xb6\x9f\xe9>o\xb6\xfb\xd4P\x96k\x97#\x8b\x11\n\xd8'\x9a\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4]\xb7\xbb\xa1\xf9W?$\x11]\x8c\x8cb\xe9\u0388\x95\x06\x8e\x9f\x89\x02\xb5\xaa\xd7,e
\x00\x00\xe0\x94]\xb8D\x00W\x00i\xa9W<\xab\x04\xb4\u6d955\xe2\x02\xb8\x8a\x02\r\u058a\xaf2\x89\x10\x00\x00\u07d4]\xc3m\xe55\x94P\xa1\xec\t\xcb\fD\xcf+\xb4+:\xe45\x89<\x94m\x89;3\x06\x00\x00\u07d4]\xc6\xf4_\xef&\xb0n3\x021?\x88M\xafH\xe2to\xb9\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94]\u0376\xb8zP<m\x8a<e\xc2\u03da\x9a\xa8\x83G\x9a\x1e\x8a\x01\xf2\xbb\xa5\xd8O\x99\xc0\x00\x00\u07d4]\xd1\x12\xf3h\xc0\xe6\xce\xffw\xa9\xdf\x02\xa5H\x16Q\xa0/\xb7\x89\t4r\xc8\\mT\x00\x00\xe0\x94]\xd5:\xe8\x97Rk\x16}9\xf1tN\xf7\xc3\xda[7\xa2\x93\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4]\xde\xd0I\xa6\xe1\xf3)\xdcK\x97\x1er,\x9c\x1f*\u0783\xf0\x8965\u026d\xc

5\u07a0\x00\x00\u07d4]\u562b\xa3D7\x8c\xabD1U[Oy\x99-
\u0090\u0189Hz\x9a0E9D\x00\x00\u07d4]\xe9\xe7\xd5\u0476g\u0415\xdd4\t\x9c\x85\xb0B\x1a\v\
u0181\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4]\xf3'|\xa8Y6\u01e0\xd2\xc0yV\x05\xad%\t^qY\x
89lk\x93[\x8b\xbd@\x00\x00\u07d4]\xff\x81\x1d\xad\x81\x9e\xce;\xa6\x02\u00c3\xfb]\xc6L\n:H\x8
9\n\x15D\xbe\x87\x9e\xa8\x00\x00\u0794^\x03\x1b\nrDq\xd4v\xf3\xbc\xd2\xeb\a\x838\xbfg\xfb\xef
\x88\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94^\a\x85S,w#\xe4\xc0\xaf\x93W\xd5'Ks\xbd\xdd\xdd\u07
8a\x05KA\xea\x9b\xdba\xdc\x00\x00\u07d4^\x11\xec\xf6\x9dU\x1d\u007fO\x84\xdf\x12\x80F\xb3\
xa12@\xa3(\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94^\x1f\xbdNX\xe21+<x\u05ea\xaa\xfa\x10
\xbf\x9c1\x89\xe3\x8a\bxg\x83&\xea\xc9\x00\x00\x00\xe0\x94^2\xc7!\x91\xb89,U\xf5\x10\xd8\xe3
2n:`P\x1db\x8a\tQ>\xa9\xde\x02C\x80\x00\x00\u07d4^Q\xb8\xa3\xbb\t\xd3\x03\xea|\x86\x05\x15\
x82\xfd`\x0f\xb3\xdc\x1a\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u0794^X\xe2U\xfc\x19\x87\n\x040_\
xf2\xa0F1\xf2\xff)K\xb1\x88\xf4?\xc2\xc0N\xe0\x00\x00\u07d4^ZD\x19t\xa8=t\u0187\xeb\xdcc?\xb
1\xa4\x9e{\x1a\u05c9\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4^eE\x8b\xe9d\xaeD\x9fqw7\x04\x97\x97
f\xf8\x89\x87a\x89\x1c\xa7\xccs[o|\x00\x00\u07d4^g\u07c9i\x10\x1a\u06bd\x91\xac\xcdk\xb1\x99\
x12t\xaf\x8d\xf2\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4^n\x97G\xe1b\xf8\xb4\\en\x0fl\xaez\
x84\xba\xc8\x0eN\x89lk\x93[\x8b\xbd@\x00\x00\u07d4^s\x1bU\xce\xd4R\xbb??\xe8q\xdd\xc3\xed
~\xe6Q\n\x8f\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4^t\xed\x80\xe9eW\x88\xe1\xbb&\x97R1\x96g
\xfeuNZ\x89\x03\t'\xf7L\x9d\xe0\x00\x00\u07d4^w.'\xf2\x88\x00\xc5\r\u0697;\xb3>\x10v.n\xea
\x89a\t=|,m8\x00\x00\u07d4^{\x8cT\xdcW\xb0@
bq\x9d\xee~\xf5\xe3~\xa3]b\x89\x9b\xf9\x81\x0f\xd0\\\x84\x00\x00\u07d4^\u007fp7\x87uX\x9f\xc6j
\x81\xd3\xf6S\xe9T\xf5U`\ub243\xf2\x89\x18\x1d\x84\xc8\x00\x00\xe0\x94^\x80n\x84W0\xf8\a>l\x
c9\x01\x8e\xe9\x0f\\\x05\xf9\t\xa3\x8a\x02\x01\xe9m\xac\u03af
\x00\x00\u07d4^\x8eM\xf1\x8c\xf0\xafw\tx\xa8\u07cd\xac\x90\x93\x15\x10\xa6y\x89lk\x93[\x8b\xb
d@\x00\x00\u07d4^\x90\xc8Xw\x19\x87V\xb06l\x0e\x17\xb2\x8eR\xb4FPZ\x89\x14JJ\x18\xef\xeb
h\x00\x00\u07d4^\x95\xfe_\xfc\xf9\x98\xf9\xf9\xac\x0e\x9a\x81\u06b8>\xadw\x00=\x89\x1dB\xc2\r
2y\u007f\x00\x00\u07d4^\xad)\x03z\x12\x89dx\xb1)j\xb7\x14\xe9\u02d5B\x8c\x81\x89\x03\xe0C\a
-
@n\x00\x00\u07d4^\xb3q\xc4\a@lB{;}\xe2q\xad<\x1e\x04&\x95y\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00
\u07d4^\u037a\xea\xb9\x10o\xfe]{Q\x96\x96`\x9a\x05\xba\ub16d\x89\x01\x15\x8eF\t\x13\xd0\x00\
x00\u07d4^\xd0\xd63\x85Y\xefD\xdcza\xed\xeb\x89?\xa5\xd8?\xa1\xb5\x89\v\xed\x1d\x02c\xd9\xf
0\x00\x00\xe0\x94^\u04fb\xc0R@\xe0\u04d9\xebm\xdf\xe6\x0fb\xdeM\x95\t\xaf\x8a)\x14\xc0$u\xf
9\xd6\xd3\x00\x00\u0594^\xd3\xf1\xeb\xe2\xaegV\xb5\xd8\xdc\x19\xca\xd0,A\x9a\xa5w\x8b\x80\u
07d4^\xd5a\x15\xbde\x05\xa8\x82s\xdf\\V\x83\x94p\xd2J-
\xb7\x89\x03\x8ee\x91\xeeVf\x80\x00\xe0\x94^\xf8\xc9a\x86\xb3y\x84\xcb\xfe\x04\u0158@n;\n\xc
3\x17\x1f\x8a\x01\xfd\x934\x94\xaa_\xe0\x00\x00\u07d4^\xfb\xdf\xe58\x99\x99c<&`Z[\xfc,\x1b\xb5
\x95\x93\x93\x89\x03\xc0W\xc9\\\xd9\b\x00\x00\xe0\x94_\x13\x15F1Fm\xcb\x13S\u0210\x93*|\x9
7\xe0\x87\x8e\x90\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4_\x16z\xa2B\xbcL\x18\x9a\xde\xcb
:\u0127\xc4R\xcf\x19/\u03c9lkLM\xa6\u077e\x00\x00\xe0\x94_\x1c\x8a\x04\xc9\rs[\x8a\x15)\t\xae\
xaeco\xb0\xce\x16e\x8a\x01{x'a\x8cZ7\x00\x00\u07d4_#\xba\x1f7\xa9lE\xbcI\x02YS\x8aT\u008b\
xa3\xb0\u0549A\rXj
\xa4\xc0\x00\x00\u07d4_&\xcf4Y\x9b\xc3n\xa6{\x9ez\x9f\x9bC0\xc9\xd5B\xa3\x8965\u026d\xc5\u
07a0\x00\x00\u07d4_)\xc9\xdev]\xde%\x85*\xf0}3\xf2\xceF\x8f\xd2\t\x82\x89lk\x93[\x8b\xbd@\x00

\x00\u07d4_/\a\xd2\u0597\xe8\xc5g\xfc\xfd\xfe\x02\x0fI\xf3`\xbe!9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4_2\x1b=\xaa\xa2\x96\xca\xdf)C\x9f\x9d\xab\x06*K\xff\xed\u0589\x04p%\x90>\xa7\xae\x00\x00\u07d4_3:;#\x10vZ\r\x182\xb9\xbeL\n\x03pL\x1c\t\x8965\u026d\xc5\u07a0\x00\x00\u07d4_4K\x01\xc7\x19\x1a2\xd0v*\xc1\x88\xf0\xec-
\xd4`\x91\x1d\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94_6>\n\xb7G\xe0-
\x1b;f\xab\xb6\x9e\xa5<{\xafR:\x8a\x02w\x01s8\xa3\n\xe0\x00\x00\u07d4_7[\x86`\f@\u0328\xb2gkz\x1a\x1d\x16D\xc5\xf5,\x89\x04F\x18\xd7Lb?\x00\x00\u07d4_>\x1eg9\xb0\xc6\"\x00\xe0\n\x006\x91\xd9\xef\xb28\u061f\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4_H?\xfb\x8fh\n\xed\xf2\xa3\x8fx3\xaf\xdc\xdeY\xb6\x1eK\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94_J\xceL\x1c\xc13\x91\xe0\x1f\x00\xb1\x98\xe1\xf2\v_\x91\xcb\xf5\x8a\x01\x0f\x0f\xa8\xb9\u04c1\x1a\x00\x00\xe0\x94_R\x12\x82\xe9\xb2x\u070c\x03Lr\xafS\xee)\xe5D=x\x8a\x01as-
/\x8f:\xe0\x00\x00\u07d4_h\xa2L~\xb4\x11vgs{39?\xb3\xc2\x14\x8aS\xb6\x89\x02\xce\u0791\x8dE<\x00\x00\u07d4_p\x8e\xaf9\xd8#\x94lQ\xb3\xa3\u9df3\xc0\x03\xe2cA\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4_t.H~:\xb8\x1a\xf2\xf9J\xfd\xbe\x1b\x9b\x8f\\\u0301\xbc\x89u\xc4E\xd4\x11c\xe6\x00\x00\u07d4_t\xed\x0e$\xff\x80\u0672\u0124K\xaa\x99uB\x8c\u05b95\x89\xa1\x8b\xce\xc3H\x88\x10\x00\x00\u07d4_v\xf0\xa3\x06&\x9cx0k=e\r\xc3\xe9\xc3p\x84\xdba\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4_w\xa1\a\xab\x12&\xb3\xf9_\x10\ue0ee\xfcl]\xff>\u0709\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4_{;\xba\xc1m\xab\x83\x1aJ\x0f\xc5;\fT\x9d\xc3l1\u0289i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94_\x93\xff\x83't\xdbQ\x14\xc5[\xb4\xbfD\xcc\U000f53d0?\x8a(\xa9\xc9\x1a&4X)\x00\x00\u07d4_\x96\x16\xc4{Jg\xf4\x06\xb9Z\x14\xfeo\xc2h9o\x17!\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4_\x98\x109\xfc\xf5\x02%\xe2\xad\xf7bu!\x12\xd1\xcc&\xb6\xe3\x89\x1b\x1aAj!S\xa5\x00\x00\u07d4_\x99\u070eI\xe6\x1dW\xda\xef`j\xcd\xd9\x1bMp\a2j\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4_\xa6\x1f\x15-
\xe6\x125\x16\xc7Q$)y(_yj\u01d1\x89\v\x0f\x11\x97)c\xb0\x00\x00\u07d4_\xa7\xbf\xe0C\x88a'\xd4\x01\x1d\x83V\xa4~\x94yc\xac\xa8\x89b\xa9\x92\xe5:\n\xf0\x00\x00\xe0\x94_\xa8\xa5Nh\x17lO\xe2\xc0\x1c\xf6q\xc5\x15\xbf\xbd\xd5(\xa8\x8aE\xe1U\xfa\x01\x10\xfa@\x00\x00\u07d4_\xad\x96\x0fk,\x84V\x9c\x9fMG\xbf\x19\x85\xfc\xb2\xc6]\xa6\x8965f3\xeb\xd8\xea\x00\x00\u07d4_\xc6\xc1\x14&\xb4\xa1\xea\xe7\xe5\x1d\xd5\x12\xad\x10\x90\xc6\xf1\xa8[\x89\x93\xfe\\W\xd7\x10h\x00\x00\u07d4_\u0344Th\x96\xdd\b\x1d\xb1\xa3
\xbdM\x8c\x1d\xd1R\x8cL\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4_\u0368G\xaa\xf8\xd7\xfa\x8b\xca\b\x02\x9c\xa2\x84\x91f\xaa\x15\xa3\x89!\u02b8\x12Y\xa3\xbf\x00\x00\u07d4_\xd1\xc3\xe3\x17x'l\xb4.\xa7@\xf5\xea\xe9\xc6A\xdb\xc7\x01\x89\n\x84Jt$\xd9\xc8\x00\x00\u07d4_\xd3\xd6w~\xc2b\n\xe8:\x05R\x8e\xd4%\a-
<\xa8\xfd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4_\xd9s\xaf6j\xa5\x15|Te\x9b\u03f2|\xbf\xa5\xac\x15\u0589\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4_\xe7w\x03\x80\x8f\x82>l9\x93R\x10\x8b\xdb,R|\xb8|\x89j@v\xcfy\x95\xa0\x00\x00\xe0\x94_\xecI\xc6e\xe6N\xe8\x9d\xd4A\xeet\x05n\x1f\x01\xe9(p\x8a\x01V\x9b\x9es4t\xc0\x00\x00\u07d4_\xf3&\xcd`\xfd\x13k$^)\xe9\bzj\u04e6R\u007f\r\x89e\xea=\xb7UF`\x00\x00\u07d4_\xf9=\xe6\xee\x05L\xadE\x9b-
^\xb0\xf6\x87\x03\x89\xdf\xcbt\x89\v\xed\x1d\x02c\xd9\xf0\x00\x00\u07d4`\x06\xe3m\x92\x9b\xf4]\x8f\x16#\x1b\x12j\x01\x1a\xe2\x83\xd9%\x89\t\x8a}\x9b\x83\x14\xc0\x00\x00\u07d4`!\xe8Z\x88\x14\xfc\xe1\xe8*A\xab\xd1\u04f2\xda\xd2\xfa\xef\xe0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4`8t\n\xe2\x8df\xba\x93\xb0\xbe\bH+2\x05\xa0\xf7\xa0{\x89\x11!a\x85\u009fp\x00\x00\u07d4`?/\xabz\xfbn\x01

{\x94v`i\xa4\xb4;8\x96I#\x89Y\xd2\xdb$\x14\u0699\x00\x00\u07d4`B'm\xf2\x98?\xe2\xbcGY\xdc\x19C\xe1\x8f\xdb\xc3Ow\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4`B\xc6D\xba\xe2\xb9o%\xf9M1\xf6x\xc9\r\xc9f\x90\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4`L\xdf\x18b\x8d\xbf\xa82\x91\x94\xd4x\xddR\x01\xee\xccK\xe7\x89\x01?0j$\t\xfc\x00\x00\u07d4`N\x94w\xeb\xf4r|t[\u02bb\xed\xcbl\xcf)\x99@\"\x8966\x9e\xd7t}&\x00\x00\u07d4`gm\x1f\xa2\x1f\xca\x05\"\x97\xe2K\xf9c\x89\u0171*p\u05c9\r\x17|Zzh\xd6\x00\x00\u07d4`gn\x92\u044b\x00\x05\t\xc6\x1d\xe5@\xe6\xc5\u0776v\xd5\t\x89A\rXj\xa4\xc0\x00\x00\u07d4`o\x17q!\xf7\x85\\!\xa5\x06#0\xc8v\"d\xa9{1\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4`\x86B6\x93\r\x04\xd8@+]\xcb\xeb\x80\u007f<\xafa\x1e\xa2\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4`\xabq\xcd&\xeamnY\xa7\xa0\xf6'\xee\a\x9c\x88^\xbb\xf6\x89\x01s\x17\x90SM\xf2\x00\x00\u07d4`\xaf\x0e\xe1\x18D<\x9b7\xd2\xfe\xadw\xf5\xe5!\u07be\x15s\x89g\x8a\x93b\xe4\x18\x00\x00\u07d4`\xb3X\xcb=\xbe\xfa7\xf4}\xf2\xd76X@\u068e;\u024c\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4`\xb8\u05b7;ySO\xb0\x8b\xb8\xcb\xce\xfa\xc7\xf3\x93\xc5{\xfe\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4`\xbeo\x95?*M%\xb6%o\xfd$#\xac\x148%.N\x89\b!\xab\rD\x14\x98\x00\x00\u0794`\xc3qO\xdd\xdbcFY\u48b1\xeaB\xc4r\x8c\u01f8\xba\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4`\xcc=D^\xbd\xf7j}z\xe5q\u0197\x1d\xffh\u0305\x85\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94`\xd5fq@\xd1&\x14\xb2\x1c\x8e^\x8a3\b.2\xdf\xcf#\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4`\xde\"\xa1Pt2\xa4{\x01\xcch\xc5*\v\xf8\xa2\xe0\u0418\x89\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\u07d4`\xe0\xbd\u0422Y\xbb\x9c\xb0\x9d?7\xe5\u034b\x9d\xac\uafca\x89JD\x91\xbdm\xcd(\x00\x00\u07d4`\xe3\xccC\xbc\xdb\x02j\xadu\x9cpf\xf5U\xbb\xf2\xacf\xf5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4a\x04+\x80\xfd`\x95\u0478{\xe2\xf0\x0f\x10\x9f\xab\xaf\xd1W\xa6\x89\x05k\xc7^-c\x10\x00\x00\u07d4a\a\xd7\x1d\xd6\xd0\xee\xfb\x11\xd4\xc9\x16@L\xb9\x8cu>\x11}\x89lk\x93[\x8b\xbd@\x00\x00\u07d4a\x0f\xd6\xeeN\xeb\xab\x10\xa8\xc5]\vK\xd2\xe7\xd6\xef\x81qV\x89\x01\x15\x95a\x06]]\x00\x00\u07d4a\x14\xb0\xea\xe5Wi\x03\xf8\v\xfb\x98\x84-$\xed\x92#\u007f\x1e\x89\x05k\xc7^-c\x10\x00\x00\u07d4a!\xaf9\x8a[-\xa6\x9fe\xc68\x1a\xec\x88\u039c\xc6D\x1f\x89\"\xb1\xc8\xc1\"z\x00\x00\x00\u07d4a&g\xf1r\x13[\x95\v,\xd1\xde\x10\xaf\xde\xcehW\xb8s\x8965\u026d\xc5\u07a0\x00\x00\u07d4a,\xed\x8d\xc0\u071e\x89\x9e\xe4oyb33\x15\xf3\xf5^D\x89\x12^5\xf9\xcd=\x9b\x00\x00\u07d4a4\xd9B\xf07\xf2\xcc=BJ#\f`=g\xab\xd3\xed\xf7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4a:\xc5;\xe5e\xd4e6\xb8q[\x9b\x8d:\xe6\x8aK\x95\x89\xcb\xd4{n\xaa\x8c\xc0\x00\x00\u07d4a?\xabD\xb1k\xbeUMD\xaf\xd1x\xab\x1d\x02\xf3z\ua949lk\x93[\x8b\xbd@\x00\x00\u07d4aN\x8b\xef=\xd2\u015bY\xa4\x14Vt@\x10\x185\x18\x84\xea\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4aQ\x84d\xfd\u0637<\x1b\xb6\xacm\xb6\x00eI8\xdb\xf1z\x89\n\u05ce\xbcZ\xc6\x00\x00\u07d4aT}7nSi\xbc\xf9x\xfc\x16,<V\xaeE5G\xe8\x89\n\u05ce\xbcZ\xc6\x00\x00\u07d4aX\xe1\a\xc5\xebT\xcbv\x04\xe0\u034d\xc1\xe0u\x00\xd9\x1c<\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94aZo6w\u007f@\xd6a~\xb5\x81\x98\x96\x18i\x83\xfd71\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d4a_\x826\\Q\x01\xf0q\xe7\xd2\xcbj\xf1Oz\xad,\x16\u0189\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4ap\xdd\x06\x87\xbdU\u0288\xb8z\xde\xf5\x1c\xfd\xc5\\M\xd4X\x89l\xb3/\\4\xfeD\x00\x00\u07d4as9G\xfa\xb8\xdb\xd3Q\xef\xd6xU\xea\x0e\x88\x13s\xa0\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4ay\x97\x99\a\xfe\u007f\x03~L8\x02\x9d`\xbc\xba\xb82\xb3\u0589WG=\x05\u06ba\xe8\x00\x00\xe0\x94a\u007f\x89O\xa7\x0e\x94\xa8jI\xcdt\xe028\xf6M<\u064a\x01\x0f\r\xba\xe6\x10\tR\x80\x00\u07d4a\u007f\

xf2\u0300>1\xc9\b\"3\xb8%\xd0%\xbe?{\x10V\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4a\x91\xdd\u0276J\x8e\b\x90\xb427\t\u05e0|H\xb9*d\x89*\x03I\x19\u07ff\xbc\x00\x00\u07d4a\x96\xc3\xd3\xc0\x90\x8d%Cf\xb7\xbc\xa5WE\"-
\x9dM\xb1\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4a\x9f\x17\x14E\xd4+\x02\xe2\xe0p\x04\xad\x8a\xfeiO\xa5=j\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4a\xad\xf5\x92\x9a^)\x81hN\xa2C\xba\xa0\x1f}\x1f^\x14\x8a\x89\x05\xfa\xbfl\x98O#\x00\x00\u07d4a\xb1\xb8\xc0\x12\xcdLx\xf6\x98\xe4p\xf9\x02V\xe6\xa3\x0fH\u0749\n\u05ce\xbcZ\xc6
\x00\x00\u07d4a\xb3\xdf.\x9e\x9f\xd9h\x13\x1f\x1e\x88\xf0\xa0\xeb[\xd7eFM\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4a\xb9\x02\u0166s\x88X&\x82\r\x1f\xe1EI\xe4\x86_\xbd\u0089\x12$\xef\xed*\u1440\x00\u07d4a\xb9\x05\xdef?\xc1s\x86R;:(\xe2\xf7\xd07\xa6U\u0349\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4a\xba\x87\xc7~\x9bYm\xe7\xba\x0e2o\xdd\xfe\xec!c\xeff\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4a\xbf\x84\u056b\x02oX\xc8s\xf8o\xf0\xdf\u0282\xb5W3\xae\x89lk\x93[\x8b\xbd@\x00\x00\u07d4a\xc4\xee|\x86LMk^7\xea\x131\xc2\x03s\x9e\x82k/\x89\x01\xa15;8*\x91\x80\x00\u07d4a\xc80\xf1eG\x18\xf0u\u032b\xa3\x16\xfa\xac\xb8[}\x12\v\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4a\xc8\xf1\xfaC\xbf\x84i\x99\xec\xf4{+2M\xfbkc\xfe:\x89+^:\xf1k\x18\x80\x00\x00\u07d4a\xc9\xdc\u8c98\x1c\xb4\x0e\x98\xb0@+\xc3\xeb(4\x8f\x03\xac\x89\n\xac\xac\u0679\xe2+\x00\x00\u07d4a\u03a7\x1f\xa4d\xd6*\a\x06?\x92\v\f\xc9\x17S\x973\u0609Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4a\xd1\x01\xa03\xee\x0e.\xbb1\x00\xed\xe7f\xdf\x1a\xd0$IT\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4a\xedU\x96\u0197
\u007f=U\xb2\xa5\x1a\xa7\xd5\x0f\a\xfa\t\xe8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4a\xff\x8eg\xb3M\x9e\xe6\xf7\x8e\xb3o\xfe\xa1\xb9\xf7\xc1W\x87\xaf\x89X\xe7\x92n\xe8X\xa0\x00\x00\u07d4b\x05\xc2\xd5dtp\x84\x8a8@\xf3\x88~\x9b\x01]4u\\\x89a\x94\x04\x9f0\xf7
\x00\x00\u07d4b(\xad\xe9^\x8b\xb1}\x1a\xe2;\xfb\x05\x18AMI~\x0e\xb8\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94b)\xdc\xc2\x03\xb1\xed\xcc\xfd\xf0n\x87\x91\fE*\x1fMzr\x8a\x06\xe1\xd4\x1a\x8f\x9e\xc3P\x00\x00\u0794b+\xe4\xb4T\x95\xfc\xd91C\xef\xc4\x12\u0599\xd6\xcd\xc2=\u0148\xf0\x15\xf2W6B\x00\x00\u07d4b3\x1d\xf2\xa3\xcb\xee5
\xe9\x11\u07a9\xf7>\x90_\x89%\x05\x89lk\x93[\x8b\xbd@\x00\x00\u07d4bVD\xc9Z\x87>\xf8\xc0l\u06de\x9fm\x8dv\x80\x04=b\x89a\x94\x04\x9f0\xf7 \x00\x00\u07d4be\xb2\xe7s\x0f6\xb7v\xb5-
\f\x9d\x02\xad\xa5]\x8e<\xb6\x8965\u026d\xc5\u07a0\x00\x00\u07d4bh\n\x15\xf8\u0338\xbd\xc0/s`\xc2Z\xd8\u03f5{\x8c\u034965\u026d\xc5\u07a0\x00\x00\u07d4b\x94\xea\xe6\xe4
\xa3\xd5`\n9\xc4\x14\x1f\x83\x8f\xf8\xe7\xccH\x89\xa00\xdc\xeb\xbd/L\x00\x00\u07d4b\x97\x1b\xf2cL\xee\v\xe3\u0249\x0fQ\xa5`\x99\u06f9Q\x9b\x89#\x8f\xd4,\\\xf0@\x00\x00\u07d4b\x9b\xe7\xab\x12jS\x98\xed\xd6\u069f\x18D~x\u0192\xa4\xfd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4b\xb4\xa9\"nah<r\xc1\x83%F\x90\xda\xf5\x11\xb4\x11z\x89\x0e\x189\x8ev\x01\x90\x00\x00\u07d4b\xb9\b\x1ew\x104^8\xe0.\x16D\x9a\xce\x1b\x85\xbc\xfcN\x891T\xc9r\x9d\x05x\x00\x00\u07d4b\xc3|R\xb9\u007fK\x04\v\x1a\xa3\x91\xd6\xde\xc1R\x89<G\a\x8965\u026d\xc5\u07a0\x00\x00\u07d4b\u0272q\xff\u0577p\xa5\xee\xe4\xed\xc9x{\\\xdcp\x97\x14\x89lk\x93[\x8b\xbd@\x00\x00\u07d4b\xd5\xccq\x17\xe1\x85\x00\xac/\x9e<&\xc8k\n\x94\xb0\xde\x15\x89\x05\xb1*\ufbe8\x04\x00\x00\u07d4b\xdcrr\x90$7_\xc3|\xbb\x9c|#\x93\xd1\x0233\x0f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4b\xe6\xb2\xf5\xeb\x94\xfazC\x83\x1f\xc8~%J?\u3fcf\x89\x89\r\x8drkqw\xa8\x00\x00\u07d4b\xf2\xe5\xcc\xec\xd5,\u0139^\x05\x97\xdf'\xcc\a\x97\x15`\x8c\x89\a\xc0\x86\x0eZ\x80\xdc\x00\x00\xe0\x94b\xfb\x8b\xd1\xf0\xe6k\x90S>\a\x1el\xbea\x11\xfe\xf0\xbcc\x8a\x03\xba\x19\x10\xbf4\x1b\x00\x00\x00\xe0\x94c\n\

x91:\x901\xc9I*\xbdLA\u06f1PT\xcf\xecD\x16\x8a\x014X\xdbg\xaf5\xe0\x00\x00\xe0\x94c\fRs\x12
mQ|\xe6q\x01\x81\x1c\xab\x16\xb8SL\xf9\xa8\x8a\x01\xfe\xcc\xc6%s\xbb\u04c0\x00\u07d4c\x100
\xa5\xb2{\a(\x8aEio\x18\x9e\x11\x14\xf1*\x81\xc0\x89\x1b\x1azB\v\xa0\r\x00\x00\u07d4c\x10\xb0
\xfd\x98\x04IW\x99P\x92\t\x0f\x17\xf0NR\xcd\xfd\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4c+\x9
1I\xd7\x01x\xa7364'^\x82\u0555?'\x96{\x89%\xf2s\x93=\xb5p\x00\x00\u07d4c,\xec\xb1\f\xfc\xf3\x8
e\u0246\xb4;\x87p\xad\xec\xe9
\x02!\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4c1\x02\x8c\xbbZ!H[\xc5\x1bVQB\x99;\xdb%\x82\x
a9\x89\x1c\xfd\xd7F\x82\x16\xe8\x00\x00\u07d4c3O\xcf\x17E\x84\x0eK\tJ;\xb4\v\xb7o\x96\x04\xc
0L\x89\u05e5\xd7\x03\xa7\x17\xe8\x00\x00\u07d4c4\nWqk\xfac\xebl\xd13r\x12\x02W[\xf7\x96\xf0
\x89\va\xe0\xa2\f\x12q\x80\x00\u07d4cN\xfc$7\x11\a\xb4\xcb\xf0?y\xa9=\xfd\x93\xe41\xd5\xfd\x8
9B5\x82\xe0\x8e\xdc\\\x80\x00\xe0\x94c\\\x00\xfd\xf05\xbc\xa1_\xa3a\r\xf38N\x0f\xb7\x90h\xb1\x8
a\x01\xe7\xe4\x17\x1b\xf4\u04e0\x00\x00\u07d4ca.xb\xc2{X|\xfbm\xaf\x99\x12\xcb\x05\x1f\x03\n\
x9f\x89\x02[\x19\u053f\xe8\xed\x00\x00\u07d4cfgU\xbdA\xb5\x98i\x97x<\x13\x040\b$+<\xb5\x89\
x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4c{\xe7\x1b:\xa8\x15\xffE=VB\xf70tE\vd\xc8*\x89lk\x93[\x
8b\xbd@\x00\x00\xe0\x94c}g\xd8\u007fXo\nZG\x9e
\xee\x13\xea1\n\x10\xb6G\x8a\n:Y&\xaf\xa1\xe70\x00\x00\u07d4c\u007fXi\xd6\xe4i_\x0e\xb9\xe2
s\x11\u0107\x8a\xff33\x80\x89j\xc0Nh\xaa\xec\x86\x00\x00\u07d4c\x97|\xad}\r\xcd\xc5+\x9a\xc9\
xf2\xff\xa16\xe8d(\x82\xb8\x89\x04\x10\u0546\xa2\nL\x00\x00\u07d4c\xa6\x1d\xc3\n\x8e;0\xa7c\x
c4!<\x80\x1c\xbf\x98s\x81x\x8965\u026d\xc5\u07a0\x00\x00\u07d4c\xacT\\\x99\x12C\xfa\x18\xae\
xc4\x1dOoY\x8eUP\x15\u0709 \x86\xac5\x10R`\x00\x00\u07d4c\xb9uMu\xd1-
8@9\xeci\x06<\v\xe2\x10\xd5\xe0\u3252\v\x86\f\xc8\xec\xfd\x80\x00\u07d4c\xbbfO\x91\x17\x03v(
YM\xa7\xe3\xc5\b\x9f\xd6\x18\xb5\xb5\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4c\u00a3\xd25\x
e5\xee\xab\xd0\u0526\xaf\u06c9\xd9F'9d\x95\x89CN\xf0[\x9d\x84\x82\x00\x00\u07d4c\xc8\xdf\xd
e\v\x8e\x01\xda\xdc.t\x8c\x82L\xc06\x9d\U00010cc9\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4c\x
d5Z\u065b\x917\xfd\x1b
\xcc+O\x03\xd4,\xba\xdd\xf34\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4c\xd8\x00H\x87u\x96\xe
0\u0084\x89\xe6P\xcdJ\xc1\x80\tjI\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\xe0\x94c\xe4\x14`>\x80\xd4\xe5\xa0\xf5\xc1\x87t
FB%\x82\b\xe4\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x94c\xe8\x8e.S\x9f\xfbE\x03\x86\xb4\xe4g\x89\xb2#\xf5GlE\x8a\x01U\x17\nw\x8e%\
xd0\x00\x00\u07d4c\xef/\xbc=\xaf^\xda\xf4\xa2\x95b\x9c\xcf1\xbc\xdf@8\xe5\x89O%\x91\xf8\x96\
xa6P\x00\x00\u07d4c\xf0\xe5\xa7R\xf7\x9fg\x12N\xedc:\xd3\xfd'\x05\xa3\x97\u0509\u0556{\xe4\xf
c?\x10\x00\x00\xe0\x94c\xf5\xb5=y\xbf.A\x14\x89Re0\"8E\xfa\xc6\xf6\x01\x8a\x06ZM\xa2]0\x16\x
c0\x00\x00\u07d4c\xfc\x93\x00\x13\x05\xad\xfb\u0278])\xd9)\x1a\x05\xf8\xf1A\v\x8965\u026d\xc5\
u07a0\x00\x00\u0794c\xfek\xccK\x8a\x98P\xab\xbeu\x8070\xc92%\x1f\x14[\x88\xfc\x93c\x92\x80\
x1c\x00\x00\u07d4d\x03\xd0bT\x96\x90\xc8\xe8\xb6>\xaeA\xd6\xc1\tGn%\x88\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4d\x04+\xa6\x8b\x12\xd4\xc1Qe\x1c\xa2\x81;sR\xbdV\xf0\x8e\x89
\x86\xac5\x10R`\x00\x00\u0794d\x05\xdd\x13\xe9:\xbc\xff7~p\x0e<\x1a\x00\x86\xec\xa2})\x88\xfc\
x93c\x92\x80\x1c\x00\x00\xe0\x94d\n\xbam\xe9\x84\xd9E\x177x\x03p^\xae\xa7\t_J\x11\x8a\x02\x
1e\x19\xe0\u027a\xb2@\x00\x00\u07d4d\v\xf8t\x15\xe0\xcf@s\x01\xe5Y\x9ah6m\xa0\x9b\xba\u0
209\x1a\xbc\x9fA`\x98\x15\x80\x00\u07d4d
\xf8\xbc\xc8\x16JaR\xa9\x9dk\x99i0\x05\xcc\xf7\xe0S\x8965f3\xeb\xd8\xea\x00\x00\u07d4d$\x1ax

D)\x0e\n\xb8U\xf1\u052au\xb5SE\x03\"$\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4d&J\xed\xd5-
\xca\xe9\x18\xa0\x12\xfb\xcd\f\x03\x0e\xe6\xf7\x18!\x8965\u026d\xc5\u07a0\x00\x00\u07d4d7\x0
e\x87 &E\x12Z5\xb2\a\xaf\x121\xfb`r\xf9\xa7\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4d=\x9a\xee\u0531\x80\x94~\u04b9 |\xceL=\xdcU\xe1\xf7\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4dC\xb8\xaec\x9d\xe9\x1c\xf7<Z\xe7c\xee\xee\xd3\u077b\x92S\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4dE\u007f\xa3;\b2PIO}\x11\x80\xdc\xe4\x8fF\xf3\xe0\xff\x89lk\x93[\x8b\xbd@\x00
\x00\u07d4dFJh\x05\xb4bA*\x90\x1d-\xb8\x17K\x06\xc2-
\ue989\x19\xc8F\xa0)\xc7\xc8\x00\x00\u07d4dK\xa6\xc6\x10\x82\xe9\x89\x10\x9f\\\x11\u0534\x0e
\x99\x16`\xd4\x03\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4db\x8co\xb8\xect:\xdb\xd8|\xe5\xe0\
x18\xd51\xd9!\x047\x89\x01s\x17\x90SM\xf2\x00\x00\u07d4dc\xf7\x15\u0554\xa1\xa4\xac\u4edc;(
\x8at\xde\xcf)M\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4df(\xa5<,A\x93\u06885\x9c\xe7\x18\xda\u07
52\xb7\xa4\x8d\x89\n\xd8\x00l/^\xf0\x00\x00\u07d4dg-
\xa3\xab\x05(!\xa0$=\x1c\xe4\xb6\xe0\xa3e\x17\xb8\xeb\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4dj\xfb\xa7\x1d\x84\x9e\x80\xc0\xedY\xca\xc5\x19\xb2x\xe7\xf7\xab\xe4\x8965\u02
6d\xc5\u07a0\x00\x00\u07d4dn\x04=\x05\x97\xa6d\x94\x8f\xbb\r\xc1Tu\xa3\xa4\xf3\xa6\xed\x89\x
01\x15\x8eF\t\x13\xd0\x00\x00\u07d4dp\xa4\xf9.\u01b0\xfc\xcd\x01#O\xa5\x90#\xe9\xff\x1f:\xac\x
89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4d{\x85\x04M\xf2\xcf\vN\u0508.\x88\x81\x9f\xe2*\xe5\xf7\x9
3\x8966;]\x9awp\x00\x00\u07d4d\x85G\x0ea\xdb\x11\n\xeb\u06ef\xd56v\x9e<Y\x9c\xc9\b\x89
\x86\xac5\x10R`\x00\x00\xe0\x94d\x8f[\u04a2\xae\x89\x02\xdb7\x84}\x1c\xb0\u06d3\x90\xb0bH\x
8a\x01\xa55\xec\xf0v\n\x04\x80\x00\xe0\x94d\x9a+\x98y\u034f\xb76\xe6p;\fwG\x84\x97\x96\xf1\x
0f\x8a\x01\x8e\xe2-
\xa0\x1a\xd3O\x00\x00\u07d4d\x9a\x85\xb96S\a_\xa6V,@\x9aV]\b{\xa3\u1e89lk\x93[\x8b\xbd@\x
00\x00\u07d4d\xad\xcc\xee\xc5=\xd9\xd9\xdd\x15\xc8\xcc\x1a\x9esm\xe4$\x1d,\x89\x03\t'\xf7L\x9
d\xe0\x00\x00\u07d4d\xcf\t5\xbf\x19\xd2\u03bb\xec\xd8x\r'\xd2\xe2\xb2\xc3Af\x89j\xcb=\xf2~\x1f\x
88\x00\x00\u07d4d\xd8\f;\x8b\xa6\x82\x82)\vu\xe6]\x89x\xa1Z\x87x,\x89j\xcb=\xf2~\x1f\x88\x00\x
00\xe0\x94d\u06e2\xd6a[\x8b\xd7W\x186\xdcu\xbcy\xd3\x14\xf5\xec\xee\x8a\x02\x1e\x19\xe0\u0
27a\xb2@\x00\x00\u07d4d\xe0!z[8\xaa@X6%\x96\u007f\xa9\x886\x908\x8bo\x89\n\u05ce\xbcZ\x
c6
\x00\x00\u07d4d\xe0*\xbb\x01l\xc2:)4\xf6\xbc\u0776\x81\x90P!\xd5c\x8965\u026d\xc5\u07a0\x00\
x00\u07d4d\xe0>\xf0p\xa5G\x03\xb7\x18NH'l\\\x00w\xefK4\x89\x11X\xe4`\x91=\x00\x00\x00\xe0\x
94d\xe2\xde!
\v\x18\x99\u00e0\xc0e;P@\x13m\r\xc8B\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4d\xec\x8a[t?4y\x
e7\a\xda\xe9\xee \u076aO@\xf1\u0649\n\u05ce\xbcZ\xc6
\x00\x00\u07d4e\x03\x86\v\x19\x10\b\xc1U\x83\xbf\u0201X\t\x93\x01v((\x8965\u026d\xc5\u07a0\x
00\x00\u07d4e\x051\x911\x9e\x06z%\xe66\x1dG\xf3\u007fc\x18\xf84\x19\x89\x15[\xd90\u007f\x9f
\xe8\x00\x00\u07d4e\t;#\x9b\xbf\xba#\xc7w\\\xa7\xdaZ\x86H\xa9\xf5L\xf7\x89\x15\xaf\x1dx\xb5\x8
c@\x00\x00\u07d4e\t\xee\xb14~\x84/\xfbA>7\x15^,\xbcs\x82s\xfd\x89lk\x93[\x8b\xbd@\x00\x00\x
e0\x94e\vBUU\xe4\xe4\xc5\x17\x18\x14h6\xa2\xc1\xeew\xa5\xb4!\x8a\x04<3\xc1\x93ud\x80\x00\x
00\u07d4e\f\xf6}\xb0`\xcc\xe1uh\xd5\xf2\xa4#h|Idv\t\x89\x05k\xc7^-
c\x10\x00\x00\u07d4e\x10\xdfB\xa5\x99\xbc\xb0\xa5\x19\u0329a\xb4\x88u\x9aogw\x89lk\x93[\x8b
\xbd@\x00\x00\u07d4e6u\xb8B\xd7\u0634a\xf7\"\xb4\x11|\xb8\x1d\xac\x8ec\x9d\x89\x01\xae6\x1f
\xc1E\x1c\x00\x00\u07d4eK~\x80\x87\x99\xa8=r\x87\xc6w\x06\xf2\xab\xf4\x9aId\x04\x89j\xcb=\xf

2~\x1f\x88\x00\x00\xe0\x94eORHG\xb3\xa6\xac\xc0\xd3\xd5\xf1\xf3b\xb6\x03\xed\xf6_\x96\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4eY4\u068etN\xaa=\xe3M\xbb\xc0\x89LN\xda\va\xf2\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4e]\\\xd7H\x96)\xe2A<!\x05\xb5\xa1r\xd93\xc2z\xf8\x89\xdb\x03\x18l\xd8@\xa6\x00\x00\u07d4e`\x18XA0\u06c3\xab\x05\x91\xa8\x12\x8d\x93\x81fj\x8d\x0e\x89\x03w\x9f\x91
\x19\xfc\x00\x00\u07d4e`\x94\x13(\xffX|\xbcV\u00ccx#\x8a{\xb5\xf4B\xf6\x89(a\x90kY\xc4z\x00\x00\u07d4eey\xda\xed\u0493p\u06777\xee?\\\xd9\xd8K\u00b3B\x89M\x85<\x8f\x89\b\x98\x00\x00\u07d4etswOc\xac=by\xfd\aC\xd5y\fO\x16\x15\x03\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4e\x80\xb1\xbc\x949\x0f\x04\xb3\x97\xbds\xe9]\x96\xef\x11\xea\xf3\xa8\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4e\x84\x9b\xe1\xaf
\x10\x0e\xb8\xa3\xbaZ[\xe4\u04ee\x8d\xb5\xa7\x0e\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4e\x9c\nr\xc7g\xa3\xa6\\\xed\x0e\x1c\xa8\x85\xa4\xc5\x1f\u0677y\x89lk\x93[\x8b\xbd@\x00\x00\u07d4e\xa5!A\xf5k\uf619\x17$\xc6\xe7\x053\x81\u068bY%\x89\x03B\x9c3]W\xfe\x00\x00\u07d4e\xa9\xda\xd4.\x162\xba>NIb?\xabb\xa1~M6\x11\x89\x05\fL\xb2\xa1\f`\x00\x00\u07d4e\xaf\x8d\x8b[\x1d\x1e\xed\xfaw\xbc\xbc\x96\xc1\xb13\xf83\x06\u07c9\x05P\x05\xf0\xc6\x14H\x00\x00\u07d4e\xaf\x90\x87\xe0QgqT\x97\u0265\xa7I\x18\x94\x89\x00M\xef\x89-
C\xf3\xeb\xfa\xfb,\x00\x00\u0794e\xb4/\xae\xcc\x1e\u07f1B\x83\u0297\x9a\xf5E\xf6;0\xe6\f\x88\xfc\x93c\x92\x80\x1c\x00\x00\u0794e\xd3>\xb3\x9c\xdadS\xb1\x9ea\xc1\xfeM\xb91p\xef\x9d4\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4e\xd8\xddN%\x1c\xbc\x02\x1f\x05\xb0\x10\xf2\xd5\xdcR\f8r\xe0\x89-
CW\x9a6\xa9\x0e\x00\x00\u07d4e\xea&\xea\xbb\xe2\xf6L\xcc\xcf\xe0h)\xc2]F7R\x02%\x89%\xf2s\x93=\xb5p\x00\x00\u07d4e\xeag\xad?\xb5j\xd5\xfb\x948}\u04ce\xb3\x83\x00\x1d|h\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94e\xeb\xae\xd2~\u06dd\xcc\x19W\xae\xe5\xf4R\xac!\x05\xa6\\\x0e\x8a\t7\u07ed\xae%\u26c0\x00\u07d4e\xee
\xb0m\x9a\u0549\xa7\xe7\xce\x04\xb9\xf5\xf7\x95\xf4\x02\xae\u0389lk\x93[\x8b\xbd@\x00\x00\u07d4e\xf544m/\xfbx\u007f\xa9\xcf\x18]t[\xa4)\x86\xbdn\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94e\xf5\x87\x0f&\xbc\xe0\x89g}\xfc#\xb5\x00\x1e\xe4\x92H4(\x8a\x01\x12\xb1\xf1U\xaa2\xa3\x00\x00\u07d4e\xfd\x02\xd7\x04\xa1*M\xac\xe9G\x1b\x06E\xf9b\xa8\x96q\u0209\x01\x8d\x1c\xe6\xe4'\u0340\x00\u07d4e\xff\x87O\xaf\xceM\xa3\x18\xd6\xc9=W\xe2\u00ca\rs\xe8
\x8968\x02\x1c\xec\u06b0\x00\x00\xe0\x94f\x05W\xbbC\xf4\xbe:\x1b\x8b\x85\xe7\xdf{<[\xcdT\x80W\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4f\b,u\xa8\xde1\xa59\x13\xbb\xd4M\xe3\xa07O\u007f\xaaA\x89O%\x91\xf8\x96\xa6P\x00\x00\u07d4f\x11\xceY\xa9\x8b\a*\xe9Y\xdcI\xadQ\x1d\xaa\xaa\xa1\x9dk\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4f
\x1b\xd2'\xaem\u01bd\xfe\xd5\xfb\u0781\x1f\xec\xfe^\x9d\u0649
>\x9e\x84\x92x\x8c\x00\x00\u07d4f#4\x81G$\x93[y1\xdd\xcaa\x00\xe0\rFw'\u0349\"\x88&\x9d\a\x83\xd4\x00\x00\u07d4f'O\xea\x82\xcd0\xb6\u009b#5\x0eOO=1\nX\x99\x89p7\x05P\xab\x82\x98\x00\x00\u07d4f,\xfa\x03\x8f\xab7\xa0\x17E\xa3d\u1e41'\xc5\x03tm\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4f5\xb4oq\x1d-\xa6\xf0\xe1cp\u034e\xe4>\xfb,-
R\x89lk\x93[\x8b\xbd@\x00\x00\u07d4f6\x04\xb0P0F\xe6$\xcd&\xa8\xb6\xfbGB\xdc\xe0*o\x89\x03\x8b\x9by~\xf6\x8c\x00\x00\u07d4f6\u05ecczH\xf6\x1d8\xb1L\xfdHe\xd3m\x14(\x05\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4f@\xcc\xf0SU\\\x13\n\xe2\xb6Vd~\xa6\xe3\x167\xb9\xab\x89j\xcb

=\xf2~\x1f\x88\x00\x00\u07d4fBK\xd8x[\x8c\xb4a\x10*\x90\x02\x83\xc3]\xfa\a\xefj\x89\x02.-
\xb2ff\xfc\x80\x00\u07d4fL\xd6}\xcc\u026c\x82(\xb4\\U\u06cdvU\ve\x9c\u0709\x15[\xd90\u007f\x9f
\xe8\x00\x00\u07d4fNC\x11\x98p\xaf\x10zD\x8d\xb1'\x8b\x04H8\xff\u036f\x89\x15\xaf\x1dx\xb5\x
8c@\x00\x00\u07d4fQso\xb5\x9b\x91\xfe\xe9\xc9:\xa0\xbdn\xa2\xf7\xb2Pa\x80\x89\x1b\x1a\xe4\x
d6\xe2\xefP\x00\x00\u07d4f[\x00\x0f\vw'P\xcc<!z^\xf4)\xa9+\xf1\u033b\x89\xd8\xd7&\xb7\x17z\x8
0\x00\x00\u07d4ff
\x06\x01\\\x1f\x8e<\xcf\xca\xeb\xc8\xeeh\a\xee\x19c\x03\x89\x1b\x1b:\x1a\xc2a\xec\x00\x00\u07d
4fgF\xfb\x93\u0453\\Z<hNrP\x10\xc4\xfa\u0431\u0609\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4fkO
7\xd5]c\xb7\xd0V\xb6\x15\xbbt\xc9k;\x01\x99\x1a\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4fq\x9
c\x06\x82\xb2\xac\u007f\x9e'\xab\xeb\xec~\u07cd\xec\xf0\xae\r\x89\x01\x15\x8eF\t\x13\xd0\x00\x
00\u07d4fq\xb1\x82\xc9\xf7A\xa0\xcd<5ls\xc21&\xd4\xf9\xe6\xf4\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4fy\xae\xec\xd8zW\xa7?3V\x81\x1d,\xf4\x9d\fM\x96\u0709
\x86\xac5\x10R`\x00\x00\u07d4f{a\xc0;\xb97\xa9\xf5\xd0\xfcZ\t\xf1\xea3c\xc7p5\x89\xe6d\x99\"\x
88\xf2(\x00\x00\u07d4f\x85\xfd.%Dp,6\v\x8b\xb9\xeex\xf10\xda\xd1m\xa5\x89lk\x93[\x8b\xbd@\x0
0\x00\xe0\x94f\x8bk\xa8\xab\b\xea\xce9\xc5\x02\xefg+\xd5\u0336\xa6z
\x8a\x06\x97\xd9]B\x013<\x00\x00\u07d4f\x92]\xe3\xe4?KA\xbf\x9d\xad\xde'\xd5H\x8e\xf5i\xea\r\x
89\x02\"\xc8\xeb?\xf6d\x00\x00\u07d4f\xb0\xc1\x00\u0111I\x93]\x14\xc0\xdc
,\u0390|\xea\x1a=\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4f\xb1\xa6=\xa4\xdc\xd9\xf8\x1f\xe5O^?\xc
b@U\xef~\xc5O\x89\n\xeb'*\u07dc\xfa\x00\x00\u07d4f\xb3\x987\xcb<\xac\x8a\x80*\xfe?\x12\xa2X
\xbb\xcab\xda\u0349\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94f\xc83\x1e\xfeq\x98\xe9\x8b-
2\xb98h\x8e2A\xd0\xe2O\x8a\x02\t\x80Q\x97\x0e9\xd0\x00\x00\u07d4f\u030a\xb2<\x00\u0478*\x
cd}s\xf3\x8c\x99\xe0\xd0ZO\xa6\x89\x05k\xc7^-
c\x10\x00\x00\u07d4f\xdc\xc5\xfbN\xe7\xfe\xe0F\xe1A\x81\x9a\xa9hy\x9ddD\x91\x89Hz\x9a0E9D
\x00\x00\u07d4f\xe0\x94'\xc1\xe6=\xee\xd7\xe1+\x8cU\xa6\xa1\x93
\xefKj\x89\t79SM(h\x00\x00\u07d4f\xec\x16\ue72a\xb4\x11\xc5Zf)\xe3\x18\xden\xe2\x16I\x1d\x89.
\xe4IU\b\x98\xe4\x00\x00\u07d4f\xf5\x04\x06\xeb\x1b\x11\xa9F\u02b4Y'\u0323tp\xe5\xa2\b\x89lk\
x93[\x8b\xbd@\x00\x00\xe0\x94f\xfd\xc9\xfe\xe3Q\xfa\x158\xeb\r\x87\xd8\x19\xfc\xf0\x9e|\x10j\x8
a\x01F'\xb5\xd97\x81\xb2\x00\x00\u07d4g\x04\x8f:\x12\xa4\xdd\x1fbld&L\xb1\u05d7\x1d\xe2\xca8
\x89\t\xc2\x00vQ\xb2P\x00\x00\u07d4g\x04\xf1i\xe0\u0433kW\xbb\u00df<EC{^\xe3\u048d\x89\x1
5[\xd90\u007f\x9f\xe8\x00\x00\u07d4g\x10\x15\xb9vp\xb1\r^X?=b\xa6\x1c\x1cy\xc5\x14?\x89\x15\
xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94g\x10\xc2\xc0<e\x99+.wK\xe5-
:\xb4\xa6\xba!~\xf7\x8a\x02t\xd6V\xac\x90\xe3@\x00\x00\u07d4g\x11\x10\xd9j\xaf\xf1\x15#\xccTk
\xf9\x94\x0e\xed\xff\xb2\xfa\xf7\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4g\x15\xc1@5\xfbW\xbb
=f\u007f{pt\x98\xc4\x10t\xb8U\x89%\xf2s\x93=\xb5p\x00\x00\u07d4g\x1b\xbc\xa0\x99\xff\x89\x9b\
xab\a\xea\x1c\xf8ie\xc3\x05L\x89`\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4g'\xda\xf5\xb9\
u058e\xfc\xabH\x9f\xed\xec\x96\xd7\xf72]\xd4#\x89lk\x93[\x8b\xbd@\x00\x00\u07d4g,\xbc\xa8D\n
\x85w\t{\x19\xaf\xf5\x93\xa2\xad\x9d(\xa7V\x89\x04V9\x18$O@\x00\x00\xe0\x94g.\xc4/\xaa\x8c\u
059a\xaaq\xb3,\u01f4\x04\x88\x1dR\xff\x91\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94g/\
xa0\xa0\x19\b\x8d\xb3\x16oa\x19C\x8d\a\xa9\x9f\x8b\xa2$\x8a\x02\xd4\xca\x05\xe2\xb4<\xa8\x0
0\x00\u07d4g1D\xf0\xec\x14.w\x0fH4\xfe\xe0\xee1\x182\xf3\b{\x89\x1b\x1bk\u05efd\xc7\x00\x00\
u07d4g5\vS1\x92o^(\xf3\xc1\xe9\x86\xf9dC\x80\x9c\x8b\x8c\x89\x13\x14\xfb7\x06)\x80\x00\x00\x
e0\x94g7\x06\xb1\xb0\xe4\xdcz\x94\x9azybX\xa5\xb8;\xb5\xaa\x83\x8a\x03h\xc8b:\x8bM\x10\x00\

x00\u07d4gB\xa2\xcf\u038dy\xa2\u0125\x1bwtt\x98\x91\"E\xcdj\x89\r\xfd[\x80\xb7\xe4h\x00\x00\u07d4gJ\xdb!\xdfL\x98\u01e3G\xacL<$&gW\xddp9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4gQ\x8e]\x02\xb2\x05\x18\x0f\x04c\xa3
\x04G\x1fu<R>\x89k\x91\x8a\xacIK\x16\x80\x00\xe0\x94g]\\\xaa`\x9b\xf7\n\x18\xac\xa5\x80F]\x8f\xb71\r\x1b\xbb\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4gc F\u0732ZT\x93i(\xa9oB?3
\xcb\ud489lk\x93[\x8b\xbd@\x00\x00\u07d4ge\xdf%(\x0e\x8eO8\u0531\xcfDo\xc5\xd7\xebe\x9e4\x89\x05k\xc7^-
c\x10\x00\x00\u07d4gv\xe13\xd9\xdc5L\x12\xa9Q\b{c\x96P\xf59\xa43\x89\x06\x81U\xa46v\xe0\x00\x00\u07d4g\x85Q<\xf72\xe4~\x87g\ap\xb5A\x9b\xe1\f\xd1\xfct\x89lk\x93[\x8b\xbd@\x00\x00\u07d4g\x947\xea\xcfCxx\xdc)=H\xa3\x9c\x87\xb7B\x1a!l\x89\x03\u007f\x81\x82\x1d\xb2h\x00\x00\u07d4g\x9b\x9a\x10\x990Q~\x89\x99\t\x9c\xcf*\x91LL\x8d\xd94\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4g\xa8\x0e\x01\x90r\x1f\x949\rh\x02r\x9d\xd1,1\xa8\x95\xad\x89lk\x13u\xbc\x91V\x00\x00\u07d4g\xb8\xa6\xe9\x0f\xdf\n\x1c\xacD\x17\x930\x1e\x87P\xa9\xfayW\x890\x84\x9e\xbe\x166\x9c\x00\x00\u07d4g\xbc\x85\xe8}\xc3LN\x80\xaa\xfa\x06k\xa8\u049d\xbb\x8eC\x8e\x89\x15\xd1\xcfAv\xae\xba\x00\x00\u07d4g\xc9&\t>\x9b\x89'\x938\x10\u0642\"\xd6.+\x82\x06\xbb\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4g\xcf\xdanp\xbfvW\u04d0Y\xb5\x97\x90\xe5\x14Z\xfd\xbea\x89#\x05\r\tXfX\x00\x00\u07d4g\u0582\xa2\x82\xefs\xfb\x8dn\x90q\xe2aOG\xab\x1d\x0f^\x8965\u026d\xc5\u07a0\x00\x00\u07d4g\u05a8\xaa\x1b\xf8\xd6\xea\xf78N\x99=\xfd\xf1\x0f\n\xf6\x8aa\x89\n\xbc\xbbW\x18\x97K\x80\x00\u07d4g\u0692.\xff\xa4r\xa6\xb1$\xe8N\xa8\xf8k$\xe0\xf5\x15\xaa\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4g\xdf$-
$\r\u0538\a\x1dr\xf8\xfc\xf3[\xb3\x80\x9dq\xe8\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4g\xee@n\xa4\xa7\xaej:8\x1e\xb4\xed\xd2\xf0\x9f\x17KI(\x898)c_\th\xb0\x00\x00\u07d4g\xf2\xbbx\xb8\xd3\xe1\x1f|E\x8a\x10\xb5\xc8\xe0\xa1\xd3tF}\x89a\t=|,m8\x00\x00\u07d4g\xfcR}\xce\x17\x85\xf0\xfb\x8b\xc7\xe5\x18\xb1\xc6i\xf7\xec\u07f5\x89\r\x02\xabHl\xed\xc0\x00\x00\u07d4h\x02}\x19U\x8e\xd73\x9a\b\xae\xe8\xde5Y\xbe\x06>\xc2\xea\x89lk\x93[\x8b\xbd@\x00\x00\u07d4h\x06@\x83\x8b\xd0zD{\x16\x8dm\x92;\x90\xcflC\xcd\u0289]\u0212\xaa\x111\xc8\x00\x00\u07d4h\a\xdd\u020d\xb4\x89\xb03\xe6\xb2\xf9\xa8\x15SW\x1a\xb3\xc8\x05\x89\x01\x9f\x8euY\x92L\x00\x00\xe0\x94h\rY\x11\xed\x8d\xd9\xee\xc4\\\x06\f\"?\x89\xa7\xf6
\xbb\u054a\x04<3\xc1\x93ud\x80\x00\x00\u07d4h\x11\xb5L\u0456c\xb1\x1b\x94\xda\x1d\xe2D\x82\x85\u035fh\u0649;\xa1\x91\v\xf3A\xb0\x00\x00\u07d4h\x19\f\xa8\x85\xdaB1\x87L\x1c\xfbB\xb1X\n!s\u007f8\x89\xcf\x15&@\xc5\xc80\x00\x00\xe0\x94h(\x97\xbcO\x8e\x89\x02\x91
\xfc\xff\xb7\x87\xc0\x1a\x93\xe6A\x84\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4h)^\x8e\xa5\xaf\xd9\t?\xc0\xa4e\xd1W\x92+]*\xe24\x89\x01\x15NS!}\xdb\x00\x00\u07d4h.\x96'oQ\x8d1\xd7\xe5n0\u07f0\t\xc1!\x82\x01\xbd\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4h5\xc8\xe8\xb7J,\xa2\xae?J\x8d\x0fk\x95J>*\x83\x92\x89\x03B\x9c3]W\xfe\x00\x00\u07d4h63\x01\n\x88hk\xeaZ\x98\xeaS\xe8y\x97\xcb\xf7>i\x89\x05k9Bc\xa4\f\x00\x00\u07d4h=\xba6\xf7\xe9O@\xeaj\xea\ry\xb8\xf5!\xdeU\an\x89\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4hA\x9cm\xd2\xd3\xceo\u02f3\xc7>/\xa0y\xf0`Q\xbd\xe6\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4hG;z}\x96Y\x04\xbe\u06e5V\u07fc\x17\x13l\xd5\xd44\x89\x05k\xc7^-
c\x10\x00\x00\u07d4hG\x82[\xde\xe8$\x0e(\x04,\x83\xca\xd6B\U000868fd\u0709QP\xae\x84\xa8\xcd\xf0\x00\x00\xe0\x94hJD\xc0i3\x9d\b\xe1\x9auf\x8b\u06e3\x03\xbe\x85S2\x8a\x0e\u04b5%\x84\x1a\xdf\xc0\x00\x00\u07d4hS\x1fM\u0680\x8fS

vz\x03\x114(\xca\f\xe2\xf3\x89\x89\x01\r:\xa56\xe2\x94\x00\x00\u07d4hy'\xe3\x04\x8b\xb5\x16*\xe7\xc1\\\xf7k\xd1$\xf9I{\x9e\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94h\x80\x9a\xf5\xd52\xa1\x1c\x1aMn2\xaa\xc7\\LR\xb0\x8e\xad\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4h\x86\xad\xa7\xbb\xb0a{\u0684!\x91\u018c\x92.\xa3\xa8\xac\x82\x89>\xe2;\xde\x0e}
\x00\x00\xe0\x94h\x88>\x15.V`\xfe\xe5\x96&\xe7\xe3\xb4\xf0Q\x10\xe6\"/\x8a\v\x94c;\xe9u\xa6*\x00\x00\u07d4h\x8aV\x9e\x96U$\xeb\x1d\n\xc3\xd3s>\xab\x90\x9f\xb3\xd6\x1e\x89G\x8e\xae\x0eW\x1b\xa0\x00\x00\xe0\x94h\x8e\xb3\x85;\xbc\xc5\x0e\xcf\xee\x0f\xa8\u007f\n\xb6\x93\u02bd\xef\x02\x8a\x06\xb1\n\x18@\x06G\xc0\x00\x00\u07d4h\xa7B_\xe0\x9e\xb2\x8c\xf8n\xb1y>A\xb2\x11\xe5{\u058d\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4h\xa8l@#\x88\xfd\xdcY\x02\x8f\xecp!\u933f\x83\x0e\xac\x89\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\xe0\x94h\xac\u06a9\xfb\x17\xd3\xc3\t\x91\x1aw\xb0_S\x91\xfa\x03N\xe9\x8a\x01\xe5.3l\xde\"\x18\x00\x00\u07d4h\xad\xdf\x01\x9dk\x9c\xabp\xac\xb1?\v1\x17\x99\x9f\x06.\x12\x89\x02\xb5\x12\x12\xe6\xb7\u0200\x00\u07d4h\xb3\x186\xa3\n\x01j\xda\x15{c\x8a\xc1]\xa7?\x18\xcf\u0789\x01h\u048e?\x00(\x00\x00\xe0\x94h\xb6\x85G\x88\xa7\xc6Il\xdb\xf5\xf8K\x9e\xc5\xef9+x\xbb\x8a\x04+\xf0kx\xed;P\x00\x00\u07d4h\xc0\x84\x90\u021b\xf0\u05b6\xf3
\xb1\xac\xa9\\\x83\x12\xc0\x06\b\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4h\xc7\xd1q\x1b\x01\x1a3\xf1o\x1fU\xb5\xc9\x02\xcc\xe9p\xbd\u05c9\b=lz\xabc`\x00\x00\u07d4h\xc8y\x1d\xc3B\xc3sv\x9e\xa6\x1f\xb7\xb5\x10\xf2Q\xd3
\x88\x8965\u026d\xc5\u07a0\x00\x00\u07d4h\u07d4|I[\ubbb8\u8273\xf9S\xd53\x87K\xf1\x06\x89\x1d\x99E\xab+\x03H\x00\x00\u07d4h\xe8\x02'@\xf4\xaf)\xebH\xdb2\xbc\xec\xdd\xfd\x14\x8d=\xe3\x8965\u026d\xc5\u07a0\x00\x00\u07d4h\xecy\u057eqUql@\x94\x1cy\u05cd\x17\u079e\xf8\x03\x89\x1b#8w\xb5
\x8c\x00\x00\u07d4h\xee\xc1\u222c1\xb6\xea\xba~\x1f\xbdO\x04\xadW\x9ak]\x89lk\x93[\x8b\xbd@\x00\x00\u07d4h\xf5%\x92\x1d\xc1\x1c2\x9buO\xbf>R\x9f\xc7#\xc84\u0349WG=\x05\u06ba\xe8\x00\x00\u07d4h\xf7\x19\xae4+\xd7\xfe\xf1\x8a\x05\u02f0/pZ\u04ce\u0572\x898\xeb\xad\\\u0710(\x00\x00\xe0\x94h\xf7W<\xd4W\xe1L\x03\xfe\xa4>0-04|\x10p\\\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4h\xf8\xf4QU\xe9\x8cP)\xa4\xeb\u0175'\xa9.\x9f\xa81 \x89\xf0{D\xb4\a\x93
\x80\x00\u07d4h\xfe\x13W!\x8d\tXI\xcdW\x98B\u012a\x02\xff\x88\x8d\x93\x89lk\x93[\x8b\xbd@\x00\x00\u07d4i\x02(\xe4\xbb\x12\xa8\u0535\u09d7\xb0\xc5\xcf*u\t\x13\x1e\x89e\xea=\xb7UF`\x00\x00\u07d4i\x05\x94\xd3\x06a<\xd3\xe2\xfd$\xbc\xa9\x99J\u064a=s\xf8\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94i\a2ir\x9ed\x14\xb2n\xc8\xdc\x0f\xd95\xc7;W\x9f\x1e\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\xe0\x94i\x19\xdd^]\xfb\x1a\xfa@G\x03\xb9\xfa\xea\x8c\xee5\xd0\rp\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d4i4\x92\xa5\xc5\x13\x96\xa4\x82\x88\x16i\xcc\xf6\xd8\xd7y\xf0\tQ\x89\x12\xbfPP:\xe3\x03\x80\x00\u07d4i=\x83\xbe\tE\x9e\xf89\v.0\xd7\xf7\u008d\xe4\xb4(N\x89lk\x93[\x8b\xbd@\x00\x00\u07d4iQp\x83\xe3\x03\xd4\xfb\xb6\xc2\x11E\x14!]i\xbcF\xa2\x99\x89\x05k\xc7^-c\x10\x00\x00\u07d4iUPel\xbf\x90\xb7]\x92\xad\x91\"\xd9\r#\xcah\xcaM\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94iX\xf8;\xb2\xfd\xfb'\xce\x04\t\xcd\x03\xf9\xc5\xed\xbfL\xbe\u074a\x04<3\xc1\x93ud\x80\x00\x00\u0794i[\x0fRBu7\x01\xb2d\xa6pq\xa2\u0708\b6\xb8\u06c8\u3601\x1b\xech\x00\x00\xe0\x94i[L\xce\bXV\xd9\xe1\xf9\xff>y\x94 #5\x9e_\xbc\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x94if\x06:\xa5\xde\x1d\xb5\xc6q\xf3\xddi\x9dZ\xbe!>\xe9\x02\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4it\u0224\x14\u03ae\xfd<.M\xfd\xbe\xf40V\x8d\x9a\x96\v\x89\x12\x1e\xa6\x8c\x11NQ\x00\x00\xe0\x94iximQP\xa9\xa2cQ?\x8ft\u0196\xf8\xb19|\xab\x8a\x01g\xf4\x82\xd3\u0171

\xc0\x00\x00\xe0\x94iy{\xfb\x12\u027e\u0582\xb9\x1f\xbcY5\x91\xd5\xe4\x027(\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94i\u007fUSk\xf8Z\xdaQ\x84\x1f\x02\x87b:\x9f\x0e\u041a\x17\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4i\x82\xfe\x8a\x86~\x93\xebJ\v\xd0QX\x93\x99\xf2\xec\x9aR\x92\x89lk\x93[\x8b\xbd@\x00\x00\u07d4i\x8a\x8ao\x01\xf9\xabh/c|yi\xbe\x88_IS\x02\xbf\x89\x01\r:\xa56\xe2\x94\x00\x00\u07d4i\x8a\xb9\xa2\xf33\x81\xe0|\fGC=\r!\xd6\xf36\xb1'\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4i\x94\xfb21\xd7\xe4\x1dI\x1a\x9dh\xd1\xfaL\xae,\xc1Y`\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4i\x9c\x9e\xe4q\x95Q\x1f5\xf8b\xcaL\"\xfd5\xae\x8f\xfb\xf4\x89\x04V9\x18$O@\x00\x00\u07d4i\x9f\xc6\u058aGuW<\x1d\u036e\xc80\xfe\xfdP9|N\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4i\xaf(\xb0tl\xac\r\xa1p\x84\xb99\x8c^6\xbb:\r\xf2\x896w\x03n\xdf\n\xf6\x00\x00\u07d4i\xb8\x0e\xd9\x0f\x84\x83J\xfa?\xf8.\xb9dp;V\tw\u0589\x01s\x17\x90SM\xf2\x00\x00\xe0\x94i\xb8\x1dY\x81\x14\x1e\u01e7\x14\x10`\xdf\u03cf5\x99\xff\xc6>\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4i\xbc\xfc\x1dC\xb4\xba\x19\xde{'K\xdf\xfb5\x13\x94\x12\xd3\u05c95e\x9e\xf9?\x0f\xc4\x00\x00\u07d4i\xbd%\xad\xe1\xa34lY\xc4\xe90\xdb*\x9dq^\xf0\xa2z\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4i\xc0\x8dtGT\xdep\x9c\xe9n\x15\xae\r\x1d9[:\"c\x8965\u026d\xc5\u07a0\x00\x00\u07d4i\xc2\xd85\xf1>\xe9\x05\x80@\x8ej2\x83\xc8\u0326\xa44\xa2\x89#\x8f\xd4,\\\xf0@\x00\x00\u07d4i\xc9N\a\u0129\xbe3\x84\xd9]\xfa<\xb9)\x00Q\x87;{\x89\x03\xcbq\xf5\x1f\xc5X\x00\x00\u07d4i\xcb>!S\x99\x8d\x86\xe5\xee
\xc1\xfc\u0466\xba\uec86?\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4i\u04ddQ\b\x89\xe5R\xa3\x96\x13[\xfc\xdb\x06\xe3~8v3\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94i\u064f8\xa3\xba=\xbc\x01\xfa\\,\x14'\xd8b\x83//p\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94i\xe2\xe2\xe7\x040|\xcc[\\\xa3\xf1d\xfe\xce.\xa7\xb2\xe5\x12\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4i\xffB\x90t\u02dblc\xbc\x91B\x84\xbc\xe5\xf0\xc8\xfb\xf7\u0409\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4i\xff\x89\x01\xb5Av?\x81|_)\x98\xf0-
\xcf\xc1\xdf)\x97\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4j\x02:\xf5}XM\x84^i\x876\xf10\u06dd\xb4\r\xfa\x9a\x89\x05[
\x1c\x89\x00\x98\x00\x00\u07d4j\x04\xf5\xd5?\xc0\xf5\x15\xbe\x94+\x8f\x12\xa9\xcbz\xb0\xf3\x97x\x89\xa9\xaa\xb3E\x9b\xe1\x94\x00\x00\u07d4j\x05\xb2\x1cO\x17\xf9\xd7?_\xb2\xb0\u02c9\xffSV\xa6\xcc~\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\xe0\x94j\x0f\x05`f\xc2\xd5f(\x85\x02s\xd7\xec\xb7\xf8\xe6\xe9\x12\x9e\x8a\x01\x0f\r)<\u01e5\x88\x00\x00\u07d4j\x13\xd5\xe3,\x1f\xd2m~\x91\xffn\x051`\xa8\x9b,\x8a\xad\x89\x02\xe6/
\xa6\x9b\xe4\x00\x00\u07d4j.\x86F\x9a[\xf3|\xee\x82\xe8\x8bL8c\x89](\xfc\xaf\x89\x1c\"\x92f8[\xbc\x00\x00\u07d4j6\x94BL|\u01b8\xbc\u067c\u02baT\f\xc1\xf5\xdf\x18\u05c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94jB\u0297\x1cex\u056d\xe2\x95\xc3\xe7\xf4\xad3\x1d\xd3BN\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4jD\xaf\x96\xb3\xf02\xaed\x1b\xebg\xf4\xb6\xc83B\xd3|]\x89\x01\x92t\xb2Y\xf6T\x00\x00\u07d4jL\x89\a\xb6\x00$\x80W\xb1\xe4cT\xb1\x9b\u0705\x9c\x99\x1a\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4jQNbB\xf6\xb6\x8c\x13~\x97\xfe\xa1\u73b5U\xa7\xe5\xf7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4jS\xd4\x1a\xe4\xa7R\xb2\x1a\xbe\xd57FI\x95:Q=\xe5\xe5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4jaY\aJ\xb5s\xe0\xeeX\x1f\x0f=\xf2\u05a5\x94b\x9bt\x89\x10\xce\x1d=\x8c\xb3\x18\x00\x00\u07d4jc7\x83?\x8fjk\xf1\f\xa7\xec!\xaa\x81\x0e\xd4D\xf4\u02c97\xbd$4\\\xe8\xa4\x00\x00\u07d4jcS\xb9qX\x9f\x18\xf2\x95\\\xba(\xab\xe8\xac\xcejWa\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4jc\xfc\x89\xab\xc7\xf3n(-
\x80x{{\x04\xaf\xd6U>q\x89\b\xacr0H\x9e\x80\x00\x00\u07d4jg\x9e7\x8f\xdc\xe6\xbf\xd9\u007f\xe

6/\x04<od\x05\u05de\x99\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4jhk\xf2
\xb5\x93\u07b9\xb72F\x15\xfb\x91D\xde\xd3\xf3\x9d\x89O%\x91\xf8\x96\xa6P\x00\x00\xe0\x94jk\
x18\xa4ZvF~.]Z.\xf9\x11\xc3\xe1))\x85{\x8a\x11]:\x99\xa9aO@\x00\x00\u07d4jt\x84M\x8e\x9c\xb5
X\x1cE\a\x9a.\x94F*l\xee\x88!\x89:\xb5:U-\xd4\xc9\x00\x00\xe0\x94j{.\r\x88\x86\u007f\xf1]
|\"+\xeb\xf9O\xa6\u0383\x97\x8a\f\xb4\x9bD\xba`-\x80\x00\x00\u07d4j|%
B\xe7F\x8a?\xf7s\xd6E\v\xba\x85\xef\xa2c\x91\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4j\x8a
C\x17\xc4_\xaa\x05T\xcc\xdbH%H\x18>)Z$\xb9\x8965\u026d\xc5\u07a0\x00\x00\u07d4j\x8c\xea-
\xe8J\x8d\xf9\x97\xfd?\x84\xe3\b=\x93\xdeW\u0369\x89\x05k\xe0<\xa3\xe4}\x80\x00\xe0\x94j\x97
Xt;`>\xea:\xa0RKB\x88\x97#\xc4\x159H\x8a\x02#\x85\xa8'\xe8\x15P\x00\x00\u07d4j\xa5s/;\x86\xf
b\x8c\x81\xef\xbek[G\xb5cs\v\x06\u020965\u026d\xc5\u07a0\x00\x00\u07d4j\xb3#\xaePV\xed\nE
0r\u016b\xe2\xe4/\xcf]q9\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4j\xb5\xb4\xc4\x1c\u0778)i\f/\xda\u007f
\xc8^b\x9d\xd5\u0549d\u052fqL2\x90\x00\x00\u07d4j\xc4\x0fS-\xfe\xe5\x11\x81\x17\u04ad5-
\xa7}Om\xa2\u0209\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4j\xc4\u053e-
\xb0\u065d\xa3\xfa\xaa\xf7RZ\xf2\x82\x05\x1dj\x90\x89\x04X\xcaX\xa9b\xb2\x80\x00\u07d4j\xcd\
u0723\xcd+I\x90\xe2\\\xd6\\$\x14\x9d\t\x12\t\x9ey\x89\xa2\xa1\xe0|\x9fl\x90\x80\x00\u07d4j\xd9\v\
xe2R\xd9\xcdFM\x99\x81%\xfa\xb6\x93\x06\v\xa8\xe4)\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d
4j\u0753!\x93\xcd8IJ\xa3\xf0:\xec\xccKz\xb7\xfa\xbc\xa2\x89\x04\xdbs%Gc\x00\x00\x00\xe0\x94j\
xe5\u007f'\x91|V*\x13*M\x1b\xf7\xec\n\u01c5\x83)&\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4j\
xeb\x9ftt.\xa4\x91\x81=\xbb\xf0\xd6\xfc\xde\x1a\x13\x1dM\xb3\x89\x17\xe5T0\x8a\xa00\x00\x00\u
07d4j\xf25\u04bb\xe0P\xe6)\x16\x15\xb7\x1c\xa5\x82\x96X\x81\x01B\x89\xa2\xa1]\tQ\x9b\xe0\x0
0\x00\u07d4j\xf6\xc7\xee\x99\xdf'\x1b\xa1[\xf3\x84\xc0\xb7d\xad\xcbM\xa1\x82\x8965f3\xeb\xd8\x
ea\x00\x00\u07d4j\xf8\xe5Yih,q_H\xadO\xc0\xfb\xb6~\xb5\x97\x95\xa3\x89lk\x93[\x8b\xbd@\x00\
x00\u07d4j\xf9@\xf6>\u0278\xd8v'*\u0296\xfe\xf6\\\xda\xce\xcd\ua262\xa1]\tQ\x9b\xe0\x00\x00\u
07d4j\xf9\xf0\xdf\uebbb_d\xbf\x91\xabw\x16i\xbf\x05)US\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07
d4j\xff\x14f\xc2b6u\xe3\xcb\x0eu\xe4#\xd3z%\xe4B\xeb\x89]\u0212\xaa\x111\xc8\x00\x00\xe0\x9
4k\r\xa2Z\xf2g\u05c3l\"k\xca\xe8\xd8r\xd2\xceR\xc9A\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d
4k\x10\xf8\xf8\xb3\xe3\xb6\r\xe9\n\xa1-
\x15_\x9f\xf5\xff\xb2,P\x89lk\x93[\x8b\xbd@\x00\x00\u07d4k\x17Y\x8a\x8e\xf5Oyz\xe5\x15\u0336
Q}\x18Y\xbf\x80\x11\x89\x05k\xc7^-c\x10\x00\x00\u07d4k
\xc0\x80`jy\xc7;\xd8\xe7[\x11qzN\x8d\xb3\xf1\u00c9\x10?sX\x03\xf0\x14\x00\x00\u07d4k\"\x84D\x
02!\xce\x16\xa88-
\xe5\xff\x02)G\"i\xde\xec\x8965\u026d\xc5\u07a0\x00\x00\u07d4k0\xf1\x829\x10\xb8m:\xcbZj\xfc\
x9d\xef\xb6\xf3\xa3\v\xf8\x89\u3bb5sr@\xa0\x00\x00\u07d4k8\u0784\x1f\xad\u007fS\xfe\x02\xda\
x11[\xd8j\xaff$f\xbd\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4kK\x99\xcb?\xa9\xf7\xb7L\u3903\x17\
xb1\xcd\x13\t\n\x1az\x89\x03\x1b2~i]\xe2\x00\x00\u07d4kZ\xe7\xbfx\xecu\xe9\f\xb5\x03\xc7x\xcc\
u04f2KO\x1a\xaf\x89+^:\xf1k\x18\x80\x00\x00\u07d4kc\xa2\u07f2\xbc\xd0\xca\xec\x00\"\xb8\x8b\x
e3\f\x14Q\xeaV\xaa\x89+\xdbk\xf9\x1f\u007fL\x80\x00\u07d4kew\xf3\x90\x9aMm\xe0\xf4\x11R-
Ep8d\x004\\\x89e\xea=\xb7UF`\x00\x00\u07d4kr\xa8\xf0a\xcf\xe6\x99j\xd4G\xd3\xc7,(\xc0\xc0\x8
a\xb3\xa7\x89\xe7\x8cj\u01d9\x12b\x00\x00\u07d4kv\rHw\xe6\xa6'\xc1\xc9g\xbe\xe4Q\xa8P}\xdd\
u06eb\x891T\xc9r\x9d\x05x\x00\x00\u07d4k\x83\xba\xe7\xb5e$EXU[\xcfK\xa8\xda
\x11\x89\x1c\x17\x89lk\x93[\x8b\xbd@\x00\x00\u07d4k\x92]\xd5\xd8\xeda2\xabm\b`\xb8,D\xe1\xa
5\x1f\x1f\xee\x89P; >\x9f\xba

\x00\x00\xe0\x94k\x94a]\xb7Pej\u00cc~\x1c\xf2\x9a\x9d\x13g\u007fN\x15\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4k\x95\x1aC'N\xea\xfc\x8a\t\x03\xb0\xaf.\xc9+\xf1\xef\xc89\x89\x05k\xc7^-c\x10\x00\x00\u07d4k\x99%!\xec\x85#p\x84\x8a\u0597\xcc-\xf6Nc\xcc\x06\xff\x8965\u026d\xc5\u07a0\x00\x00\u07d4k\xa8\xf7\xe2_\xc2\xd8qa\x8e$\xe4\x01\x84\x19\x917\xf9\xf6\xaa\x89\x15\xafd\x86\x9ak\xc2\x00\x00\u07d4k\xa9\xb2\x1b5\x10k\xe1Y\xd1\xc1\xc2ez\xc5l\u049f\xfdD\x89\xf2\xdc}G\xf1V\x00\x00\x00\u07d4k\xafz*\x02\xaex\x80\x1e\x89\x04\xadz\xc0Q\b\xfcV\xcf\xf6\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94k\xb2\xac\xa2?\xa1bm\x18\xef\xd6w\u007f\xb9}\xb0-\x8e\n\xe4\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4k\xb4\xa6a\xa3:q\xd4$\u051b\xb5\xdf(b.\xd4\xdf\xfc\xf4\x89\",\x8e\xb3\xfff@\x00\x00\u07d4k\xb5\b\x13\x14j\x9a\xddB\xee\"\x03\x8c\x9f\x1fti\xd4\u007fG\x89\n\xdaUGK\x814\x00\x00\u07d4k\xbc?5\x8af\x8d\u0461\x1f\x03\x80\xf3\xf71\bBj\xbdJ\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4k\xbd\x1eq\x93\x90\xe6\xb9\x10C\xf8\xb6\xb9\u07c9\x8e\xa8\x00\x1b4\x89llO\xa6\xc3\xdaX\x80\x00\u07d4k\xc8Z\xcdY(r.\xf5\tS1\xee\x88\xf4\x84\xb8\u03c3W\x89\t\xc2\x00vQ\xb2P\x00\x00\u07d4k\xd3\xe5\x9f#\x9f\xaf\xe4wk\xb9\xbd\xddk\ue0fa]\x9d\x9f\x8965\u026d\xc5\u07a0\x00\x00\u07d4k\xd4W\xad\xe0Qy]\xf3\xf2F\\89\xae\xd3\xc5\xde\xe9x\x8964\xbf9\xab\x98x\x80\x00\u07d4k\xe1c\x13d>\xbc\x91\xff\x9b\xb1\xa2\xe1\x16\xb8T\xea\x93:E\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4k\xe7Y^\xa0\xf0hH\x9a'\x01\xecFl\x15\x8d\xdcC\xe1x\x89lk\x93[\x8b\xbd@\x00\x00\u07d4k\xe9\x03\x0e\xe6\xe2\xfb\u0111\xac\xa3\xde@\"\xd3\x01w+{}\x89\x01s\x17\x90SM\xf2\x00\x00\xe0\x94k\xec1\x1a\xd0P\b\xb4\xaf5<\x95\x8c@\xbd\x06s\x9a?\xf3\x8a\x03w\xf6*\x0f\nbp\x00\x00\u07d4k\xf7\xb3\xc0e\xf2\xc1\xe7\xc6\xeb\t+\xa0\xd1Pf\xf3\x93\u0478\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4k\xf8o\x1e/+\x802\xa9\\Mw8\xa1\t\xd3\xd0\xed\x81\x04\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4l\x05\xe3N^\xf2\xf4.\u041d\xef\xf1\x02l\xd6k\xcbi`\xbb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4l\b\xa6\xdc\x01s\xc74)U\xd1\xd3\xf2\xc0e\xd6/\x83\xae\u01c9\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4l\n\xe9\xf0C\xc84\xd4Bq\xf14\x06Y=\xfe\tO8\x9f\x89RD*\xe13\xb6*\x80\x00\u07d4l\f\xc9\x17\xcb\xee}|\t\x97c\xf1Nd\xdf}4\xe2\xbf\t\x89\r\x8drkqw\xa8\x00\x00\xe0\x94l\x0eq/@\\Yr_\xe8)\xe9wK\xf4\xdf\u007fM\xd9e\x8a\f(h\x88\x9c\xa6\x8aD\x00\x00\xe0\x94l\x10\x12\x05\xb3#\xd7uD\xd6\xdcR\xaf7\xac\xa3\xce\xc6\xf7\xf1\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4l\x15\xec5 \xbf\x8e\xbb\xc8\xbd\x0f\xf1\x97x7T\x94\u03dd\x89l\xb7\xe7Hg\xd5\xe6\x00\x00\xe0\x94l\x1d\xdd3\xc8\x19f\u07066!w`q\xa4\x12\x94\x82\xf2\xc6_\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4l%2\u007f\x8d\u02f2\xf4^V\x1e\x86\xe3]\x88P\xe5:\xb0Y\x89;\xcd\xf9\xba\xfe\xf2\xf0\x00\x00\u07d4l.\x9b\xe6\u052bE\x0f\xd1%1\xf3?\x02\x8caFt\xf1\x97\x89\xc2\x12z\xf8X\xdap\x00\x00\u07d4l5\x9eX\xa1=Ex\xa93\x8e3\\g\xe7c\x9f_\xb4\u05c9\v\xd1[\x94\xfc\x8b(\x00\x00\u07d4l=\x18pA&\xaa\x99\xee3B\xce`\xf5\xd4\xc8_\x18g\u0349\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4lGK\xc6jTx\x00f\xaaOQ.\xef\xa7s\xab\xf9\x19\u01c9\x05\x18\x83\x15\xf7v\xb8\x00\x00\u07d4lNBn\x8d\xc0\x05\u07e3Ql\xb8\xa6\x80\xb0.\ua56e\x8e\x89Hz\x9a0E9D\x00\x00\u07d4lR\xcf\b\x95\xbb5\xe6V\x16\x1eM\xc4j\xe0\xe9m\xd3\xe6,\x89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4lT\"\xfbK\x14\xe6\u064b`\x91\xfd\xecq\xf1\xf0\x86@A\x9d\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4l\\:T\u0367\xc2\xf1\x18\xed\xbaCN\xd8\x1en\xbb\x11\xddz\x89\n\u05ce\xbcZ\xc6\x00\x00\u07d4lc\xf8EV\u0490\xbf\u0359\xe44\ue657\xbf\xd7yWz\x89lk\x93[\x8b\xbd@\x00\x00\u07d4lc\xfc\x85\x02\x9a&T\u05db+\xeaM\xe3I\xe4REw\u0149#\xc7W\a+\x8d\xd0\x00\x00\u07d4led\xe5\xc9\xc2N\xaa\xa7D\xc9\xc7\xc9h\xc9\xe2\xc9\xf1\xfb\xae\x89I\x9bB\xa2\x119d\x00\x00\xe0

\x94lg\xd6\xdb\x1d\x03Ql\x12\x8b\x8f\xf24\xbf=I\xb2m)A\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4lg\xe0\u05f6.*\bPiE\xa5\xdf\xe3\x82c3\x9f\x1f\"\x89j\xcb=\xf2
~\x1f\x88\x00\x00\u07d4lj\xa0\xd3\vdr\x19\x90\xb9PJ\x86?\xa0\xbf\xb5\xe5}\xa7\x89\x92^\x06\xe
e\xc9r\xb0\x00\x00\u07d4lqJX\xff\xf6\xe9}\x14\xb8\xa5\xe3\x05\xeb$@eh\x8b\xbd\x89\xd8\xd7&\x
b7\x17z\x80\x00\x00\u07d4l\x80\rKI\xba\a%\x04`\xf9\x93\xb8\xcb\xe0\v&j%S\x89\x1a\xb2\xcf|\x9f\
x87\xe2\x00\x00\u07d4l\x80\x8c\xab\xb8\xff_\xbbc\x12\xd9\xc8\xe8J\xf8\xcf\x12\xef\bu\x89\xd8\xd
8X?\xa2\xd5/\x00\x00\xe0\x94l\x82
)!\x8a\xc8\xe9\x8a&\f\x1e\x06@)4\x889\x87[\x8a\x01\x0f\x97\xb7\x87\xe1\xe3\b\x00\x00\u07d4l\x
84\u02e7|m\xb4\xf7\xf9\x0e\xf1=^\xe2\x1e\x8c\xfc\u007f\x83\x14\x89lk\x93[\x8b\xbd@\x00\x00\xe
0\x94l\x86\x87\xe3Aw\x10\xbb\x8a\x93U\x90!\xa1F\x9ej\x86\xbcw\x8a\x02[-
\xa2x\xd9k{\x80\x00\xe0\x94l\x88,'s,\xef\\|\x13\xa6\x86\xf0\xa2\xeawUZ\u0089\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4l\xa5\xde\x00\x81}\xe0\xce\xdc\xe5\xfd\x00\x01(\xde\xde\x12
d\x8b<\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4l\xa6\xa12\xce\x1c\u0488\xbe\xe3\x0e\xc7\xcf\x
ef\xfb\x85\xc1\xf5\nT\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94l\xb1\x1e\xcb2\xd3\u0382\x96\x011\x0
66\xf5\xa1\f\xf7\u03db_\x8a\x04?\u851c8\x01\xf5\x00\x00\u07d4l\xc1\xc8x\xfal\u078a\x9a\v\x83\x
11$~t\x1eFB\xfem\x895e\x9e\xf9?\x0f\xc4\x00\x00\xe0\x94l\xcb\x03\xac\xf7\xf5<\xe8z\xad\xcc!\xa
9\x93-
\xe9\x15\xf8\x98\x04\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4l\xd2\x12\xae\xe0N\x01?=*\xba\u
04a0#`k\xfb\\j\u01c9lj\xccg\u05f1\xd4\x00\x00\u07d4l\xd2(\xdcq!i0\u007f\xe2|\xebtw\xb4\x8c\xfc\x
82r\xe5\x89\x044\xea\x94\u06caP\x00\x00\u07d4l\xe1\xb0\xf6\xad\xc4pQ\xe8\xab8\xb3\x9e\xdbA\
x86\xb0;\xab\u0309Ay\x97\x94\xcd$\xcc\x00\x00\u07d4l\xea\xe3s=\x8f\xa4=l\xd8\f\x1a\x96\xe8\xe
b\x93\x10\x9c\x83\xb7\x89\x10'\x94\xad
\xdah\x00\x00\u07d4m\x05i\xe5U\x8f\xc7\xdf'f\xf2\xba\x15\u070a\xef\xfc[\xebu\x89\xd8\xe6\x00\x
1el0+\x00\x00\u07d4m\x12\x0f\f\xaa\xe4O\xd9K\xca\xfeU\xe2\xe2y\uf5ba\\z\x89\xd8\xd7&\xb7\x1
7z\x80\x00\x00\u07d4m\x14V\xff\xf0\x10N\xe8D\xa31G7\x8438\xd2L\xd6l\x89\a\xb0l\xe8\u007f\xd
dh\x00\x00\u07d4m
\xef\x97\x04g\nP\v\xb2i\xb5\x83.\x85\x98\x02\x04\x9f\x01\x89\a\f\x1c\xc7;\x00\xc8\x00\x00\xe0\x9
4m/\x97g4\xb9\xd0\a\r\x18\x83\xcfz\u02b8\xb3\xe4\x92\x0f\xc1\x8a\x02\x1e\x19\xe0\u027a\xb2@
\x00\x00\u07d4m9\xa9\u93c1\xf7i\xd7:\xad,\xea\xd2v\xac\x13\x87\xba\xbe\x89\x15[\xd90\u007f\x9
f\xe8\x00\x00\u07d4m;x6\xa2\xb9\u0619r\x1aM#{R#\x85\xdc\xe8\xdf\u034966\xc2^f\xec\xe7\x00\
x00\u07d4m?+\xa8V\u033b\x027\xfava\x15k\x14\xb0\x13\xf2\x12@\x8965\u026d\xc5\u07a0\x00\x
00\xe0\x94m@\b\xb4\xa8\x88\xa8&\xf2H\xeej\v\r\xfd\xe9\xf92\x10\xb9\x8a\x01'\xfc\xb8\xaf\xae
\xd0\x00\x00\u07d4m@\xca'\x82m\x97s\x1b>\x86\xef\xfc\u05f9*Aa\xfe\x89\x89lk\x93[\x8b\xbd@\x
00\x00\u07d4mD\x97J1\u0447\xed\xa1m\xddG\xb9\xc7\xecP\x02\xd6\x1f\xbe\x892\xf5\x1e\u06ea
\xa30\x00\x00\xe0\x94mK\\\x05\xd0j
\x95~\x17H\xabm\xf2\x06\xf3C\xf9/\x01\x8a\x02\x1f6\x06\x99\xbf\x82_\x80\x00\xe0\x94mL\xbf=\x
82\x84\x83:\xe9\x93D0>\b\xb4\xd6\x14\xbf\xda;\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4mY\xb2\
x1c\xd0\xe2t\x88\x04\u066b\xe0d\xea\u00be\xf0\xc9_'\x89lk\x93[\x8b\xbd@\x00\x00\u07d4mc\u0
4ce\xe8\xb9\x0e\x0en\xd8\xf1\x92\xed\xa0Q\xb2\u05a5\x8b\xfd\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\
u07d4mf4\xb5\xb8\xa4\x01\x95\xd9I\x02z\xf4\x82\x88\x02\t,\ued89\xa2\xa1]\tQ\x9b\xe0\x00\x00\x
e0\x94m}\x1c\x94\x95\x11\xf8\x83\x03\x80\x8c`\xc5\xea\x06@\xfc\xc0&\x83\x8a\x02\x1e\x19\xe0\
u027a\xb2@\x00\x00\u07d4m\x84m\xc1&W\xe9\x1a\xf2P\bQ\x9c>\x85\u007fQp}\u0589\xf8\xd3\v\

xc9#B\xf8\x00\x00\u07d4m\x91\x93\x99k\x19F\x17!\x11\x06\xd1c^\xb2l\u0136ll\x89\x15\xaa\x1e~\x9d\xd5\x1c\x00\x00\u07d4m\x99\x97P\x98\x82\x02~\xa9G#\x14$\xbe\xde\xde)e\u043a\x89l\x81\u01f3\x11\x95\xe0\x00\x00\u07d4m\xa0\xed\x8f\x1di3\x9f\x05\x9f*\x0e\x02G\x1c\xb4O\xb8\u00fb\x892\xbc8\xbbc\xa8\x16\x00\x00\u07d4m\xb7+\xfdC\xfe\xf4e\xcaV2\xb4Z\xabra@N\x13\xbf\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94m\xbe\x8a\xbf\xa1t(\x06&9\x817\x1b\xf3\xd3U\x90\x80kn\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4m\xc3\xf9+\xaa\x1d!\u06b78+\x892a\xa05o\xa7\xc1\x87\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4m\xc7\x05:q\x86\x16\xcf\u01cb\xeec\x82\xeeQ\xad\xd0\xc7\x030\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94m\xcc~d\xfc\xaf\xcb\xc2\xdcl\x0e^f,\xb3G\xbf\xfc\xd7\x02\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4m\xda_x\x8alh\x8d\u07d2\x1f\xa3\x85.\xb6\xd6\xc6\xc6)f\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4m\xdb`\x92w\x9dXB\xea\xd3x\xe2\x1e\x81
\xfdLk\xc12\x89lk\x93[\x8b\xbd@\x00\x00\u07d4m\xdf\xefc\x91U\u06ab\n\\\xb4\x95:\xa8\u016f\xaa\x88\x04S\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4m\xe0/-
\xd6~\xfd\xb794\x02\xfa\x9e\xaa\xcb\xcfX\x9d.V\x89@\x13\x8b\x91~\u07f8\x00\x00\u07d4m\u4d418\\\xf7\xfc\x9f\xe8\xc7}\x13\x1f\xe2\xeew$\xc7j\x89})\x97s=\xcc\xe4\x00\x00\u07d4m\xe4\xd1R\x19\x18/\xaf:\xa2\xc5\xd4\xd2Y_\xf20\x91\xa7'\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4m\xed\xf6.t?M,*K\x87\xa7\x87\xf5BJz\xeb9<\x89\t\xc2\x00vQ\xb2P\x00\x00\u07d4m\xf2Of\x85\xa6/y\x1b\xa37\xbf?\xf6~\x91\xf3\u053c:\x89ukI\xd4\nH\x18\x00\x00\u07d4m\xf5\xc8O{\x90\x9a\xab>a\xfe\x0e\xcb\x1b;\xf2`\"*\u0489\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4m\xff\x90\xe6\xdc5\x9d%\x90\x88+\x14\x83\xed\xbc\xf8\x87\xc0\xe4#\x8965\u026d\xc5\u07a0\x00\x00\u07d4n\x01\xe4\xadV\x9c\x95\xd0\a\xad\xa3\r^-
\xb1(\x88I\"\x94\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4n\a;f\u0478\xc6gD\u0600\x96\xa8\u0759\xec~\x02(\u0689\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4n\x0e\xe7\x06\x12\xc9v(}I\x9d\u07e6\xc0\xdc\xc1,\x06\xde\xea\x89\a\v\u0579V!F\x00\x00\xe0\x94n\x12\xb5\x1e\"[JCr\xe5\x9a\u05e2\xa1\xa1>\xa3\u04e17\x8a\x03\x00F\xc8\xccw_\x04\x00\x00\u07d4n\x1a\x04l\xaf[JW\xf4\xfdK\xc1sb!&\xb4\xe2\xfd\x86\x89a\t=|,m8\x00\x00\u07d4n\x1e\xa4\xb1\x83\xe2R\u027bwg\xa0\x06\u05346\x96\u02ca\xe9\x89\x0f\xf3x<\x85\xee\u0400\x00\u07d4n%[p\n\xe7\x13\x8aK\xac\xf2(\x88\xa9\xe2\xc0\n(^\xec\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4n'\n\xd5)\xf1\xf0\xb8\xd9\xcbm$'\xec\x1b~-
\xc6Jt\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4n.\xab\x85\u0709\xfe)\xdc\n\xa1\x852G\u06b4:R=V\x89\x04V9\x18$O@\x00\x00\u07d4n:Q\xdbt=3M/\xe8\x82$\xb5\xfe|\x00\x8e\x80\xe6$\x89\x05\xbf\v\xa6cOh\x00\x00\u07d4nL*\xb7\xdb\x02i9\xdb\u04fch8J\xf6`\xa6\x18\x16\xb2\x89\t\r\x97/22<\x00\x00\u07d4nM.9\u0203f)\u5d07\xb1\x91\x8af\x9a\xeb\u07556\x8965\u026d\xc5\u07a0\x00\x00\u07d4n\\-
\x9b\x1cTj\x86\xee\xfd]\nQ
\xc9\xe4\xe70\x19\x0e\x89\n\xd2\x01\xa6yO\xf8\x00\x00\u07d4n`\xae\u19cf\x8e\u068bBLs\xe3S5J\xe6|0B\x89\xbd5\xa4\x8d\x99\x19\xe6\x00\x00\u07d4nd\xe6\x12\x9f\"N7\x8c\x0ensj~z\x06\xc2\x11\xe9\xec\x8965\u026d\xc5\u07a0\x00\x00\u07d4nm[\xbb\xb9\x05;\x89\xd7D\xa2s\x16\u00a7\xb8\xc0\x9bT}\x891Rq\n\x02>m\x80\x00\u07d4nr\xb2\xa1\x18j\x8e)\x16T;\x1c\xb3jh\x87\x0e\xa5\u0457\x89\n\x15D\xbe\x87\x9e\xa8\x00\x00\u07d4nv\x1e\xaa\x0f4_w{TA\xb7:\x0f\xa5\xb5k\x85\xf2-\x89lk\x93[\x8b\xbd@\x00\x00\u07d4ny\xed\u0504[\anL\u060d\x18\x8bnC-
\xd9?5\xaa\x893\xc5I\x901r\f\x00\x00\u07d4n\x82\x12\xb7\"\xaf\xd4\b\xa7\xa7>\xd3\xe29^\xe6EJ\x030\x89\b\x9e\x91y\x94\xf7\x1c\x00\x00\u07d4n\x84\x87m\xbb\x95\xc4\vfV\xe4+\xa9\xae\xa0\x8a\x99;T\u0709;\xbc`\xe3\xb6\u02fe\x00\x00\u07d4n\x84\xc2\xfd\x18\xd8\tW\x14\xa9h\x17\x18\x9c

\xa2\x1c\xcab\xba\xb1\x89\x12{lp&!\u0340\x00\u07d4n\x86m\x03-
@Z\xbd\xd6\\\xf6QA\x1d\x807\x96\xc2#\x11\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94n\x89\x9eY\xa9
\xb4\x1a\xb7\xeaA\xdfu\x17\x86\x0f*\xcbY\xf4\xfd\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4n\x89\
xc5\x1e\xa6\xde\x13\xe0l\xdct\x8bg\xc4A\x0f\u9f2b\x03\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d
4n\x8a&h\x9fz/\xde\xfd\x00\x9c\xba\xaaS\x10%4P\u06ba\x89o!7\x17\xba\xd8\xd3\x00\x00\u07d4n
\x96\xfa\xed\xa3\x05C\x02\xc4_X\xf1a2L\x99\xa3\xee\xbbb\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u
07d4n\xb0\xa5\xa9\xae\x96\xd2,\xf0\x1d\x8f\xd6H;\x9f8\xf0\x8c,\x8b\x89\xd8\xd7&\xb7\x17z\x80\x
00\x00\u07d4n\xb3\x81\x96\x17@@X&\x8f\f<\xff5\x96\xbf\xe9\x14\x8c\x1c\x89Z\x87\xe7\xd7\xf5\
xf6X\x00\x00\xe0\x94n\xb5W\x8ak\xb7\xc3!S\x19[\r\x80
\xa6\x91HR\xc0Y\x8a\x8b\u00ab\xf4\x02!\xf4\x80\x00\x00\u07d4n\xbb^iW\xaa\x82\x1e\xf6Y\xb6\x
01\x8a9:PL\xaeDP\x89lk\x93[\x8b\xbd@\x00\x00\u07d4n\xbc\xf9\x95\u007f_\xc5\u916d\xd4u\";\x0
4\xb8\xc1Jz\xed\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4n\xc3e\x95q\xb1\x1f\x88\x9d\xd49\xbc\x
d4\xd6u\x10\xa2[\xe5~\x89\x06\xaa\xf7\xc8Qm\f\x00\x00\u07d4n\u021b9\xf9\xf5'jU>\x8d\xa3\x0en
\xc1z\xa4~\xef\u01c9\x18BO_\v\x1bN\x00\x00\u07d4n\xc9m\x13\xbd\xb2M\u01e5W)?\x02\x9e\x0
2\xddt\xb9zU\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4n\xca\xef\xa6\xfc>\xe54bm\xb0,o\x85\xa0
\u00d5W\x1ew\x89
\x86\xac5\x10R`\x00\x00\u07d4n\u04a1+\x02\xf8\u0188\u01f5\u04e6\xea\x14\xd66\x87\u06b3\xb
6\x89lk\x93[\x8b\xbd@\x00\x00\u07d4n\u0604E\x9f\x80\x9d\xfa\x10\x16\xe7p\xed\xaf>\x9f\xefF\xf
a0\x89\xb8R\xd6x
\x93\xf1\x00\x00\xe0\x94n\xdf\u007fR\x83r\\\x95>\xe6C\x17\xf6a\x88\xaf\x11\x84\xb03\x8a\x01\xb
4d1\x1dE\xa6\x88\x00\x00\u07d4n\xe8\xaa\xd7\xe0\xa0e\u0605-
|;\x9an_\xdcK\xf5\f\x00\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4n\xef\u0705\x0e\x87\xb7\x15\xc
7'\x91w<\x03\x16\xc3U\x9bX\xa4\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4n\xf9\xe8\u0276!}Vv\
x9a\xf9}\xbb\x1c\x8e\x1b\x8b\xe7\x99\u0489\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\u07d4n\xfb\xa8\xf
b*\u0176s\a)\xa9r\xec\"D&\xa2\x87\u00ed\x89\x0fY\x85\xfb\xcb\xe1h\x00\x00\xe0\x94n\xfd\x90\xb
55\xe0\v\xbd\x88\x9f\xda~\x9c1\x84\xf8y\xa1Q\u06ca\x02#\x85\xa8'\xe8\x15P\x00\x00\u07d4o\x0
5\x16f\xcbO{\u04b1\x90r!\xb8)\xb5U\u05e3\xdbt\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4o\x0e\xd
d#\xbc\xd8_`\x15\xf9(\x9c(\x84\x1f\xe0L\x83\xef\xeb\x89\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\u07d4
o\x13zq\xa6\xf1\x97\xdf,\xbb\xf0\x10\u073d<DN\xf5\xc9%\x89lk\x93[\x8b\xbd@\x00\x00\u07d4o\x
17`e\xe8\x8e<o\xe6&&}\x18\xa0\x88\xaa\xa4\u06c0\xbc\x89\xbe\xd1\xd0&=\x9f\x00\x00\x00\u07d
4o\x18\xecv~2\x05\b\x19_\x13tP\x0e?.\x12V\x89\xff\x8965\u026d\xc5\u07a0\x00\x00\u07d4o\x1fl\
a\xb8\xf6\x1f\fQV\x8di(\x06\xb3\x82\xf5\x03$\xf5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4o$\u026f+v4
\x80Q]\x1b\tQ\xbbw\xa5@\xf1\xe3\xf9\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4o%v\xdaM\u20fb\xe8\
xe3\xeei\xdd\xd6n^q\x1d\xb3\xf5\x89DY\x1dg\xfe\u0300\x00\x00\u07d4o)\xbb7[\xe5\xed4\ud65b\x
b80\xee)W\xdd\xe7m\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4o*1\x90\x0e$\x03\x95\xb1\x9f\x15\x
9c\x1d\x00\xdf\xe4\u0618\xeb\u07c9lf\x06E\xaaG\x18\x00\x00\u07d4o*B\xe6\xe03\xd0\x10a\x13\
x19)\xf7\xa6\xee\x158\x02\x1eR\x89lk\x93[\x8b\xbd@\x00\x00\u07d4o9\xcc7\u02aa-
\u071ba\x0fa1\xe0a\x9f\xaew*<\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4oD\xca\t\xf0\u01a8)
L\xbdQ\x9c\xdcYJ\xd4,gW\x9f\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4oP\x92\x97w\x82L)
\x1aI\xc4m\xc8T\xf3y\xa6\xbe\xa0\x80\x89\x13\x84\x00\xec\xa3d\xa0\x00\x00\xe0\x94ol\xf2\x06I\x
a9\xe9s\x17z\xc6}\xba\xde\xe4\xeb\xe5\u01fd\u068a\x01\x13c)}\x01\xa8`\x00\x00\u07d4oy\x1d5\x
9b\xc3Sj1]c\x82\xb8\x83\x11\xaf\x8e\xd6\xdaG\x89\x04\xfc\xc1\xa8\x90'\xf0\x00\x00\u07d4oyM\x

bd\xf6#\u06a6\xe0\xd0\at\xadibs|\x92\x1e\xa4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4oz\u0181\xd4^
A\x8f\u038b:\x1d\xb5\xbc;\xe6\xf0l\x98l\x89lk\x93[\x8b\xbd@\x00\x00\u07d4o\x81\xf3\xab\xb1\xf9
3\xb1\xdf9k\x8e\x9c\xc7#\xa8\x9b|\x98\x06\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\u07d4o\x8f\r\x15\u0316\xfb\u007f\xe9O\x10e\xbci@\xf8\xd1)W\xb2\x
8965\u026d\xc5\u07a0\x00\x00\u07d4o\x92\xd6\xe4T\x8cx\x99e\t\xeehK.\u26e3\xc52\xb4\x8965\
u026d\xc5\u07a0\x00\x00\xe0\x94o\xa6\r\xf8\x18\xa5Dd\x18\xb1\xbb\xd6(&\u0e42^\x13\x18\x8a\x
02\u02d2\u030fg\x14@\x00\x00\u07d4o\xa68\x8d@+0\xaf\xe5\x994\u00f9\xe1=\x11\x86G`\x18\x
89$R\x1e*0\x17\xb8\x00\x00\u07d4o\xa7
\x15\xfaxin\xfd\x9a\x86\x17O\u007f\x1f!\x01\x92\x86\xb1\x89Hz\x9a0E9D\x00\x00\u07d4o\xc2^~\
x00\xcaO`\xa9\xfeo(\xd1\xfd\xe3T.-
\x10y\x89*\xef5;\xcd\xdd`\x00\x00\u07d4o\xc56b7\x1d\xcaX{Y\x85\r\xe7\x86\x06\xe25\x9d\xf3\x83
\x89\t\xc2\x00vQ\xb2P\x00\x00\xe0\x94o\xcc,s+\u0753J\xf6\xcc\xd1hF\xfb&\uf272\xaa\x9b\x8a\x0
2\x1e+\x1dB&\x1dI\x00\x00\u07d4o\xd4\xe0\xf3\xf3+\xeem7g\xfd\xbc\x9d5:m:\xabx\x99\x89%\xb0
d\xa8u\xea\x94\x00\x00\xe0\x94o\xd9G\u0567;\x17P\b\xaen\xe8\"\x81c\xda(\x9b\x16}\x8a\x06ZM\
xa2]0\x16\xc0\x00\x00\u07d4o\u064eV=\x12\xce\x0f\xd6\x0fO\x1f\x85\n\u35a9\x82<\x02\x89D[\xe
3\xf2\uf1d4\x00\x00\xe0\x94o\u077d\x9b\xcaf\xe2\x87e\xc2\x16,\x843T\x8c\x10R\xed\x11\x8a\x11
\x84B\x9b\x82\xa8\x18\x80\x00\x00\u07d4o\xf5\xd3a\xb5*\u0436\x8b\x15\x88`~\xc3\x04\xaeVe\xf
c\x98\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4o\xf6\u0310\xd6I\xdeN\x96\xcf\xfe\xe1\az[0*\x84\x8d\
u02c9\x01\x8c\xe7\x9cx\x80,\x00\x00\u07d4o\xfe\\\xf8,\xc9\xea^6\xca\xd7\u0097L\xe7$\x9f7I\xe6\
x89i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94p\x05\xa7r(+\x1fb\xaf\xdac\xf8\x9b]\u01abd\xc8L\xb9\x8a\
x03\xcf\xc8.7\xe9\xa7@\x00\x00\u07d4p\a\x11\xe3\x11\xbb\x94sU\xf7U\xb5y%\f\xa7\xfdvZ>\x89a
\t=|,m8\x00\x00\u07d4p\x10\xbe-
\xf5{\u042b\x9a\xe8\x19l\xd5\n\xb0\xc5!\xab\xa9\xf9\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4p#\xc7\t
V\xe0J\x92\xd7\x00%\xaa\u0497\xb59\xaf5Xi\x89lk\x93[\x8b\xbd@\x00\x00\u07d4p%\x96]+\x88\x
da\x19}DY\xbe=\xc98cD\xcc\x1f1\x89l\xb7\xe7Hg\xd5\xe6\x00\x00\u07d4p(\x02\xf3m\x00%\x0f\xa
bS\xad\xbc\u0596\xf0\x17oc\x8aI\x89lk\x93[\x8b\xbd@\x00\x00\u07d4pH\x19\xd2\xe4Mn\xd1\xda
%\xbf\u0384\u011f\u0322V\x13\xe5\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4pJn\xb4\x1b\xa3O\
x13\xad\xdd\xe7\xd2\xdb}\xf0l\x15\u01e2!\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4pJ\xb1\x15\r^\x1
0\xf5\xe3l\x95\b\xf0\xbfpe\x0f\x02\x8dK\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4pJ\xe2\x1dv-
n\x1d\xde(\xc25\xd11\x04Yr6\xdb\x1a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4pM$<)x\xe4l,\x86\xad\x
be\xcd$n;)_\xf63\x89m\x12\x1b\xeb\xf7\x95\xf0\x00\x00\u07d4pM]\xe4\x84m9\xb5<\xd2\x1d\x1cI\
xf0\x96\xdb\\\x19\xba)\x89\b=lz\xabc`\x00\x00\u07d4p]\xdd85T\x82\xb8\xc7\u04f5\x15\xbd\xa1P\r\
xd7\u05e8\x17\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94pan(\x92\xfa&\x97\x05\xb2\x04k\x8f\x
e3\xe7/\xa5X\x16\u04ca\x04<3\xc1\x93ud\x80\x00\x00\u07d4pg\x0f\xbb\x05\xd30\x14DK\x8d\x1e
\x8ew\x00%\x8b\x8c\xaam\x89lk\x93[\x8b\xbd@\x00\x00\u07d4p\x81\xfak\xaa\xd6\u03f7\xf5\x1b,\
xca\x16\xfb\x89p\x99\x1ad\xba\x89\f\xae\xc0\x05\xf6\xc0\xf6\x80\x00\xe0\x94p\x85\xae~~M\x93!\
x97\xb5\u01c5\x8c\x00\xa3gF&\xb7\xa5\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4p\x86\xb4\xb
d\xe3\xe3]J\xeb$\xb8%\xf1\xa2\x15\xf9\x9d\x85\xf7E\x89lh\xcc\u041b\x02,\x00\x00\u07d4p\x8a*\x
f4%\u03b0\x1e\x87\xff\xc1\xbeT\xc0\xf52\xb2\x0e\xac\u0589\aE\u0503\xb1\xf5\xa1\x80\x00\u07d
4p\x8e\xa7\a\xba\xe45\u007f\x1e\xbe\xa9Y\u00e2P\xac\u05aa!\xb3\x89\x1b\x1a\xe4\xd6\xe2\xefP
\x00\x00\u0794p\x8f\xa1\x1f\xe3=\x85\xad\x1b\xef\u02ee8\x18\xac\xb7\x1fj}~\x88\xfc\x93c\x92\x8
0\x1c\x00\x00\u07d4p\x9101\x16\xd5\xf28\x9b##\x8bMej\x85\x96\u0644\u04c9;N~\x80\xaaX3\x00

\x00\u07d4p\x99\xd1/n\xc6V\x89\x9b\x04\x9avW\x06]b\x99h\x92\u0209\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4p\x9f\xe9\xd2\xc1\xf1\xceB
|\x95\x85\x04J`\x89\x9f5\x94/\x89lk\x93[\x8b\xbd@\x00\x00\u07d4p\xa05I\xaaah\xe9~\x88\xa5\b3\nZ\v\xeatq\x1a\x89Hz\x9a0E9D\x00\x00\u07d4p\xa4\x06}D\x8c\xc2]\xc8\xe7\x0ee\x1c\xea|\xf8N\x92\x10\x9e\x89\t\x8a}\x9b\x83\x14\xc0\x00\x00\u07d4p\xab4\xbc\x17\xb6o\x9c;c\xf1Q'O*r|S\x92c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4p\xc2\x13H\x8a\x02\f<\xfb9\x01N\xf5\xbad\x04rK\u02a3\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4p\xd2^\xd2\u022d\xa5\x9c\b\x8c\xf7\r\xd2+\xf2\u06d3\xac\xc1\x8a\x899GEE\u4b7c\x00\x00\u07d4p\xe5\xe9\xdas_\xf0w$\x9d\u02da\xaf=\xb2\xa4\x8d\x94\x98\xc0\x8965\u026d\xc5\u07a0\x00\x00\u07d4p\xfe\xe0\x8b\x00\xc6\xc2\xc0J<b\\\x1f\xf7|\xaf\x1c2\xdf\x01\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4q\x01\xbdy\x9eA\x1c\xde\x14\xbd\xfa\xc2[\x06z\u0210\xea\xb8\xe8\x89N\x9b\x8a\xaeH\xdeG\x00\x00\u07d4q\t\xdd\x01\x1d\x15\xf3\x12-
\x9d:'X\x8c\x10\xd7wDP\x8b\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94q\v\x02t\xd7\x12\xc7~\b\xa5p}o>p\xc0\xce=\x92\u03ca\x01Z\xf1\u05cbX\xc4\x00\x00\x00\u0794q\v\xe8\xfd^)\x18F\x8b\u2abe\xa8\r\x82\x845\u05d6\x12\x88\xf4?\xc2\xc0N\xe0\x00\x00\u07d4q\x13]\x8f\x05\x96<\x90ZJ\a\x92)\t#Z\x89jR\ua262\xa1]\tQ\x9b\xe0\x00\x00\u07d4q\x1e\xcfw\xd7\x1b=\x0e\xa9\\\xe4u\x8a\xfe\u0379\xc11\a\x9d\x89)3\x1eeX\xf0\xe0\x00\x00\u07d4q!?\xca14\x04
N\u02e8q\x97t\x1a\xa9\xdf\xe9c8\x89\x03@\xaa\xd2\x1b;p\x00\x00\xe0\x94q+vQ\x02\x14\xdcb\x0fl:\x1d\u049a\xa2+\xf6\xd2\x14\xfb\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4q/\xf77\n\x13\xed6\ts\xfe\u071f\xf5\xd2\xc9:P^\x9e\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4q3\x84:x\xd99\u019dD\x86\xe1\x0e\xbc{`*4\x9f\xf7\x89\x11\xd5\xca\xcc\xe2\x1f\x84\x00\x00\u07d4qH\xae\xf32a\xd8\x03\x1f\xac?q\x82\xff5\x92\x8d\xafT\u0649\xdeB\xee\x15D\u0750\x00\x00\u07d4qcu\x8c\xbblLR^\x04\x14\xa4\n\x04\x9d\xcc\xcc\xe9\x19\xbb\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4qh\xb3\xbb\x8c\x16s!\u067d\xb0#\xa6\xe9\xfd\x11\xaf\u026f\u0649a\t=|,m8\x00\x00\u07d4qirN\xe7\"q\xc54\xca\xd6B\x0f\xb0N\xe6D\u02c6\xfe\x89\x16<+@\u06e5R\x00\x00\u07d4qj\xd3\xc3:\x9b\x9a\n\x18\x96sW\x96\x9b\x94\xee}*\xbc\x10\x89\x1a!\x17\xfeA*H\x00\x00\xe0\x94qk\xa0\x1e\xad*\x91'\x065\xf9_%\xbf\xaf-\xd6\x10\xca#\x8a\ty\xe7\x01
V\xaax\x00\x00\u07d4qmP\u0320\x1e\x93\x85\x00\xe6B\x1c\xc0p\xc3P|g\u04c7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4qv,cg\x8c\x18\xd1\xc67\x8c\xe0h\xe6f8\x13\x15\x14~\x89lk\x93[\x8b\xbd@\x00\x00\u07d4qxL\x10Q\x17\xc1\xf6\x895y\u007f\xe1Y\xab\xc7NC\xd1j\x89l\x81\u01f3\x11\x95\xe0\x00\x00\xe0\x94qyro\\q\xae\x1bm\x16\xa6\x84(\x17Nk4\xb26F\x8a\x01\x8e\xa2P\t|\xba\xf6\x00\x00\xe0\x94q|\xf9\xbe\xab680\x8d\xed~\x19^\f\x86\x13-
\x16?\xed\x8a\x032n\xe6\xf8e\xf4\"\x00\x00\u07d4q\x80\xb8>\xe5WC\x17\xf2\x1c\x80r\xb1\x91\u0615\xd4aS\u00c9\x18\xef\xc8J\xd0\u01f0\x00\x00\u07d4q\x94kq\x17\xfc\x91^\xd1\a8_B\u065d\xda\xc62I\u0089lk\x93[\x8b\xbd@\x00\x00\xe0\x94q\x9e\x89\x1f\xbc\xc0\xa3>\x19\xc1-\xc0\xf09\xca\x05\xb8\x01\u07ca\x01OU8F:\x1bT\x00\x00\u07d4q\xc7#\n\x1d5\xbd\u0581\x9e\u0539\xa8\x8e\x94\xa0\xeb\a\x86\u0749\uc80b5=$\x14\x00\x00\u07d4q\xd2\xccm\x02W\x8ce\xf7<W^v\u038f\xbc\xfa\xdc\xf3V\x89\x03\xec\xc0xh\x8aH\x00\x00\u07d4q\xd9INP\xc5\xddY\u0159\u06e3\x81\v\xa1u^e7\xf0\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4q\xe3\x8f\xf5E\xf3\x0f\xe1L\xa8c\xd4\xf5)\u007f\u050cs\xa5\u0389\xc2\x12z\xf8X\xdap\x00\x00\u07d4q\xea[\x11\xad\x8d)\xb1\xa4\xcbg\xbfX\xcal\x9f\x9c3\x8c\x16\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4q\xec:\xec?\x8f\x92!\xf9\x14\x9f\xed\xe0i\x03\xa0\xf9\xa22\xf2\x89\n\u05ce\xbcZ\xc6

\x00\x00\u07d4q\xf2\xcd\u0470F\xe2\xda/\xbbZ&r4\"\xb82^%\xa3\x89\x05k9Bc\xa4\f\x00\x00\u07d4q\xfa\"\xccm3
k}p\x1a\x16:\r\xab1\xaeM1\u0589WG=\x05\u06ba\xe8\x00\x00\u07d4r\x01\xd1\xc0i
\xcd9z\u8b46\x9b\u0366\xe4\u007f\xfb\x1bZ\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4r\a*\x0e\xf1\xcf\xf3\xd5g\xcd\xd2`\xe7\b\xdd\xc1\x1c\xbc\x9a1\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94r\tO9Q\xff\xc9w\x1d\xce\xd2:\xda\b\v\xca\xf9\xc7\u0327\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94r\t\x94\xdb\xe5j:\x95\x92\x97t\xe2\x0e\x1f\xe5%\xcf7\x04\xe4\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4r\x0ek\"\xbfC\tf\xfa2\xb6\xac\xb9\xa5\x06\xee\xbff,a\x89\b=lz\xabc`\x00\x00\xe0\x94r\x11X\xbeWb\xb1\x19\u031b
5\xe8\x8e\xe4\xeex\U0009b08a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4r\x1f\x9d\x17\xe5\xa0\xe7B\x05\x94z\xeb\x9b\u01a7\x93\x89a\x03\x8f\x89\x02\xd0A\xd7\x05\xa2\xc6\x00\x00\u07d4r\"\xfe\xc7q\x17\x81\xd2n\xaaN\x84\x85\xf7\xaa?\xacD$\x83\x89\x18\xb8Ep\x02*
\x00\x00\u07d4r9=7\xb4Q\xef\xfb\x9e\x1f\xf3\xb8U'\x12\xe2\xa9p\xd8\u00895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4r=\x8b\xaa%Q\u04ad\xdcC\xc2\x1bE\xe8\xafL\xa2\xbf\xb2\u0089_h\xe8\x13\x1e\u03c0\x00\x00\xe0\x94r@#\x00\xe8\x1d\x14l.dN+\xbd\xa1\xda\x16<\xa3\xfbV\x8a\x01{x\x83\xc0i\x16`\x00\x00\xe0\x94rH\v\xed\xe8\x1a\xd9d#\xf2\"\x8b\\a\xbeD\xfbR1\x00\x8a\x01Z\xf1\u05cbX\xc4\x00\x00\x00\u07d4rL\xe8X\x85~\xc5H\x1c\x86\xbd\x90n\x83\xa0H\x82\xe5\x82\x1d\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94rj\x14\xc9\x0e?\x84\x14Lv\\\xff\xac\xba>\r\xf1\x1bH\xbe\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4r\x83\xcdFu\xdaX\u0116UaQ\xda\xfd\x80\xc7\xf9\x95\xd3\x18\x89)3\x1eeX\xf0\xe0\x00\x00\u07d4r\x86\xe8\x9c\xd9\u078fz\x8a\x00\xc8o\xfd\xb59\x92\u0752Q\u0449i*\xe8\x89p\x81\xd0\x00\x00\u07d4r\x8f\x9a\xb0\x80\x15}\xb3\a1V\xdb\xca\x1a\x16\x9e\xf3\x17\x94\a\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4r\x94\xc9\x18\xb1\xae\xfbM%\x92~\xf9\u05d9\xe7\x1f\x93\xa2\x8e\x85\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94r\x94\uc763\x10\xbckK\xbd\xf5C\xb0\xefE\xab\xfc>\x1bM\x8a\x04\xa8\x9fT\xef\x01!\xc0\x00\x00\u07d4r\x9a\xadF'tNS\xf5\xd6c\t\xaatD\x8b:\xcd\xf4o\x89lk\x93[\x8b\xbd@\x00\x00\u07d4r\xa2\xfc\x86u\xfe\xb9r\xfaA\xb5\r\xff\u06fa\xe7\xfa*\u07f7\x89\x9a\xb4\xfcg\xb5(\xc8\x00\x00\u07d4r\xa8&\b&)G&\xa7[\xf3\x9c\u066a\x9e\a\xa3\xea\x14\u0349lk\x93[\x8b\xbd@\x00\x00\u07d4r\xb0Yb\xfb*\u0549\xd6Z\xd1j\"U\x9e\xba\x14X\xf3\x87\x89\a?u\u0460\x85\xba\x00\x00\u07d4r\xb5c?\xe4w\xfeT.t/\xac\xfdi\f\x13xT\xf2\x16\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4r\xb7\xa0=\xda\x14\u029cf\x1a\x1dF\x9f\xd376\xf6s\xc8\xe8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4r\xb9\x04D\x0e\x90\xe7
\u05ac\x1c*\u05dc2\x1d\xcc\x1c\x1a\x86\x89T\x06\x923\xbf\u007fx\x00\x00\xe0\x94r\xb9\nM\xc0\x97#\x94\x92\u0179w}\xcd\x1eR\xba+\xe2\u008a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4r\xbb'\u02d9\xf3\xe2\xc2\u03d0\xa9\x8fp}0\xe4\xa2\x01\xa0q\x89X\xe7\x92n\xe8X\xa0\x00\x00\xe0\x94r\xc0\x83\xbe\xad\xbd\xc2'\xc5\xfbC\x88\x15\x97\xe3.\x83\xc2`V\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4r\xcd\x04\x8a\x11\x05tH)\x83I-
\xfb\x1b\xd2yB\xa6\x96\xba\x89lk\x93[\x8b\xbd@\x00\x00\u07d4r\xd0=M\xfa\xb3P\f\xf8\x9b\x86\x86o\x15\xd4R\x8e\x14\xa1\x95\x89\xf3K\x82\xfd\x8e\x91
\x00\x00\u07d4r\u06bb[n\ud799\xbe\x91X\x88\xf6V\x80V8\x16\b\xf8\x89\vL\x96\xc5,\xb4\xfe\x80\x00\u07d4r\xfbI\u009d#\xa1\x89P\u0132\xdc\r\xdfA\x0fS-
oS\x89lk\x93[\x8b\xbd@\x00\x00\u07d4r\xfe\xaf\x12EyR9Td[\u007f\xaf\xff\x03x\xd1\xc8$.\x8965\u026d\xc5\u07a0\x00\x00\u07d4s\x01\xdcL\xf2mq\x86\xf2\xa1\x1b\xf8\xb0\x8b\xf2)F?d\xa3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4s\x04G\xf9|\xe9\xb2_\"\xba\x1a\xfb6\xdf'\xf9Xk\ub6c9,s\xc97t,P\x00\

x00\u07d4s\x06\xde\x0e(\x8bV\xcf\u07d8~\xf0\xd3\xcc)f\a\x93\xf6\u0749\x1b\x8a\xbf\xb6.\xc8\xf6\
x00\x00\xe0\x94s\r\x87c\u01a4\xfd\x82J\xb8\xb8Y\x16\x1e\xf7\xe3\xa9j\x12\x00\x8a\x04<3\xc1\x9
3ud\x80\x00\x00\u07d4s\x12\x81sH\x95(\x01.v\xb4\x1a^(\u018b\xa4\xe3\xa9\u050965\u026d\xc5\
u07a0\x00\x00\u07d4s\x13F\x12\bETUFTE\xa4Y\xb0l7s\xb0\xeb0\x89lk\x93[\x8b\xbd@\x00\x00\u
07d4s/\xea\xd6\x0f{\xfd\u05a9\xde\u0101%\xe3s]\xb1\xb6eO\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\
u07d4sB#\xd2\u007f\xf2>Y\x06\xca\xed\"YW\x01\xbb4\x83\f\xa1\x89lk\x93[\x8b\xbd@\x00\x00\u0
7d4sG>r\x11Q\x10\xd0\xc3\xf1\x17\b\xf8nw\xbe+\xb0\x98<\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u
07d4sRXm\x02\x1a\xd0\xcfw\xe0\xe9(@JY\xf3t\xffE\x82\x89\xb8Pz\x82\a(
\x00\x00\u07d4sU\v\xebs+\xa9\u076f\xdaz\xe4\x06\xe1\x8f\u007f\xeb\x0f\x8b\xb2\x89\x97\xc9\xc
eL\xf6\xd5\xc0\x00\x00\u07d4s[\x97\xf2\xfc\x1b\xd2K\x12\an\xfa\xf3\xd1(\x80s\xd2\f\x8c\x89\x01\x
15\x8eF\t\x13\xd0\x00\x00\u07d4s^2\x86f\xedV7\x14+3\x06\xb7|\xccT`\xe7,=\x89j\xb8\xf3xy\u025
1\x00\x00\u07d4sc\u0350\xfb\xab[\xb8\u011a\xc2\x0f\xc6,9\x8f\xe6\xfbtL\x89lk\x93[\x8b\xbd@\x0
0\x00\u07d4skDP=\xd2\xf6\xddTi\xffL[-
\xb8\xeaO\xece\u0409\x11\x04\xeeu\x9f!\xe3\x00\x00\xe0\x94sk\xf1@,\x83\x80\x0f\x89>X1\x92X*
\x13N\xb52\xe9\x8a\x02\x1e\x19\u0493\xc0\x1f&\x00\x00\xe0\x94s\x8c\xa9M\xb7\u038b\xe1\xc3\
x05l\u0598\x8e\xb3v5\x9f3S\x8a\x05f[\x96\xcf5\xac\xf0\x00\x00\u07d4s\x91K\"\xfc/\x13\x15\x84$}\
x82\xbeO\ucfd7\x8a\u053a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4s\x93'\t\xa9\u007f\x02\u024eQ\xb
0\x911(e\x12#\x85\xae\x8e\x89M\x85<\x8f\x89\b\x98\x00\x00\u07d4s\x93\xcb\xe7\xf9\xba!e\xe5\x
a7U5\x00\xb6\xe7]\xa3\xc3:\xbf\x89\x05k\xc7^-
c\x10\x00\x00\u07d4s\xb4\u0519\xde?8\xbf5\xaa\xf7i\xa6\xe3\x18\xbcm\x126\x92\x89lk\x93[\x8b\
xbd@\x00\x00\xe0\x94s\xbe\xddo\xda{\xa3'!\x85\b{cQ\xfc\x13=HN7\x8a\x01\x12&\xbf\x9d\xceYx\
x00\x00\u07d4s\xbf\xe7q\x0f1\u02b9l\xb7\xa2`O\xbfR9\xce\xe7\x90\x15\x89lk\x93[\x8b\xbd@\x00
\x00\xe0\x94s\u03c0\xae\x96\x88\xe1X\x0eh\xe7\x82\xcd\b\x11\xf7\xaaIM,\x8a\x01\xa4\xab\xa2%
\xc2\a@\x00\x00\xe0\x94s\xd7&\x9f\xf0l\x9f\xfd3uL\xe5\x88\xf7J\x96j\xbb\xbb\xba\x8a\x01e\xc9fG
\xb3\x8a
\x00\x00\u07d4s\xd8\xfe\xe3\u02c6M\xce\"\xbb&\u029c/\bm^\x95\xe6;\x8965\u026d\xc5\u07a0\x0
0\x00\u07d4s\xdf<>yU\xf4\xf2\xd8Y\x83\x1b\xe3\x80\x00\xb1\ak8\x84\x89j\xcb=\xf2~\x1f\x88\x00\
x00\u07d4s\u48b6\f\U0010e2ef+w~\x17Z[\x1eM\f-\x8f\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94t\n\xf1\xee\xfd3e\u05cb\xa7\xb1,\xb1\xa6s\xe0j\arF\x8a\x04+\xf0kx\xed;P\x
00\x00\xe0\x94t\v\xfdR\xe0\x16g\xa3A\x9b\x02\x9a\x1b\x8eEWj\x86\xa2\u06ca\x03\x8e\xba\xd5\x
cd\xc9\x02\x80\x00\x00\u07d4t\x0fd\x16\x14w\x9d\u03e8\x8e\xd1\xd4%\xd6\r\xb4*\x06\f\xa6\x896
\"\xc6v\b\x10W\x00\x00\u07d4t\x12\u027c0\xb4\xdfC\x9f\x021\x00\xe69$\x06j\xfdS\xaf\x89\x1b\x1
a\xe4\xd6\xe2\xefP\x00\x00\u07d4t\x16\x93\xc3\x03vP\x85\x13\b
\xcc+c\xe9\xfa\x92\x13\x1b\x89A\rXj
\xa4\xc0\x00\x00\u07d4t!\xce[\xe3\x81s\x8d\u0703\xf0&!\x97O\xf0hly\xb8\x89Xx\x8c\xb9K\x1d\x8
0\x00\x00\u07d4t1j\xdf%7\x8c\x10\xf5v\u0574\x1aoG\xfa\x98\xfc\xe3=\x89\x128\x13\x1e\\z\xd5\x0
0\x00\u07d4t6Q\xb5^\xf8B\x9d\xf5\f\xf8\x198\xc2P\x8d\xe5\u0207\x0f\x89lk\x93[\x8b\xbd@\x00\x
00\u07d4t=\xe5\x00&\xcag\xc9M\xf5O\x06b`\xe1\xd1J\xcc\x11\xac\x89lk\x93[\x8b\xbd@\x00\x00\
u07d4tE /\ft)z\x00N\xb3rj\xa6\xa8-
\xd7\xc0/\xa1\x89lk\x93[\x8b\xbd@\x00\x00\u07d4tK\x03\xbb\xa8X*\xe5I\x8e-\xc2-
\x19\x94\x94g\xabS\xfc\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4tL\fw\xba\u007f#i
\xd1\xe44\xde]\xa3>H\xeb\xf0,\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4tP\xff\u007f\x99\xea\xa9\x11

bu\u07ach\xe4(\xdf[\xbc\u0639\x89lk\x93[\x8b\xbd@\x00\x00\u07d4tV\u0172\xc5Cn>W\x10\b\x93
?\x18\x05\xcc\xfe4\xe9\xec\x8965\u026d\xc5\u07a0\x00\x00\u07d4tZ\u04eb\xc6\xee\xeb$qh\x9bS
\x9ex\x9c\xe2\xb8&\x83\x06\x89=A\x94\xbe\xa0\x11\x92\x80\x00\xe0\x94tZ\xec\xba\xf9\xbb9\xb7
Jg\xea\x1c\xe6#\xde6\x84\x81\xba\xa6\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4t\\\xcf-
\x81\x9e\u06fd\u07a8\x11{\\I\xed<*\x06n\x93\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4tb\u021c\
xaa\x9d\x8dx\x91\xb2T]\xef!otd\u057b!\x89\x05\xea\xedT\xa2\x8b1\x00\x00\u07d4td\x8c\xaa\xc7
H\xdd\x13\\\xd9\x1e\xa1L(\xe1\xbdM\u007f\xf6\xae\x89\xa8\r$g~\xfe\xf0\x00\x00\xe0\x94tq\xf7.\xe
b0\x06$\xeb(.\xabM\x03r<d\x9b\x1bX\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94tz\xbc\x96I\x05
m9&\x04M(\u00ed\t\xed\x17\xb6}p\x8a\x01\x0f\r\xba\xe6\x10\tR\x80\x00\u07d4t\u007f\xf7\x94;q\x
dcM\u0371f\x80x\xf8=\xd7\xccE
\u0089\x03@\xaa\xd2\x1b;p\x00\x00\xe0\x94t\x80\xdeb%O+\xa8+W\x82\x19\xc0{\xa5\xbeC\r\xc3\
u02ca\x01}\xa3\xa0L{>\x00\x00\x00\xe0\x94t\x84\xd2k\xec\xc1\xee\xa8\xc61^\xc3\xee\nE\x01\x17
\u0706\xa0\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4t\x86:\xce\xc7]\x03\xd5>\x86\x0ed\x00/,\x16^
S\x83w\x8965\u026d\xc5\u07a0\x00\x00\u07d4t\x89\u030a\xbeu\u0364\xef\r\x01\xce\xf2`^G\xed\
xa6z\xb1\x89\a?u\u0460\x85\xba\x00\x00\u07d4t\x8c(^\xf1#?\xe4\xd3\x1c\x8f\xb17\x833r\x1c\x12
\xe2z\x89lk\x93[\x8b\xbd@\x00\x00\u07d4t\x90\x87\xac\x0fZ\x97\xc6\xfa\xd0!S\x8b\xf1\xd6\u0361
\x8e\r\xaa\x8965\u026d\xc5\u07a0\x00\x00\u07d4t\x95\xaex\xc0\xd9\x02a\xe2\x14\x0e\xf2\x061\x
04s\x1a`\xd1\xed\x89\x01\xdbPq\x89%!\x00\x00\u07d4t\x9aJv\x8b_#rH\x93\x8a\x12\xc6#\x84{\xd
4\xe6\x88\u0709\x03\xe73b\x87\x14
\x00\x00\u07d4t\x9a\xd6\xf2\xb5pk\xbe/h\x9aD\u0136@\xb5\x8e\x96\xb9\x92\x89\x05k\xc7^-
c\x10\x00\x00\u07d4t\xa1\u007f\x06K4N\x84\xdbce\u0695\x91\xff\x16(%vC\x89\x01\x15\x8eF\t\x1
3\xd0\x00\x00\u07d4t\xae\xec\x91]\xe0\x1c\u019b,\xb5\xa65o\xee\xa1FX\xc6\u0149\f\x9a\x95\xee
)\x86R\x00\x00\u07d4t\xaf\xe5I\x02\xd6\x15x%v\xf8\xba\xac\x13\xac\x97\f\x05\x0fn\x89\t\xa1\xaa\
xa3\xa9\xfb\xa7\x00\x00\u07d4t\xb7\xe0\"\x8b\xae\xd6YW\xae\xbbM\x91m3:\xae\x16O\x0e\x89lk\
x93[\x8b\xbd@\x00\x00\u07d4t\xbcJ^
E\xf4\xff\x8d\xb1\x84\xcf:\x9b\f\x06Z\xd8\a\u0489lk\x93[\x8b\xbd@\x00\x00\u07d4t\xbc\xe9\xec86
-
l\x94\u032c&\xd5\xc0\xe1:\x8b;\x1d@\x8965&A\x04B\xf5\x00\x00\u07d4t\xbfzZ\xb5\x92\x93\x14\x
9b\\`\xcf6Bc\xe5\xeb\xf1\xaa\r\x89\x06G\f>w\x1e<\x00\x00\xe0\x94t\xc7<\x90R\x8a\x15s6\xf1\xe7\
xea
b\n\xe5?\xd2G(\x8a\x01\xe6:.S\x8f\x16\xe3\x00\x00\u07d4t\u0464\xd0\xc7RN\x01\x8dN\x06\xed;d
\x80\x92\xb5\xb6\xaf,\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94t\xd3f\xb0{/VG}|pw\xaco\x
e4\x97\xe0\xebeY\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4t\xd3zQt{\xf8\xb7q\xbf\xbfC\x9493\xd1\x00\xd2\x14\x83\x8965\u026d\xc5\u07a0\x
00\x00\u07d4t\xd6q\u065c\xbe\xa1\xabW\x90cu\xb6?\xf4+PE\x1d\x17\x8965\u026d\xc5\u07a0\x0
0\x00\u07d4t\xeb\xf4BVF\xe6\u03c1\xb1\t\xce{\xf4\xa2\xa6=\x84\x81_\x89\x02+\x1c\x8c\x12'\xa0\
x00\x00\u07d4t\xed3\xac\xf4?5\xb9\x8c\x920\xb9\xe6d.\xcbS0\x83\x9e\x89$\xf6\xdf\xfbI\x8d(\x00\
x00\u07d4t\xef(i\xcb\xe6\b\x85`E\xd8\xc2\x04\x11\x18W\x9f\"6\xea\x89\x03<\xd6E\x91\x95n\x00\x
00\u07d4t\xfcZ\x99\xc0\xc5F\x05\x03\xa1;\x05\tE\x9d\xa1\x9c\xe7\u0350\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4u\v\xbb\x8c\x06\xbb\xbf$\bC\xccux.\xe0/\b\xa9tS\x89-
C\xf3\xeb\xfa\xfb,\x00\x00\u07d4u\x14\xad\xbd\xc6?H?0M\x8e\x94\xb6\u007f\xf30\x9f\x18\v\x82\x
89!\u0120n-

\x13Y\x80\x00\u0794u\x17\xf1l(\xd12\xbb@\xe3\xba6\u01ae\xf11\xc4b\xda\x17\x88\xfc\x93c\x92\
x80\x1c\x00\x00\u07d4u\x1a,\xa3Nq\x87\xc1c\u048e6\x18\xdb(\xb1<\x19m&\x89\x1b\x1a\xe4\xd6
\xe2\xefP\x00\x00\xe0\x94u\x1a\xbc\xb6\xcc\x030Y\x91\x18\x15\xc9o\u04516\n\xb0D-
\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4u&\xe4\x82R\x9f\n\x14\xee\u0248q\xdd\xdd\x0er\x1b\f
\u0662\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4u)\xf3y{\xb6\xa2\x0f~\xa6I$\x19\xc8L\x86vA\xd
8\x1c\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94u*^\xe22a,\xd3\x00_\xb2n[Y}\xe1\x9fwk\xe6\x8a\x01'\
xfc\xb8\xaf\xae
\xd0\x00\x00\u07d4u,\x9f\xeb\xf4/f\xc4x{\xfa~\xb1|\xf53;\xbaPp\x89j\x99\xf2\xb5O\xddX\x00\x00\u
07d4u930F\u07b1\xef<M\xafPa\x99\x93\xf4D\xe1\xdeh\x89@\x13\x8b\x91~\u07f8\x00\x00\u07d4
uS\xaa#\xb6\x8a\xa5\xf5~\x13_\xe3\x9f\xdc#^\xac\xa8\u024c\x8965\u026d\xc5\u07a0\x00\x00\xe
0\x94uZ`\xbfR/\xbd\x8f\xff\x97#Dk~4:phV~\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4u_X~^\xff\xf7
s\xa2
rj\x13\xd0\xf2\x13\r\x9f\x89k\x8965\u026d\xc5\u07a0\x00\x00\u07d4ub\x18e\xb6Y\x13e`n\xd3x0\x
8c-
\x1d\xefO\",\x89\xa8\r$g~\xfe\xf0\x00\x00\xe0\x94ucl\xdb\x10\x90P\xe4=]n\xc4~5\x9e!\x8e\x85~\u
028a\x04\u0632'l\x89b(\x00\x00\u07d4ufIab\xbaXCw\xbe\x04\nO\x87wzpz\xca\xeb\x89\xd8\xd7&\
xb7\x17z\x80\x00\x00\u07d4uk\x84\xeb\x85\xfc\xc1\xf4\xfc\xdc\u00b0\x8d\xb6\xa8n\x13_\xbc%\x
89\xae\x8ez\v\xb5u\xd0\x00\x00\u07d4uoE\xe3\xfai4z\x9a\x97:r^<\x98\xbcM\xb0\xb5\xa0\x89\n\u
05ce\xbcZ\xc6
\x00\x00\u07d4u{e\x87m\xbf)\xbf\x91\x1dO\x06\x92\xa2\u027e\xb1\x13\x98\b\x89\u07d3\xa5\x93
7\xd6\u0740\x00\u07d4u\u007f\xa5TF\xc4`\x96\x8b\xb7K^\xbc\xa9lN\xf2\xc7\t\u01497\b\xba\xed=
h\x90\x00\x00\u07d4u\x80J\xacd\xb4\x19\x90\x83\x98)\x02\x99M\x9c^\u0602\x8f\x11\x89\x1e=\a\
xb0\xa6
\xe4\x00\x00\u07d4u\x92\u019d\x06{Q\xb6\xccc\x9d\x11d\xd5W\x8c`\xd2\xd2D\x89\x01\x15\x8eF
\t\x13\xd0\x00\x00\u07d4u\xab\xe5'\x0f:x\xce\x00|\xf3\u007f\x8f\xbc\x04]H\x9b{\xb1\x89lj\xccg\u05
f1\xd4\x00\x00\xe0\x94u\xacTp\x17\x13L\x04\xae\x1e\x11\xd6\x0ec\xec\x04\u044d\xb4\xef\x8a\x0
1EB\xba\x12\xa37\xc0\x00\x00\u07d4u\xb0\xe9\xc9B\xa4\xf0\xf6\xf8m?\x95\xff\x99\x80\"\xfag\x96
;\x89P\xc5\xe7a\xa4D\b\x00\x00\xe0\x94u\xb9V\x96\xe8\xecE\x10\xd5hh\xa7\xc1\xa75\u018b$H\
x90\x8a\x01Z\xf1\u05cbX\xc4\x00\x00\x00\u07d4u\xbe\x8f\xf6^W\x88\xae\u01b2\xa5-
_\xa7\xb1\xe7\xa0;\xa6u\x89\x03\xab\xcd\xc54=t\x00\x00\u07d4u\xc1\x1d\x02M\x12\xaeHl\x10\x9
5\xb7\xa7\xb9\u012f>\x8e\u07b9\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94u\xc1\xad#\xd2?$\x
b3\x84\xd0\xc3\x14\x91w\xe8f\x97a\r!\x8a\x01\\[\xcdl(\x8b\xbd\x00\x00\u07d4u\xc2\xff\xa1\xbe\xf5
l\x19\xd2\t\u007fz\x14-
.\x14\xf9\xb0JX\x89\x90\xf3XP@2\xa1\x00\x00\u07d4u\xd6|\xe1N\x8d)\xe8\xc2\xff\u3051{\x93\v\x
1a\xff\x1a\x87\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4u\xde~\x93R\xe9\v\x13\xa5\x9aXx\xff\xec\u
01c3\x1c\xacM\x82\x89\x94\x89#z\u06daP\x00\x00\u07d4u\xf7S\x9d0\x9e\x909\x98\x9e\xfe.\x8b-
\xbd\x86Z\r\xf0\x88\x89\x85[[\xa6\\\x84\xf0\x00\x00\u07d4v\b\xf47\xb3\x1f\x18\xbc\vd\u04c1\xae\x
86\xfd\x97\x8e\u05f3\x1f\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94v\x0f\xf35N\x0f\u0793\
x8d\x0f\xb5\xb8,\xef[\xa1\\=)\x16\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4v\x1an6,\x97\xf
b\xbd|Yw\xac\xba-
\xa7F\x876_I\x89\t\xf7J\xe1\xf9S\xd0\x00\x00\u07d4v\x1el\xae\xc1\x89\xc20\xa1b\xec\x00e0\x19
>g\u03dd\x19\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94v\x1f\x8a:*\U00028f7e\x1d\xa0\t2\x1f\xb2\x97

d\xebb\xa1\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4v)\x98\xe1\xd7R'\xfc\xedzp\xbe\x10\x9aL\vN\xd8d\x14\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4v-o0\u06b9\x915\xe4\xec\xa5\x1dRC\xd6\xc8b\x11\x02\u0549\x0fI\x89A\xe6d(\x00\x00\u07d4v3\x1e0yl\xe6d\xb2p\x0e\rASp\x0e\u0706\x97w\x89lk\x93[\x8b\xbd@\x00\x00\u07d4v8\x86\xe33\xc5o\xef\xf8[\xe3\x95\x1a\xb0\xb8\x89\xce&.\x95\x89lk\x93[\x8b\xbd@\x00\x00\u07d4v:|\xba\xb7\rzd\u0427\xe5)\x80\xf6\x81G%\x93I\f\x89
\x86\xac5\x10R`\x00\x00\u07d4v>\xec\u0c0a\u021e2\xbf\xa4\xbe\xcev\x95\x14\xd8\xcb[\x85\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4v@\xa3\u007f\x80R\x98\x15\x15\xbc\xe0x\u0693\xaf\xa4x\x9bW4\x89lk\x93[\x8b\xbd@\x00\x00\u0794vA\xf7\xd2j\x86\xcd\xdb+\xe10\x81\x81\x0e\x01\xc9\xc8<K
\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4vF\x92\xcc\xcb3@]\u042b\f3y\xb4\x9c\xaf\x8eb!\xba\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4vMR\x12&:\xffJ*\x14\xf01\xf0N\xc7I\u0708>E\x89dI\xe8NG\xa8\xa8\x00\x00\xe0\x94vO\xc4mB\x8bm\xbc\"\x8a\x0f_U\xc9P\x8cw.\xab\x9f\x8a\x05\x81v{\xa6\x18\x9c@\x00\x00\u07d4vPn\xb4\xa7\x80\xc9Q\xc7J\x06\xb0=;\x83b\xf0\x99\x9dq\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94v[\xe2\xe1/b\x9ecI\xb9}!\xb6*\x17\xb7\xc80\xed\xab\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94vb\x81P\xe2\x99[['\x9f\xc8>\r\xd5\xf1\x02\xa6q\xdd\x1c\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4vk7Y\xe8yN\x92m\xacG=\x91:\x8f\xb6\x1a\xd0\xc2\u0249\x04\xb0m\xbb\xb4\x0fJ\x00\x00\u07d4vp\xb0/,<\xf8\xfdOG0\xf38\x1aq\xeaC\x1c3\u01c9\x0e~\xeb\xa3A\vt\x00\x00\u07d4vz\x03eZ\xf3`\x84\x1e\x81\r\x83\xf5\xe6\x1f\xb4\x0fL\xd1\x13\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4vz\u0190y\x1c.#E\x10\x89\xfelp\x83\xfeU\u07b6+\x89,s\xc97t,P\x00\x00\u07d4v\u007f\xd7y}Qi\xa0_sd2\x1c\x19\x84:\x8c4\x8e\x1e\x89\x01\x04\xe7\x04d\xb1X\x00\x00\u0794v\x84o\r\xe0;Zv\x97\x1e\xad)\x8c\xdd\b\x84:K\xc6\u0188\xd7\x1b\x0f\u088e\x00\x00\xe0\x94v\x84\x98\x93N7\xe9\x05\xf1\xd0\xe7{D\xb5t\xbc\xf3\xecJ\xe8\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4v\x8c\xe0\u06a0)\xb7\xde\xd0\"\xe5\xfcWM\x11\xcd\xe3\xec\xb5\x17\x89\x11t\xa5\xcd\xf8\x8b\xc8\x00\x00\xe0\x94v\x93\xbd\xebo\xc8+[\xcar\x13U\"1u\xd4z\bKM\x8a\x04\xa8\x9fT\xef\x01!\xc0\x00\x00\u07d4v\xaa\xf8\xc1\xac\x01/\x87R\xd4\xc0\x9b\xb4f\a\xb6e\x1d\\\xa8\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4v\xab\x87\xddZ\x05\xad\x83\x9aN/\xc8\xc8Z\xa6\xba\x05d\x170\x89lk\x93[\x8b\xbd@\x00\x00\u07d4v\xaf\xc2%\xf4\xfa0}\xe4\x84U+\xbe\x1d\x9d?\x15\aLJ\x89\xa2\x90\xb5\u01ed9h\x00\x00\xe0\x94v\xbe\xca\xe4\xa3\x1d6\xf3\xcbW\u007f*CYO\xb1\xab\xc1\xbb\x96\x8a\x05C\xa9\xce\x0e\x132\xf0\x00\x00\u07d4v\xc2u5\xbc\xb5\x9c\xe1\xfa-\x8c\x91\x9c\xab\xebJk\xba\x01\u0449lk\x93[\x8b\xbd@\x00\x00\u07d4v\xca\"\xbc\xb8y\x9eS'\u012a*}\tI\xa1\xfc\xce_)\x89R\xa0?\"\x8cZ\xe2\x00\x00\u07d4v\xca\u0108\x11\x1aO\u0555\xf5h\xae:\x85\x87p\xfc\x91]_\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94v\u02dc\x8bi\xf48vu\u0102S\xe24\xcb~\rt\xa4&\x8a\x01\x90\xf4H.\xb9\x1d\xae\x00\x00\u07d4v\xf8:\xc3\xda0\xf7\t&(\xc73\x9f
\x8b\xfc\x14,\xb1\ue25a\x18\xff\xe7B}d\x00\x00\xe0\x94v\xf9\xad=\x9b\xbd\x04\xae\x05\\\x14w\xc0\xc3^u\x92\xcb* \x8a\b\x83?\x11\xe3E\x8f
\x00\x00\u07d4v\xff\xc1W\xadk\xf8\xd5m\x9a\x1a\u007f\u077c\x0f\xea\x01\n\xab\xf4\x8965\u026d\xc5\u07a0\x00\x00\u07d4w\x02\x8e@\x9c\xc4:;\xd3=!\xa9\xfcS\xec`n\x94\x91\x0e\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4w\f/\xb2\u0128\x17S\xac\x01\x82\xeaF\x0e\xc0\x9c\x90\xa5\x16\xf8\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4w\r\x98\xd3\x1bCS\xfc\xee\xe8V\fL\u03c0>\x88\xc0\xc4\xe0\x89

\x86\xac5\x10R`\x00\x00\xe0\x94w\x13\xab\x807A\x1c\t\xbah\u007fo\x93d\xf0\xd3#\x9f\xac(\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4w\x15\a\xae\xeej%]\xc2\u035d\xf5QT\x06-
\b\x97\xb2\x97\x89\x12\x1e\xa6\x8c\x11NQ\x00\x00\u07d4w\x19\x88\x87\x95\xadtY$\xc7W`\u0771\x82}\xff\xd8\u0368\x89lkLM\xa6\u077e\x00\x00\u07d4w'\xaf\x10\x1f\n\xab\xa4\xd2:\x1c\xaf\xe1|
n\xb5\u06b1\xc6\u0709lk\x93[\x8b\xbd@\x00\x00\u07d4w,)\u007f\n\u0454H.\xe8\xc3\xf06\xbd\xeb\x01\xc2\x01\xd5\u0309\n\u05ce\xbcZ\xc6 \x00\x00\xe0\x94w0o\xfe.J\x8f<\xa8&\xc1\xa2I\xf7!-
\xa4:\xef\xfd\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4w1A\x12}\x8c\xf3\x18\xae\xbf\x886Z\xdd=U'
\xd8[j\x8966\u05ef^\u024e\x00\x00\u07d4wF\xb6\xc6i\x9c\x8f4\xca'h\xa8
\xf1\xff\xa4\xc2\a\xfe\x05\x89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4wQ\xf3c\xa0\xa7\xfd\x053\x19\b\t
\u076f\x93@\xd8\xd1\x12\x91\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4wW\xa4\xb9\xcc=\x02G\
u032a\xeb\x99\t\xa0\xe5n\x1d\xd6\xdc\u0089\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4w\\\x10\xc9>\
r\xb7
[&CE\x823\xc6O\xc3?\xd7[\x89lk\x93[\x8b\xbd@\x00\x00\u07d4wa~\xbcK\xeb\xc5\xf5\xdd\xeb\x1
bzp\xcd\xebj\xe2\xff\xa0$\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4wiC\xff\xb2\xef\\\xdd5\xb8<(\xbc\x
04k\xd4\xf4gp\x98\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94wp\x1e,I=\xa4|\x1bX\xf4!\xb5I]\xeeE\x
be\xa3\x9b\x8a\x01H\xf6I\xcfaB\xa5\x80\x00\u07d4wy\x8f
\x12W\xb9\xc3R\x04\x95pW\xb5Ft\xae\xfaQ\u07c9\b\x13\xcaV\x90m4\x00\x00\u07d4w\x8cC\xd1\
x1a\xfe;Xo\xf3t\x19-
\x96\xa7\xf2=+\x9b\u007f\x89\x8b\xb4\xfc\xfa;}k\x80\x00\u07d4w\x8cy\xf4\xde\x19S\xeb\u0398\xf
e\x80\x06\xd5:\x81\xfbQ@\x12\x8963\x03\"\xd5#\x8c\x00\x00\u07d4w\x92t\xbf\x18\x03\xa36\xe4\
u04f0\r\u0753\xf2\xd4\xf5\xf4\xa6.\x8965\u026d\xc5\u07a0\x00\x00\u07d4w\xa1q\"\xfa1\xb9\x8f\x
17\x11\xd3*\x99\xf0>\xc3&\xf3=\b\x89\\(=A\x03\x94\x10\x00\x00\u07d4w\xa3I\a\xf3\x05\xa5L\x85\
xdb\t\xc3c\xfd\xe3\xc4~j\xe2\x1f\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4w\xa7i\xfa\xfd\xec\xf4\xa68
v-[\xa3\x96\x9d\xf61
\xa4\x1d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4w\xbekd\xd7\xc73\xa46\xad\xec^\x14\xbf\x9a\xd7@
+\x1bF\x8965\u026d\xc5\u07a0\x00\x00\u07d4w\xbf\xe9<\u0367P\x84~A\xa1\xaf\xfe\xe6\xb2\u0696\xe7!N\x89\x10CV\x1a\x88)0\x00\x00\u07d4w\u0126\x97\xe6\x03\xd4+\x12\x05l\xbb\xa7a\xe7\
xf5\x1d\x04C\xf5\x89$\xdc\xe5M4\xa1\xa0\x00\x00\u07d4w\xcc\x02\xf6#\xa9\u03d8S\t\x97\xeag\xd9\\;I\x18Y\xae\x89Is\x03\xc3n\xa0\xc2\x00\x00\u07d4w\xd4?\xa7\xb4\x81\xdb\xf3\xdbS\f\xfb\xf5\
xfd\xce\xd0\xe6W\x181\x89lk\x93[\x8b\xbd@\x00\x00\u07d4w\xda^lr\xfb6\xbc\xe1\xd9y\x8f{\xcd\xf
1\u044fE\x9c.\x89\x016\x95\xbbl\xf9>\x00\x00\u07d4w\xf4\xe3\xbd\xf0V\x88<\xc8r\x80\xdb\xe6@\
xa1\x8a\r\x02\xa2\a\x89\n\x81\x99:+\xfb[\x00\x00\u0794w\xf6\t\u0287 \xa0#&,U\xc4o-
&\xfb90\xaci\x88\xf0\x15\xf2W6B\x00\x00\u07d4w\xf8\x1b\x1b&\xfc\x84\xd6\u0797\uf2df\xbdr\xa3
10\xccJ\x8965\u026d\xc5\u07a0\x00\x00\u07d4x\x19\xb0E\x8e1N+S\xbf\xe0\f8I_\u0539\xfd\xf8\u
0589\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4x\x1b\x15\x01dz.\x06\xc0\xedC\xff\x19\u007f\xcc\xec
5\xe1p\v\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4x/R\xf0\xa6v\xc7w\x16\xd5t\xc8\x1e\xc4hO\x9a\x
02\n\x97\x89.\x14\xe2\x06\xb70\xad\x80\x00\u07d4x5]\xf0\xa20\xf8=\x03,p1TAM\xe3\xee\u06b5
W\x89lk\x93[\x8b\xbd@\x00\x00\u07d4x6\xf7\xefk\u01fd\x0f\xf3\xac\xafD\x9c\x84\xddk\x1e,\x93\x
9f\x89\xe0\x8d\xe7\xa9,\xd9|\x00\x00\u07d4x7\xfc\xb8v\xda\x00\xd1\xeb;\x88\xfe\xb3\xdf?\xa4\x0
4/\xac\x82\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4x>\uc2a5\xda\xc7{.f#\xedQ\x98\xa41\xab\xba\
xee\a\x89\x17\xda:\x04\u01f3\xe0\x00\x00\u07d4x\\\x8e\xa7t\xd70D\xa74\xfay\n\x1b\x1et>w\xed|\
x89\f\xf1Rd\f\\\x83\x00\x00\u07d4x`\xa3\xde8\xdf8*\xe4\xa4\xdc\xe1\x8c\f\a\xb9\x8b\xce=\xfa\x89

65\u026d\xc5\u07a0\x00\x00\xe0\x94xcCq\xe1s\x04\xcb\xf39\xb1E*L\xe48\xdcvL\u038a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4xd\u0719\x9f\xe4\xf8\xe0\x03\xc0\xf4=\xec\u00da\xae\x15\"\xdc\x0f\x89\x05\x1e\x10+\xd8\xec\xe0\x00\x00\u07d4xtj\x95\x8d\xce\xd4\xc7d\xf8vP\x8cAJh4,\uce49\x02\xbe7O\xe8\xe2\xc4\x00\x00\xe0\x94x}1?\xd3k\x05>\xee\xae\xdb\xcet\xb9\xfb\x06x32\x89\x8a\x05\xc0X\xb7\x84'\x19`\x00\x00\u07d4x\x85\x9c[T\x8bp\r\x92\x84\xce\xe4\xb6c</R\xe5)\u0089\xa00\xdc\xeb\xbd/L\x00\x00\xe0\x94x\x8e\x80\x97A\xa3\xb1J\"\xa4\xb1\xd97\xc8,\xfe\xa4\x89\uef8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4x\xa1\xe2T@\x9f\xb1\xb5Z|\xb4\u074e\xba;0\u023a\xd9\xef\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94x\xa5\xe8\x99\x00\xbd?\x81\xddq\xba\x86\x9d%\xfe\xc6Ra\xdf\x15\x8a\n\xfd\x81/\xee\x03\xd5p\x00\x00\xe0\x94x\xb9x\xa9\xd7\xe9\x1e\xe5)\xeaO\u0137o\xea\xf8v/i\x8c\x8a\x06\u01b95\xb8\xbb\xd4\x00\x00\x00\xe0\x94x\xce>=GJ\x8a\x04{\x92\xc4\x15B$-
\n\b\xc7\x0f\x99\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4x\u03c36\xb3(\xdb=\x87\x81:G+\x9e\x89\xb7^\f\xf3\xbc\x8965\u026d\xc5\u07a0\x00\x00\u07d4x\xd4\xf8\xc7\x1c\x1eh\xa6\x9a\x98\xf5/\xcbE\u068a\xf5n\xa1\xa0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4x\xdf&\x81\xd6\xd6\x02\xe2!B\xd5A\x16\u07a1]EIW\xaa\x89\x10'\x94\xad
\xdah\x00\x00\u07d4x\xe0\x8b\xc53A<&\u2473\x14?\xfa|\u026f\xb9{x\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4x\xe8?\x80\xb3g\x8cz\nN>\x8c\x84\xdc\xcd\xe0dBbw\x89a\t=|,m8\x00\x00\u07d4x\xf5\xc7G\x85\xc5f\x8a\x83\x80r\x04\x8b\xf8\xb4SYM\u06ab\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4y\x0f\x91\xbd]\x1c\\\xc4s\x9a\xe9\x13\x00\u06c9\xe1\xc10<\x93\x89lk\x93[\x8b\xbd@\x00\x00\u07d4y\x17\u5f42\xa9y\x0f\xd6P\xd0C\xcd\xd90\xf7y\x963\u06c9\xd8\xd4`,&\xbfl\x00\x00\u07d4y\x19\xe7b\u007f\x9b}T\xea;\x14\xbbM\xd4d\x9fO9\xde\xe0\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4y\x1f`@\xb4\xe3\xe5\r\xcf5S\xf1\x82\u0357\xa9\x060\xb7]\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4y0\xc2\xd9\xcb\xfa\x87\xf5\x10\xf8\xf9\x87w\xff\x8a\x84H\xcaV)\x89\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4yE)\u041d\x01rq5\x970\x02pu\xb8z\xd8=\xaen\x89\x10\xce\x1d=\x8c\xb3\x18\x00\x00\u07d4yKQ\u00deS\xd9\xe7b\xb0a;\x82\x9aD\xb4r\xf4\xff\xf3\x89$5\xe0dxA\u0300\x00\xe0\x94yU\x1c\xed\xe3v\xf7G\xe3ql\x8dy@\rvm.\x01\x95\x8a\t\xcb7\xaf\xa4\xffxh\x00\x00\u07d4y^\xbc&&\xfc9\xb0\xc8b\x94\xe0\xe87\xdc\xf5#U0\x90\x8965\u026d\xc5\u07a0\x00\x00\u07d4yn\xbb\xf4\x9b>6\xd6v\x94\xady\xf8\xff6vz\xc6\xfa\xb0\x89\x03K\xc4\xfd\xde'\xc0\x00\x00\u07d4yo\x87\xbaaz)0\xb1g\v\xe9.\xd1(\x1f\xb0\xb3F\xe1\x89\x06\xf5\xe8o\xb5((\x00\x00\u07d4yt'\xe3\xdb\xf0\xfe\xaez%\x06\xf1-
\xf1\xdc@2n\x85\x05\x8965\u026d\xc5\u07a0\x00\x00\u07d4yu\x10\xe3\x86\xf5c\x93\xce\xd8\xf4w7\x8aDLHO}\xad\x8965\u026d\xc5\u07a0\x00\x00\u07d4y{\xb7\xf1W\xd9\xfe\xaa\x17\xf7m\xa4\xf7\x04\xb7M\xc1\x03\x83A\x89\xb5\x0f\u03ef\xeb\xec\xb0\x00\x00\u07d4y\x88\x90\x131\xe3\x87\xf7\x13\xfa\u03b9\x00\\\xb9\xb6Q6\xeb\x14\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4y\x89\u041f8&\xc3\u5bccu*\x81\x15r:\x84\xd8\tp\x89\x16\x86\xf8aL\xf0\xad\x00\x00\xe0\x94y\x95\xbd\x8c\xe2\xe0\xc6{\xf1\u01e51\xd4w\xbc\xa1\xb2\xb9ua\x8a\x01BH\xd6\x17\x82\x9e\xce\x00\x00\u07d4y\xae\xb3Ef\xb9t\xc3ZX\x81\xde\xc0 \x92}\xa7\xdf]%\x89lk\x93[\x8b\xbd@\x00\x00\u07d4y\xb1
\xeb\x88\x06s#!(\x8fgZ'\xa9\"_\x1c\xd2\ub245\xa0\xbf7\xde\xc9\xe4\x00\x00\u07d4y\xb4\x8d-a7\u00c5Ma\x1c\x01\xeaBBz\x0fY{\xb7\x89\nZ\xa8P\t\xe3\x9c\x00\x00\u07d4y\xb8\xaa\xd8y\xdd0V~\x87x\xd2\xd21\xc8\xf3z\xb8sN\x89lk\x93[\x8b\xbd@\x00\x00\u07d4y\xbf/{n2\x8a\xaf&\xe0\xbb\t?\xa2-
\xa2\x9e\xf2\xf4q\x89a\t=|,m8\x00\x00\u07d4y\xc10\xc7b\xb8v[\x19\u04ab\u0260\x83\xab\x8f:\xad

y@\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4y\xc1\xbe\x19q\x1fs\xbe\xe4\xe61j\xe7T\x94Y\xaa\u0
3a2\xe0\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4y\xc6\x00/\x84R\xca\x15\u007f\x13\x17\xe8\n/
\xaf$GUY\xb7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4y\xca\xc6IO\x11\xef'\x98t\x8c\xb52\x85\
xbd\x8e\"\xf9|\u0689lk\x93[\x8b\xbd@\x00\x00\u07d4y\u03e9x\n\xe6\xd8{,1\x88?\t'i\x86\u021ag5\
x8965\u026d\xc5\u07a0\x00\x00\u07d4y\u06e2VG-
\xb4\xe0X\xf2\xe4\xcd\xc3\xeaN\x8aBw83\x89O%\x91\xf8\x96\xa6P\x00\x00\u07d4y\xed\x10\xcf\x
1fm\xb4\x82\x06\xb5\t\x19\xb9\xb6\x97\b\x1f\xbd\xaa\xf3\x89lk\x93[\x8b\xbd@\x00\x00\u0794y\xf
0\x8e\x01\xce\t\x88\xe6<\u007f\x8f)\b\xfa\xdeC\xc7\xf9\xf5\u0248\xfc\x93c\x92\x80\x1c\x00\x00\u
07d4y\xfdmH1Pf\xc2\x04\xf9e\x18i\xc1\tl\x14\xfc\x97\x81\x89lk\x93[\x8b\xbd@\x00\x00\u0794y\xff
\xb4\xac\x13\x81*\vx\u0123{\x82u\">\x17k\xfd\xa5\x88\xf0\x15\xf2W6B\x00\x00\u07d4z\x05\x89\x
b1C\xa8\xe5\xe1\a\u026cf\xa9\xf9\xf8Yz\xb3\u7ac9Q\xe92\xd7n\x8f{\x00\x00\u07d4z\nx\xa9\xcc9
?\x91\xc3\xd9\xe3\x9ak\x8c\x06\x9f\a^k\xf5\x89Hz\x9a0E9D\x00\x00\u07d4z\x13p\xa7B\xec&\x87
\xe7a\xa1\x9a\u0167\x942\x9e\xe6t\x04\x89\xa2\xa12ga\xe2\x92\x00\x00\xe0\x94z-
\xfcw\x0e$6\x811\xb7\x84w\x95\xf2\x03\xf3\xd5\r[V\x8a\x02i\xfe\xc7\xf06\x1d
\x00\x00\u07d4z3\x83N\x85\x83s>-
R\xae\xadX\x9b\u046f\xfb\x1d\xd2V\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94z6\xab\xa5\xc3\x1e\
xa0\xca~'{\xaa2\xecF\u0393\xcfu\x06\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94z8\x11\"\xba\xday
\x1az\xb1\xf6\x03}\xac\x80C'S\xba\xad\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94zH\xd8
w\xb6:\x8f\x8f\x93\x83\xe9\xd0\x1eS\xe8\fR\x8e\x95_\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4
zO\x9b\x85\x06\x90\xc7\xc9F\x00\xdb\xee\f\xa4\xb0\xa4\x11\xe9\xc2!\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4zc\x86\x9f\xc7g\xa4\u01b1\xcd\x0e\x06I\xf3cL\xb1!\xd2K\x89\x043\x87O
c,\xc6\x00\x00\u07d4zg\xdd\x04:PO\xc2\xf2\xfcq\x94\xe9\xbe\xcfHL\xec\xb1\xfb\x89\r\x8drkqw\xa
8\x00\x00\xe0\x94zk&\xf48\u0663RD\x91U\xb8\x87l\xbd\x17\xc9\u065bd\x8a\x01EB\xba\x12\xa3
7\xc0\x00\x00\u07d4zmx\x1cw\u013a\x1f\xca\xdfhsA\xc1\xe3\x17\x99\xe9='\x89\x0e\u0683\x8cI)\
b\x00\x00\u07d4zph\xe1\xc37\\\x0eY\x9d\xb1\xfb\xe6\xb2\xea#\xb8\xf4\a\u0489lk\x93[\x8b\xbd@\
x00\x00\u07d4zt\xce\xe4\xfa\x0fcp\xa7\x89O\x11l\xd0\f\x11G\xb8>Y\x89+^:\xf1k\x18\x80\x00\x00\
u07d4zy\xe3\x0f\xf0W\xf7\n=\x01\x91\xf7\xf5?v\x157\xaf}\xff\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\
xe0\x94zzO\x80sW\xa4\xbb\xe6\x8e\x1a\xa8\x0692\x10\xc4\x11\u0333\x8a\x06ZM\xa2]0\x16\xc0\
x00\x00\u07d4z\x85c\x86y\x01
o?+\xf0\xfa>\x1c\x81\t\u02bc\u0345\x89\amA\xc6$\x94\x84\x00\x00\xe0\x94z\x87\x97i\n\xb7{Tp\x
bf|\f\x1b\xbaa%\b\xe1\xac}\x8a\x01\xe0\x92\x96\xc37\x8d\xe4\x00\x00\u07d4z\x8c\x89\xc0\x14P\
x9dV\u05f6\x810f\x8f\xf6\xa3\xec\xecsp\x89\x10CV\x1a\x88)0\x00\x00\xe0\x94z\x94\xb1\x99\x92\
u03b8\xcec\xbc\x92\xeeKZ\xde\xd1\fM\x97%\x8a\x03\x8d\x1a\x80d\xbbd\xc8\x00\x00\u07d4z\xa7
\x9a\xc0C\x16\u030d\b\xf2\x00e\xba\xa6\xd4\x14(\x97\xd5N\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4z
\xadM\xbc\u04ec\xf9\x97\u07d3XiV\xf7+d\u062d\x94\xee\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0
\x94z\xb2V\xb2\x04\x80\n\xf2\x017\xfa\xbc\xc9\x16\xa22Xu%\x01\x8a\x04<3\xc1\x93ud\x80\x00\x
00\u07d4z\xbaV\xf6:H\xbc\b\x17\u05b9p9\x03\x9az\xd6/\xae.\x89
\x86\xac5\x10R`\x00\x00\xe0\x94z\xbb\x10\xf5\xbd\x9b\xc3;\x8e\xc1\xa8-
d\xb5[k\x18wuA\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4z\u010d@\xc6d\u031am\x89\xf1\xc5\xf5
\xc8\n\x1cp\xe7D\u6263\x10b\xbe\xee\xd7\x00\x00\x00\u07d4z\u014fo\xfcO\x81\a\xaen07\x8eN\x
9f\x99\xc5\u007f\xbb$\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4z\xd3\xf3\aao\x19\u0731C\xe6D
M\xab\x9c<3a\x1fR\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4z\xd8,\xae\xa1\xa8\xb4\xed\x

051\x9b\x9c\x98p\x17<\x81N\x06\xee\x89!d\xb7\xa0J\u0220\x00\x00\u07d4z\xde]f\xb9D\xbb\x86\f\x0e\xfd\xc8bv\u054fFS\xf7\x11\x89lk\x93[\x8b\xbd@\x00\x00\u07d4z\xdf\xed\xb0m\x91\xf3\xccs\x90E\v\x85U\x02p\x88<{\xb7\x89\x11x\xfa@Q]\xb4\x00\x00\u07d4z\xe1\xc1\x9eS\xc7\x1c\xeeLs\xfa\xe2\xd7\xfcs\xbf\x9a\xb5\u348965\u026d\xc5\u07a0\x00\x00\u07d4z\xe6Y\xeb;\xc4hR\xfa\x86\xfa\xc4\xe2\x1cv\x8dP8\x89E\x89\x0f\x81\f\x1c\xb5\x01\xb8\x00\x00\u07d4z\xea%\xd4+&\x12(n\x99\xc56\x97\u01bcA\x00\xe2\u06ff\x89lk\x93[\x8b\xbd@\x00\x00\u07d4z\xef{U\x1f\v\x9cF\xe7U\xc0\xf3\x8e[:s\xfe\x11\x99\xf5\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4{\v1\xffn$t^\xad\x8e\u067b\x85\xfc\v\xf2\xfe\x1dU\u0509+^:\xf1k\x18\x80\x00\x00\xe0\x94{\x0f\xea\x11v\xd5!Y3:\x14<)IC\xda6\xbb\u0774\x8a\x01\xfc}\xa6N\xa1L\x10\x00\x00\u07d4{\x11g<\xc0\x19bk)\f\xbd\xce&\x04o~m\x14\x1e!\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4{\x12!b\xc9\x13\xe7\x14l\xad\v~\xd3z\xff\xc9*\v\xf2\u007f\x89Q\xaf\tk#\x01\u0440\x00\u07d4{\x1b\xf5:\x9c\xbe\x83\xa7\u07a44W\x9f\xe7*\xac\x8d*\f\u0409\n\xd4\xc81j\v\f\x00\x00\u07d4{\x1d\xaf\x14\x89\x1b\x8a\x1e\x1b\xd4)\u0633k\x9aJ\xa1\u066f\xbf\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4{\x1f\xe1\xabM\xfd\x00\x88\xcd\xd7\xf6\x01c\xefY\xec*\xee\x06\xf5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4{%\xbb\x9c\xa8\xe7\x02!~\x933\"RP\xe5<6\x80MH\x89e\xea=\xb7UF`\x00\x00\u07d4{'\xd0\xd1\xf3\xdd<\x14\x02\x94\xd0H\x8bx>\xbf@\x15'}\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94{@\a\xc4^ZW?\u06f6\xf8\xbdtk\xf9J\xd0J<&\x8a\x038!\xf5\x13]%\x9a\x00\x00\u07d4{C\xc7\xee\xa8\xd6#U\xb0\xa8\xa8\x1d\xa0\x81\xc6Dk3\xe9\xe0\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4{M*8&\x90i\xc1\x85Ww\rY\x1d$\xc5\x12\x1f^\x83\x89%\xf2s\x93=\xb5p\x00\x00\xe0\x94{au\xec\x9b\xef\xc78$\x955\xdd\xde4h\x8c\xd3n\xdf%\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94{f\x12hy\x84M\xfa4\xfee\xc9\xf2\x88\x11\u007f\xef\xb4I\xad\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4{j\x84q\x8d\xd8nc3\x84)\xac\x81\x1d|\x8a\x86\x0f!\xf1\x89a\t=|,m8\x00\x00\xe0\x94{q,z\xf1\x16v\x00jf\xd2\xfc\\\x1a\xb4\xc4y\xce`7\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4{s$-
u\u029a\xd5X\xd6P)\r\xf1v\x92\xd5L\u0638\x89lnY\xe6|xT\x00\x00\u07d4{v\x1f\xeb\u007f\u03e7\xde\xd1\xf0\xeb\x05\x8fJ`\v\xf3\xa7\b\u02c9\xf9]\xd2\xec'\xcc\xe0\x00\x00\xe0\x94{\x82|\xae\u007f\xf4t\t\x18\xf2\xe00\xab&\u02d8\xc4\xf4l\xf5\x8a\x01\x94hL\v9\xde\x10\x00\x00\xe0\x94{\x892\x86B~r\xdb!\x9a!\xfcM\xcd_\xbfY(<1\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4{\x92&\xd4o\xe7Q\x94\v\xc4\x16\xa7\x98\xb6\x9c\xcf\r\xfa\xb6g\x89\u3bb5sr@\xa0\x00\x00\u07d4{\x98\xe2<\xb9k\xee\xe8\n\x16\x80i\ube8f
\xed\xd5\\\u03c9\v\xa0\xc9\x15\x87\xc1J\x00\x00\u07d4{\xb0\xfd\xf5\xa6c\xb5\xfb\xa2\x8d\x9c\x90*\xf0\xc8\x11\xe2R\xf2\x98\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4{\xb9W\x1f9K\v\x1a\x8e\xbaVd\xe9\u0635\xe8@g{\xea\x89\x01\x11du\x9f\xfb2\x00\x00\xe0\x94{\xb9\x84\xc6\u06f9\xe2y\x96j\xfa\xfd\xa5\x9c\x01\xd0&'\xc8\x04\x8a\x01\xb4d1\x1dE\xa6\x88\x00\x00\u07d4{\xbb\xec^p\xbd\xea\u063b2\xb4(\x05\x98\x8e\x96H\xc0\xaa\x97\x8966\u05ef^\u024e\x00\x00\u07d4{\xca\x1d\xa6\xc8\nf\xba\xa5\xdbZ\u0245A\u013e'kD}\x89$\xcf\x04\x96\x80\xfa<\x00\x00\u07d4{\u0772\xee\x98\xde\x19\xeeL\x91\xf6a\xee\x8eg\xa9\x1d\x05K\x97\x8965\u026d\xc5\u07a0\x00\x00\u0794{\xe2\xf7h\f\x80-
\xa6\x15L\x92\xc0\x19J\xe72Qzqi\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4{\xe7\xf2Eiq\x88;\x9a\x8d\xbeL\x91\xde\xc0\x8a\xc3N\x88b\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4{\xe8\u0334\xf1\x1bf\xcan\x1dW\xc0\xb59b!\xa3\x1b\xa5:\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94{\xeb\x81\xfb/^\x91Rk*\xc9y^v\u019b\xcf\xf0K\xc0\x8a\x0e\xb2.yO\n\x8d`\x00\x00\u07d4|\b\x83\x05L-
\x02\xbcz\x85+\x1f\x86\xc4'w\xd0\xd5\xc8V\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4|\x0f^\a

C\xc9\xeet\x02B\x19~x\xccK\x98\xcd\xf9`\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4|\x1d\xf2JO\u007f\xb2\u01f4r\xe0\xbb\x00l\xb2}\xcd\x16AV\x8965\u026d\xc5\u07a
0\x00\x00\u07d4|)\xd4}W\xa73\xf5k\x9b!pc\xb5\x13\xdc;1Y#\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\
u07d4|+\x96\x03\x88JO.FN\u03b9}\x17\x93\x8d\x82\x8b\xc0,\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u0
7d4|8,\x02\x96a.N\x97\xe4@\xe0-
8q';U\xf5;\x89\n\xb6@9\x12\x010\x00\x00\u07d4|>\xb7\x13\xc4\xc9\xe08\x1c\xd8\x15L|\x9a}\xb8d
\\\xde\x17\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4|D\x01\xae\x98\xf1.\xf6\xde9\xae$\u03df\xc5\x1f\x80\xeb\xa1k\x89\n\u05ce\xbcZ\x
c6
\x00\x00\u07d4|E\xf0\xf8D*V\xdb\u04dd\xbf\x15\x99\x95A\\R\xedG\x9b\x89lk\x93[\x8b\xbd@\x00\
x00\xe0\x94|S-\xb9\xe0\xc0l&\xfd@\xac\xc5j\xc5\\\x1e\xe9-
<:\x8a?\x87\bW\xa3\xe0\xe3\x80\x00\x00\u07d4|`\xa0_zJ_\x8c\xf2xC\x916.uZ\x83A\xefY\x89f\x94
\xf0\x18*7\xae\x00\x00\u07d4|`\xe5\x1f\v\xe2(\xe4\xd5o\xdd)\x92\xc8\x14\xdaw@\u01bc\x89\n\u0
5ce\xbcZ\xc6 \x00\x00\u07d4|i$\xd0|>\xf5\x89\x19f\xfe\nxV\xc8{\xef\x9d
4\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94|\x8b\xb6Zo\xbbI\xbdA3\x96\xa9\xd7\xe3\x10S\xbb\xb3z\x
a9\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94|\x9a\x11\f\xb1\x1f%\x98\xb2\xb2\x0e,\xa4\x002^
A\xe9\xdb3\x8a\x05\x81v{\xa6\x18\x9c@\x00\x00\u07d4|\xbc\xa8\x8f\xcaj\x00`\xb9`\x98\\\x9a\xa1\
xb0%4\xdc\"\b\x89\x19\x12z\x13\x91\xea*\x00\x00\u07d4|\xbe\xb9\x992\xe9~n\x02\x05\x8c\xfcb\u
0432k\xc7\u0325+\x89lk\x93[\x8b\xbd@\x00\x00\u07d4|\xc2Jj\x95\x8c
\xc7\xd1$\x96`\xf7Xb&\x95\v\r\x9a\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4|\xd2\x0e\u0335\x18\xb6\
f\xab\t[r\x0fW\x15p\u02aaD~\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4|\xd5\xd8\x1e\xab7\xe1
\x1ebv\xa3\xa1\t\x12Q`~\r~8\x89\x03hM^\xf9\x81\xf4\x00\x00\u07d4|\xdft!9E\x95=\xb3\x9a\xd0\xe
8\xa9x\x1a\xddy.M\x1d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4|\xe4hdF\U000547be\xd6r\x15\xeb\rZ\
x1d\xd7,\x11\xb8\x89x9\xd3!\xb8\x1a\xb8\x00\x00\u07d4|\xefMC\xaaA\u007f\x9e\xf8\xb7\x87\xf8\
xb9\x9dS\xf1\xfe\xa1\ue209g\x8a\x93
b\xe4\x18\x00\x00\u07d4}\x03P\xe4\v3\x8d\xdasfa\x87+\xe3?\x1f\x97R\xd7U\x89\x02\xb4\xf5\xa6
\U00051500\x00\xe0\x94}\x04\xd2\xed\xc0X\xa1\xaf\xc7a\xd9\u025a\xe4\xfc\\\x85\xd4\u0226\x8a
B\xa9\xc4g\\\x94g\xd0\x00\x00\u07d4}\v%^\xfbW\xe1\x0fp\b\xaa\"\xd4\x0e\x97R\xdf\xcf\x03x\x89\
x01\x9f\x8euY\x92L\x00\x00\xe0\x94}\x13\xd6pX\x84\xab!W\u074d\xccpF\xca\xf5\x8e\xe9K\xe4\x
8a\x1d\r\xa0|\xbb>\xe9\xc0\x00\x00\u07d4}'>c~\xf1\xea\u0101\x11\x94\x13\xb9\x1c\x98\x9d\xc5\x
ea\xc1\"\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4}*R\xa7\xcf\f\x846\xa8\xe0\a\x97kl&\xb7\"\x
9d\x1e\x15\x89\x17\xbf\x06\xb3*$\x1c\x00\x00\u07d4}4\x805i\xe0\v\u05b5\x9f\xff\b\x1d\xfa\\\n\xb4
\x19zb\x89\\\xd8|\xb7\xb9\xfb\x86\x00\x00\u07d4}4\xffY\xae\x84\nt\x13\u01baL[\xb2\xba,u\xea\xb
0\x18\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4}9(R\xf3\xab\xd9/\xf4\xbb[\xb2l\xb6\bt\xf2\xbeg\x95\x
8966\xc2^f\xec\xe7\x00\x00\u07d4}DRg\u015a\xb8\u04a2\xd9\xe7\t\x99\x0e\th%\x80\u011f\x8965
\u026d\xc5\u07a0\x00\x00\xe0\x94}U\x13\x97\xf7\x9a)\x88\xb0d\xaf\xd0\xef\xeb\xee\x80,w!\xbc\x
8a\bW\xe0\xd6\xf1\xdav\xa0\x00\x00\u07d4}Z\xa3?\xc1KQ\x84\x1a\x06\x90n\xdb+\xb4\x9c*\x11ri\
x89\x10D\x00\xa2G\x0eh\x00\x00\xe0\x94}]/s\x94\x9d\xad\xda\bV\xb2\x06\x98\x9d\xf0\a\x8dQ\xa
1\xe5\x8a\x02<upr\xb8\xdd\x00\x00\x00\xe0\x94}n\x99\r\xaaq\x05\xde%&3\x983\xf7{\\\v\x85\xd8O
\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4}s\x8608\xcc\xca\"\xf9j\xff\xda\x10InQ\xe1\xe6\xcdH\x89
\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4}}\xd5\xeeaM\xbbo\xbf\xbc\xd2c\x05$z\x05\x8cA\xfa\xa1\x
89lk\x93[\x8b\xbd@\x00\x00\u07d4}~|aw\x9a\xdbw\x06\xc9M2@\x9a+\xb4\u953f`\x89.\xf2\r\x9f\xc

7\x1a\x14\x00\x00\u07d4}\x82\xe5#\xcc-
\u0151\xda9T\u8dbb,\xafda\u6709}\x8d\xc2\xef\xff\xb1\xa9\x00\x00\u07d4}\x85\x84\x93\xf0t\x15\x
e0\x91-
\x05y<\x97!\x13\xea\u8b88\x89b\x8d\xd1w\u04bc(\x00\x00\u07d4}\x90\x1b(\xbf\u007f\x88\xefs\xd
8\xf7<\u0297VI\x13\xea\x8a$\x893\xc5I\x901r\f\x00\x00\u07d4}\x98\x0fKVk\xb0EQ~L\x14\xc8wP\
u0793FtK\x89Hz\x9a0E9D\x00\x00\xe0\x94}\x9cYc\x1e+\xa2\xe8\xe8(\x91\xf3\x97\x99\"\xaa\xa3\
xb5g\xa1\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4}\x9d\"\x1a=\xf8\x9d\xdd{_a\xc1F\x8cg\x87\u
05b33\xe6\x89\a{\"|\xd8;\xe8\x00\x00\u07d4}\xa7a4E\xa2\x12\x99\xaat\xf0\xadqC\x1e\xc4?\xbb\x
1b\xe9\x89\x03\xaf\xb0\x87\xb8v\x90\x00\x00\u07d4}\xb4\xc7\u0577\x97\xe9)nc\x82\xf2\x03i=\xb
4\tD\x9db\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4}\xb9\xea\xccR\xe4)\u0703\xb4a\xc5\xf4\xd8
`\x10\xe58:(\x8965\u026d\xc5\u07a0\x00\x00\u07d4}\xd4m\xa6w\xe1a\x82^\x12\xe8\r\xc4F\xf5\x8
2v\xe1\x12|\x89,s\xc97t,P\x00\x00\u07d4}\xd8\u05e1\xa3O\xa1\xf8\xe7<\xcb\x00_\u00a0:\x15\xb8
\"\x9c\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4}\xddW\x16\\\x87\xa2p\u007f\x02]\xcf\xc2P\x8c\t\x83GY\xbc\x89K\xe4\xe7&{j\xe0\x
00\x00\u07d4}\xe4B\xc8#\x86\x15M.\x99<\xbd\x12\x80\xbb|\xa6\xb1*\u0689\xd8\xf2\xe8$~\xc9H\
x00\x00\u07d4}\xe7\xfeA\x9c\xc6\x1f\x91\xf4\b\xd24\u0300\xd5\xca=\x05M\x99\x89\x01\x15\x8eF\
t\x13\xd0\x00\x00\u07d4}\xec\u664a\xe1\x90\r\xd3w\f\xf4\xb98\x12\xba\xd8O\x03\"\x89\x05k\xc7^
-c\x10\x00\x00\u07d4}\xfc4-
\xff\xcfE\xdf\xeet\xf8L\t\x959{\u04661r\x89\r\x8drkqw\xa8\x00\x00\u07d4}\xfd)b\xb5u\xbc\xbe\xee\
x97\xf4\x91B\xd6<0\xab\x00\x9ff\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4~\x1e)r\x1dl\xb9\x10
W\xf6\xc4\x04-
\x8a\v\xbcdJ\xfes\x89\b\xa9\xab\xa5W\xe3l\x00\x00\u07d4~#ff\xb2\xd0nc\xeaN*\xb8CW\xe2\xdf\x
c9w\xe5\x0e\x8965f3\xeb\xd8\xea\x00\x00\xe0\x94~$\xd9\xe2,\xe1\xda<\xe1\x9f!\x9c\xce\xe5#7h
s\xf3g\x8a\x01?\xd9\a\x9c\xaa`\xff\x00\x00\u07d4~$\xfb\xda\u0490\x17^\xb2\xdfm\x18\n\x19\xb9\
xa9\xf4\x13p\xbe\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94~&\x8f\x13\x1d\xdfh|\xc3%\xc4\x12\xf7\
x8b\xa9a
^\x91\x12\x8a\x03cd\xee}0\x1b<\x00\x00\u07d4~))\x008I5Y\x19N\x94mNF\v\x96\xfc8\xa1V\x89\x
10\xc1<Rwc\x88\x00\x00\u07d4~+\xa8m\xa5.x]\x86%3O3\x97\xba\x1cK\xf2\xe8\u0449\n\xad\xec\
x98?\xcf\xf4\x00\x00\u07d4~?c\xe11)\xa2!\xba\x1a\xb0c&4,\u064bQ&\xae\x89V\xa0&Y\xa5#4\x00
\x00\u07d4~Gc~\x97\xc1F\"\x88+\xe0W\xbe\xa2)8o@R\xe5\x89\x17\xda:\x04\u01f3\xe0\x00\x00\
u07d4~N\x94\tpA!\xd1\xd7y\x97\x02o\xf0n\xa9\xb1\x9a\x8b\x90\x89\x8d\x16T\x9e\u054f\xa4\x00\
x00\u07d4~Y\xdc`\xbe\x8b/\xc1\x9a\xbd\nW\x82\xc5,(@\v\u0397\x8965\u026d\xc5\u07a0\x00\x00
\u07d4~[\x19\xae\x1b\xe9O\xf4\xde\xe65I*\x1b\x01-\x14\xdb\x02\x13\x89\x05k\xc7^-
c\x10\x00\x00\u07d4~]\x99\x93\x10NL\xb5E\xe1y\xa2\xa3\xf9q\xf7D\xf9\x84\x82\x89lk\x93[\x8b\x
bd@\x00\x00\u07d4~q\x17\x1f)I\xfa\f:\xc2T%K\x1f\x04@\xe5\xe6\xa08\x89\x02+\x1c\x8c\x12'\xa0
\x00\x00\u07d4~|\x1e\x9aa\xa0\x8a\x83\x98H5\xc7\x0e\xc3\x1d4\xd3\xea\xa8\u007f\x89\nZ\xa8P\
t\xe3\x9c\x00\x00\u07d4~\u007f\x18\xa0.\u032a]a\xab\x8f\xbf\x03\x03C\xc44\xa2^\xf7\x89\x03\x9f
\xba\xe8\xd0B\xdd\x00\x00\u07d4~\x81\xf6D\x9a\x037A\x91\xf3\xb7\xcb\x05\xd98\xb7.\t\r\xff\x89\
x05k\xc7^-
c\x10\x00\x00\u07d4~\x86I\xe6\x90\xfc\x8c\x1b\xfd\xa1\xb5\xe1\x86X\x1fd\x9bP\xfe3\x89\x05V\xf
6L\x1f\xe7\xfa\x00\x00\u07d4~\x87\x86>\xc4:H\x1d\xf0M\x01wb\xed\xcb\\\xaab\x9bZ\x89\x02\"\xc
8\xeb?\xf6d\x00\x00\u07d4~\x8f\x96\xcc)\xf5{\tu\x12\f\xb5\x93\xb7\u0743=`kS\x89\n\xad\xec\x98?

\xcf\xf4\x00\x00\u07d4~\x97*\x8a|*D\xc9;!Cl8\xd2\x1b\x92R\xc3E\xfe\x89a\t=|,m8\x00\x00\u07d4~\x99\u07fe\x98\x9d;\xa5)\u0457Q\xb7\xf41\u007f\x89S\xa3\xe2\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4~\xa0\xf9n\xe0\xa5s\xa30\xb5h\x97v\x1f=L\x010\xa8\xe3\x89Hz\x9a0E9D\x00\x00\u0794~\xa7\x91\xeb\xab\x04E\xa0\x0e\xfd\xfcNJ\x8e\x9a~ue\x13m\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4~\xab\xa05\xe2\xaf7\x93\xfdtgK\x10%@\xcf\x19\n\u0779\x89E\x02l\x83[`D\x00\x00\xe0\x94~\xb4\xb0\x18\\\x92\xb6C\x9a\b\xe72!h\xcb5<\x8awJ\x8a\x02'\x19l\xa0I\x83\xca\x00\x00\xe0\x94~\xbd\x95\xe9\xc4p\xf7(5\x83\xdcn\x9d,M\xce\v\ua3c4\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4~\u0425\xa8G\xbe\xf9\xa9\xda|\xba\x1dd\x11\xf5\xc3\x161&\x19\x89\x02(\xeb7\xe8u\x1d\x00\x00\u07d4~\xda\xfb\xa8\x98K\xafc\x1a\x82\vk\x92\xbb\xc2\xc56U\xf6\xbd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4~\xdb\x02\xc6\x1a\"r\x87a\x1a\xd9Pici\xccNdzh\x89\x0e\u0683\x8cI)\b\x00\x00\u07d4~\xe5\u0280]\xce#\xaf\x89\xc2\xd4D\xe7\xe4\af\xc5Lt\x04\x89\r\v\xd4\x12\xed\xbd\x82\x00\x00\xe0\x94~\xe6\x04\u01e9\xdc)\t\xce2\x1d\u6e72OWgWuU\x8a\x01+\xf9\u01d8\\\xf6-\x80\x00\u07d4~\xf1o\xd8\xd1[7\x8a\x0f\xba0k\x8d\x03\u0758\xfc\x92a\x9f\x89%\xf2s\x93=\xb5p\x00\x00\u07d4~\xf9\x8bR\xbe\xe9S\xbe\xf9\x92\xf3\x05\xfd\xa0'\xf8\x91\x1cXQ\x89\x1b\xe7\"i\x96\xbc\x80\x00\u07d4~\xfc\x90vj\x00\xbcR7,\xac\x97\xfa\xbd\x8a<\x83\x1f\x8e\u0349\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u07d4~\xfe\xc0\xc6%<\xaf9\u007fq(|\x1c\a\xf6\xc9X+[\x86\x89\x1a,\xbc\xb8O0\u0540\x00\u07d4\u007f\x01\xdc|7G\xca`\x8f\x98=\xfc\x8c\x9b9\xe7U\xa3\xb9\x14\x89\v8l\xad_zZ\x00\x00\u07d4\u007f\x06b\xb4\x10)\x8c\x99\xf3\x11\u04e1EJ\x1e\xed\xba/\xeav\x89\n\u05ce\xbcZ\xc6\x00\x00\xe0\x94\u007f\x06\u021dY\x80\u007f\xa6\v\xc6\x016\xfc\xf8\x14\u02ef%C\xbd\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u007f\v\x90\xa1\xfd\u050f'\xb2h\xfe\xb3\x83\x82\xe5]\xdbP\xef\x0f\x892\xf5\x1e\u06ea\xa30\x00\x00\u07d4\u007f\x0e\xc3\u06c0F\x92\xd4\xd1\xea2E6Z\xab\x05\x90\a[\u0109\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u007f\x0f\x04\xfc\xf3zS\xa4\xe2N\xden\x93\x10Nx\xbe\x1d<\x9e\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\u007f\x13\xd7`I\x8dq\x93\xcahY\xbc\x95\xc9\x018d#\xd7l\x8a\x01\x0f\f\xf0d\xddY\x00\x00\u07d4\u007f\x15\n\xfb\x1aw\u00b4Y(\xc2h\xc1\u9f74d\x1dG\u0609lk\x93[\x8b\xbd@\x00\x00\u07d4\u007f\x16\x19\x98\x8f7\x15\xe9O\xf1\xd2S&-\xc5X\x1d\xb3\xde\x1c\x890\xca\x02O\x98{\x90\x00\x00\u07d4\u007f\x1c\x81\xee\x16\x97\xfc\x14K|\v\xe5I;V\x15\xae\u007f\xdd\u0289\x1b\x1d\xaba\u04ead\x00\x00\u07d4\u007f#\x82\xff\xd8\xf89VFy7\xf9\xbar7F#\xf1\x1b8\x89\x86\xac5\x10R`\x00\x00\u07d4\u007f7\t9\x1f?\xbe\xba5\x92\xd1u\xc7@\xe8z\tT\x1d\x02\x89\x1a\x05V\x90\xd9\u06c0\x00\x00\u07d4\u007f8\x9c\x12\xf3\xc6\x16OdFVlwf\x95\x03\xc2y%'\x89\x05V\xf6L\x1f\xe7\xfa\x00\x00\xe0\x94\u007f:\x1eE\xf6~\x92\u0200\xe5s\xb43y\xd7\x1e\xe0\x89\xdbT\x8a\x15-\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94\u007f=r\x03\u0224G\xf7\xbf6\u060a\xe9\xb6\x06*^\xeex\xae\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\u007fF\xbb%F\r\xd7\xda\xe4!\x1c\xa7\xf1Z\xd3\x12\xfc}\xc7\\\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\u007fI\xe7\xa4&\x98\x82\xbd\x87\"\u0526\xf5f4v)b@y\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\u007fI\xf2\a&G\x1a\xc1\u01e8>\xf1\x06\xe9w\\\xebf%f\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d4\u007fK^'\x85x\xc0F\xcc\xea\xf6W0\xa0\xe0h2\x9e\u0576\x89e\xea=\xb7UF`\x00\x00\u07d4\u007fOY;a\x8c3\v\xa2\xc3\xd5\xf4\x1e\xce\xeb\x92\xe2~Bl\x89\x96n\xdcuk|\xfc\x00\x00\u07d4\u007fT\x14\x91\u04ac\x00\xd2a/\x94\xaa\u007f\v\xcb\x01FQ\xfb\u0509\x14b\fW\xdd\xda\xe0\x00\x00\u07d4\u007fZ\xe0Z\xe0\xf8\xcb\xe5\xdf\xe7!\xf0D\u

05e7\xbe\xf4\xc2y\x97\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\u007f`:\xec\x17Y\xea_\a\xc7\xf8\xd4\x1a\x14(\xfb\xba\xf9\xe7b\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u007falo\x00\x8a\u07e0\x82\xf3M\xa7\xd0e\x04`6\x80u\xfb\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u007fa\xfal\xf5\xf8\x98\xb4@\xda\u016b\xd8`\rmi\x1f\xde\xf9\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\xe0\x94\u007fe\\g\x89\xed\xdfE\\\xb4\xb8\x80\x99r\x0698\x9e\ubb0a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\u007fk(\u0204!\xe4\x85~E\x92\x81\u05c4ai$\x89\xd3\xfb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u007fn\xfboC\x18\x87m.\xe6$\xe2u\x95\xf4DF\xf6\x8e\x93\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\u007fq\x92\xc0\xdf\x1c}\xb6\xd9\xede\xd7\x11\x84\xd8\xe4\x15Z\x17\xba\x89\x04Sr\x8d3\x94,\x00\x00\u07d4\u007fz:!\xb3\xf5\xa6]\x81\xe0\xfc\xb7\xd5-
\xd0\n\x1a\xa3m\xba\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\u007f\x8d\xbc\xe1\x80\xed\x9cV65\xaa\xd2\xd9{L\xbcB\x89\x06\u0649\x90\xf54`\x8ar\x88\x00\x00\xe0\x94\u007f\x99=\xdb~\x02\u0082\xb8\x98\xf6\x15_h\x0e\xf5\xb9\xaf\xf9\a\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\u007f\x9f\x9bV\xe4(\x9d\xfbX\xe7\x0f\xd5\xf1*\x97\xb5m5\u01a5\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\u007f\xa3~\xd6x\x87u\x1aG\x1f\x0e\xb3\x06\xbeD\xe0\xdb\xcd`\x89\x899vt\u007f\xe1\x1a\x10\x00\x00\u07d4\u007f\xaa0\xc3\x15\x19\xb5\x84\xe9rP\xed*<\xf38^\xd5\xfdP\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u007f\xcf[\xa6fo\x96lTH\xc1{\xf1\xcb\v\xbc\xd8\x01\x9b\x06\x89\x05k\xc3\u042e\xbeI\x80\x00\xe0\x94\u007f\xd6y\xe5\xfb\r\xa2\xa5\xd1\x16\x19M\xcbP\x83\x18\xed\u0140\xf3\x8a\x01c\x9eI\xbb\xa1b\x80\x00\x00\u07d4\u007f\u06e01\u01cf\x9c\tmb\xd0Z6\x9e\uac3c\xccU\u5257\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\u007f\xdb\u00e8D\xe4\r\x96\xb2\xf3\xa652.`e\xf4\xca\x0e\x84\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u007f\xdf\u020dx\xbf\x1b(Z\xc6O\x1a\xdb5\xdc\x11\xfc\xb09Q\x89|\x06\xfd\xa0/\xb06\x00\x00\u07d4\u007f\xea\x19b\xe3]b\x05\x97h\xc7I\xbe\u0756\u02b90\xd3x\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u007f\xef\x8c8w\x9f\xb3\a\xeco\x04K\xeb\xe4\u007f<\xfa\xe7\x96\xf1\x89\t#@\xf8l\xf0\x9e\x80\x00\u07d4\u007f\xf0\xc6?p$\x1b\xec\xe1\x9bs~SA\xb1+\x10\x901\u0609\x12\xc1\xb6\xee\xd0=(\x00\x00\xe0\x94\u007f\xfa\xbf\xbc9\f\xbeC\u0389\x18\x8f\bh\xb2}\xcb\x0f\f\xad\x8a\x01YQ\x82\"K&H\x00\x00\xe0\x94\u007f\xfd\x02\xed7\fp`\xb2\xaeS\xc0x\xc8\x01!\x90\u07fbu\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u0794\x80\x02*\x12\a\xe9\x10\x91\x1f\xc9(I\xb0i\xab\f\xda\xd0C\u04c8\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\x80\t\xa7\xcb\u0452\xb3\xae\u052d\xb9\x83\xd5(ER\xc1ltQ\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x80\x0e}c\x1cnW:\x903/\x17\xf7\x1f_\u045bR\x8c\xb9\x89\b=lz\xabc`\x00\x00\u07d4\x80\x15m\x10\ufa320\u0254\x10c\r7\xe2i\xd4\t<\xea\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x80\x172\xa4\x81\u00c0\xe5~\xd6-l)\u0799\x8a\xf3\xfa;\x13\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x80\x1de\xc5\x18\xb1\x1d\x0e?OG\x02!Ap\x13\xc8\xe5>\u0149\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x80&CZ\xacr\x8dI{\x19\xb3\xe7\xe5|(\xc5c\x95O+\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4\x80-\xc3\xc4\xff-
}\x92^\u215fJ\x06\u05fa`\xf10\x8c\x89\x05P\x94\f\x8f\xd3L\x00\x00\u07d4\x800\xb1\x11\u0198?\x04\x85\u076c\xa7b$\xc6\x18\x064x\x9f\x89\x04V9\x18$O@\x00\x00\u07d4\x805\xbc\xff\xae\xfd\xee\xea5\x83\fI}\x14(\x9d6
#\u0789\x10CV\x1a\x88)0\x00\x00\u07d4\x805\xfeNkj\xf2z\u44a5xQ^\x9d9\xfao\xa6[\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x80C\xed\"\xf9\x97\u58a4\xc1n6D\x86\xaed\x97V\x92\u0109=I\x04\xff\xc9\x11.\x80\x00\u07d4\x80C\xfd\u043cL\x97=\x16c\xd5_\xc15P\x8e\xc5\xd4\xf4\xfa\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x80L\xa9IrcOc:Q\xf3V\v\x1d\x06\xc0\xb2\x93\xb3\xb1\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\x80R-

\u07d4N\xc5.'\xd7$\xedL\x93\xe1\xf7\xbe`\x83\u0589\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x80Y\x1aB\x17\x9f4\xe6M\x9d\xf7]\xcdF;(hoUt\x8a\x04<3\xc1\x93ud\x80\x00\x0
0\u07d4\x80\\\xe5\x12\x97\xa0y;\x81 g\xf0\x17\xb3\xe7\xb2\u07db\xb1\xf9\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\x80]\x84o\xb0\xbc\x02\xa73r&\u0585\xbe\x9e\xe7s\xb9\x19\x8a\x8a\x04<0
\xfb\b\x84\xa9l\x00\x00\u07d4\x80c7\x9a{\xf2\u02d2:\x84\xc5\t>h\xda\xc7\xf7T\x81\u0149\x11v\x1
0.n2\xdf\x00\x00\u07d4\x80hTX\x8e\xcc\xe5AI_\x81\u008a)\x03s\xdf\x02t\xb2\x89\x1f\x8c\xdf\\n\x
8dX\x00\x00\u07d4\x80oD\xbd\xebh\x807\x01^\x84\xff!\x80I\xe3\x823*3\x89l]\xb2\xa4\xd8\x15\xd
c\x00\x00\u07d4\x80tF\x18\xde9jT1\x97\xeeH\x94\xab\xd0c\x98\xdd|'\x89lk\x93[\x8b\xbd@\x00\x0
0\u07d4\x80w\xc3\xe4\xc4EXn\tL\xe1\x02\x93\u007f\xa0[s{V\x8c\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x80\x90\u007fY1H\xb5|F\xc1w\xe2=%\xab\u012a\xe1\x83a\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x80\x97s\x16\x94NYB\xe7\x9b\x0e:\xba\u04cd\xa7F\be\x19\x89\x02\x1auJm
\xc5(\x00\x00\xe0\x94\x80\xa0\xf6\xcc\x18l\xf6
\x14\x00sn\x06Z9\x1fR\xa9\xdfJ\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x80\xab\xecZ\x
a3n\\\x9d\t\x8f\x1b\x94(\x81\xbdZ\xca\u0196=\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x80\xb2=8\v\x
82\\F\xe098\x99\xa8UVF-\xa0\u1309lk\x93[\x8b\xbd@\x00\x00\u07d4\x80\xb4-
\xe1p\xdb\xd7#\xf4T\xe8\x8fw\x16E-\x92\x98P\x92\x89\x10F#\xc0v-
\xd1\x00\x00\u07d4\x80\xb7\x9f3\x83\x90\u047a\x1b77\xa2\x9a\x02W\xe5\xd9\x1e\a1\x89\x01\x1
5\x8eF\t\x13\xd0\x00\x00\u07d4\x80\xbf\x99^\u063a\x92p\x1d\x10\xfe\u011f\x9e}\x01M\xbe\xe0&\
x89\x1f\x047\xca\x1a~\x12\x80\x00\u07d4\x80\xc0N\xfd1\x0fD\x04\x83\xc7?tK[\x9edY\x9c\xe3\xe
c\x89A\rXj
\xa4\xc0\x00\x00\u07d4\x80\u00e9\xf6\x95\xb1m\xb1Yr\x86\u0473\xa8\xb7il9\xfa'\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x80\xc5>\xe7\xe35\u007f\x94\xce\rxh\x00\x9c
\x8bJ\x13\x01%\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x80\xcc!\xbd\x99\xf3\x90\x05\u014f\xe4\xa4
H\x90\x92
!\x8ff\u02c966\xc9yd6t\x00\x00\u07d4\x80\xd5\xc4\fY\xc7\xf5N\xa3\xa5_\xcf\xd1uG\x1e\xa3P\x99\
xb3\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x80\xda/\u0762\x9a\x9e'\xf9\xe1\x15\x97^i\xae\x9c\xf
b\xf3\xf2~\x89\n\u05ce\xbcZ\xc6 \x00\x00\xe0\x94\x80\xe7\xb3
R0\xa5f\xa1\xf0a\xd9\"\x81\x9b\xb4\xd4\u04a0\xe1\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4\
x80\xea\x1a\xcc\x13n\xcaKh\xc8B\xa9Z\xdfk\u007f\xee~\xb8\xa2\x89\xd8\xd7&\xb7\x17z\x80\x00\
x00\u07d4\x80\xf0z\xc0\x9e{,<\n=\x1e\x94\x13\xa5D\xc7:A\xbe\u02c9\x01\x15\x8eF\t\x13\xd0\x00
\x00\u07d4\x81\r\xb2Vu\xf4^\xa4\xc7\xf3\x17\u007f7\xce)\xe2-g\x99\x9c\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x81\x13\x9b\xfd\u0326V\xc40
?r\x95\x8cT;e\x80\xd4\f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x81\x14a\xa2\xb0\u0290\xba\xda\xc
0j\x9e\xa1nx{3\xb1\x96\u0309\b\xe3\xf5\v\x17<\x10\x00\x00\u07d4\x81\x16M\xeb\x10\x81J\xe0\x
83\x91\xf3,\bf{bH\xc2}z\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\x81\x18i1\x18A7\xd1\x19*\u0
20c\xd3\xe1\xe5\xd0\xfd\xb8jt\x89\x9d5\x95\xab$8\xd0\x00\x00\u0794\x81*U\xc4<\xae\xdcYr\x18
7\x90\x00\xceQ\rT\x886\xfd\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x81.\xa7\xa3\xb2\xc8n\xed2\
xffO,sQL\xc6;\xac\xfb\u038965\u026d\xc5\u07a0\x00\x00\u07d4\x814\xdd\x1c\x9d\xf0\xd6\u0225\
x81$&\xbbU\xc7a\u0283\x1f\b\x89\x06\xa2\x16\v\xb5|\xcc\x00\x00\u07d4\x81A5\u068f\x98\x11\a
W\x83\xbf\x1a\xb6pb\xaf\x8d>\x9f@\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x81I\x8c\xa0{\x0f/
\x17\xe8\xbb\xc7\xe6\x1a\u007fJ\xe7\xbef\xb7\x8b\x89\x05\x81\xfb\xb5\xb3;\xb0\x00\x00\u07d4\x
81Um\xb2sI\xab\x8b'\x00ID\xedP\xa4n\x94\x1a\x0f_\x89\u063beI\xb0+\xb8\x00\x00\u07d4\x81U\x

falQ\xeb1\xd8\bA-
t\x8a\xa0\x86\x10P\x18\x12/\x89e\xea=\xb7UF`\x00\x00\xe0\x94\x81V6\v\xbd7\ta\xce\xcakf\x91\xd7P\x06\xad
L\xf2\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\x81a\xd9@\xc3v\x01\x00\xb9\b\x05)\xf8\xa6\x03%\x03\x0fn\u0709\x10CV\x1a\x88)0\x00\x00\xe0\x94\x81d\xe7\x83\x14\xae\x16\xb2\x89&\xccU=,\xcb\x16\xf3V'\r\x8a\x01\xca\x13N\x95\xfb2\xc8\x00\x00\u07d4\x81e\u02b0\xea\xfbZ2\x8f\xc4\x1a\xc6M\xaeq[.\xef,e\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x81h\xed\xce\u007f)a\xcf)[\x9f\xcdZE\xc0l\xde\xdan\xf5\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x81m\x97r\xcf\x119\x91\x16\xcc\x1er\xc2lgt\xc9\xed\xd79\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x81s\xc85djg.\x01R\xbe\x10\xff\xe8Ab\xdd%nL\x89\x1a\xab\xdf!E\xb40\x00\x00\u07d4\x81t\x93\u035b\xc6#p*$\xa5o\x9f\x82\xe3\xfdH\xf3\xcd1\x89\x9eK#\xf1-
L\xa0\x00\x00\u07d4\x81y\xc8\tp\x18,\u0177\xd8*M\xf0n\xa9M\xb6:%\xf3\x89'o%\x9d\xe6k\xf4\x00\x00\u07d4\x81z\xc3;\xd8\xf8GVsr\x95\x1fJ\x10\u05e9\x1c\xe3\xf40\x89\n\xd7\xc4\x06\xc6m\xc1\x80\x00\xe0\x94\x81\x8f\xfe'\x1f\u00d75e\xc3\x03\xf2\x13\xf6\xd2\u0689\x89~\xbd\x8a\x016\xe0SB\xfe\u1e40\x00\u07d4\x81\x97\x94\x81!s.c\xd9\xc1H\x19N\xca\xd4n0\xb7I\u0209\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x81\x9a\xf9\xa1\xc2s2\xb1\xc3i\xbb\xda\x1b=\xe1\xc6\xe93\xd6@\x89\x11\t\xe6T\xb9\x8fz\x00\x00\xe0\x94\x81\x9c\u06a506x\xef|\xecY\u050c\x82\x16:\xcc`\xb9R\x8a\x03\x13QT_y\x81l\x00\x00\u07d4\x81\x9e\xb4\x99\vZ\xbaUG\t=\xa1+k<\x10\x93\xdfmF\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x81\xa8\x81\x96\xfa\xc5\xf2<>\x12\xa6\x9d\xecK\x88\x0e\xb7\xd9s\x10\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x81\xbc\xcb\xff\x8fD4~\xb7\xfc\xa9['\xce|\x95$\x92\xaa\xad\x89\b@\xc1!e\xddx\x00\x00\u07d4\x81\xbdu\xab\xd8e\xe0\xc3\xf0J\vO\xdb\xcbt\xd3@\x82\xfb\xb7\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x81\xc1\x8c*#\x8d\xdcL\xba#\n\a-
\xd7\xdc\x10\x1eb\x02s\x89Hz\x9a0E9D\x00\x00\u07d4\x81\xc9\xe1\xae\xe2\xd36]S\xbc\xfd\u0356\xc7\xc58\xb0\xfd~\xec\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\x81\u03edv\t\x13\xd3\xc3\"\xfc\xc7{I\u00ae9\a\xe7On\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94\x81\xd6\x19\xffW&\xf2@_\x12\x90Lr\xeb\x1e$\xa0\xaa\xeeO\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x81\xef\u25aev\xc8`\xd1\xc5\xfb\xd3=G\xe8\u0399\x96\xd1W\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x81\xf8\xde,(=_\u052f\xbd\xa8]\xed\xf9v\x0e\xab\xbb\xb5r\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x82\f\x19)\x11\x96P[e\x05\x9d\x99\x14\xb7\t\v\xe1\u06c7\u0789\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4\x82\x1c\xb5\xcd\x05\xc7\uf41f\xe1\xbe`s=\x89c\xd7`\xdcA\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x82\x1dy\x8a\xf1\x99\x89\u00ee[\x84\xa7\xa7(<\xd7\xfd\xa1\xfa\xbe\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\x82\x1e\xb9\t\x94\xa2\xfb\xf9K\xdc23\x91\x02\x96\xf7o\x9b\xf6\xe7\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x82$\x9f\xe7\x0fa\u01b1o\x19\xa3$\x84\x0f\xdc\x02\x021\xbb\x02\x8a\x02\x036\xb0\x8a\x93c[\x00\x00\u07d4\x82(\xeb\xc0\x87H\x0f\xd6EG\xca(\x1f^\xac\xe3\x04\x14S\xb9\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94\x82)\u03b9\xf0\xd7\b9I\x8dD\xe6\xab\xed\x93\xc5\xca\x05\x9f]\x8a\x1a\x1c\x1b<\x98\x9a
\x10\x00\x00\u07d4\x82.\xdf\xf66V:a\x06\xe5.\x9a%\x98\xf7\xe6\xd0\xef'\x82\x89\x01\xf4\xf9i=B\u04c0\x00\u07d4\x822\x19\xa2Yv\xbb*\xa4\xaf\x8b\xadA\xac5&\xb4\x936\x1f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x822\xd1\xf9t.\u07cd\xd9'\xda5;*\xe7\xb4\xcb\xceu\x92\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\x824\xf4c\u0444\x85P\x1f\x8f\x85\xac\xe4\x97,\x9bc-
\xbc\u0309lk\x93[\x8b\xbd@\x00\x00\u07d4\x827htg7\xcem\xa3\x12\xd5>TSN\x10o\x96|\xf3\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x82;\xa7dr8\xd1\x13\xbc\xe9\x96JC\u0420\x98\x11\x8b\xf

eM\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x82@t1(\x06\xdaGHCBf\xee\x00!@\u305a\u0089Q\xb1\u04c3\x92a\xac\x00\x00\
u07d4\x82C\x8f\u04b3*\x9b\xddgKI\xd8\xcc_\xa2\xef\xf9x\x18G\x89\x01\x15\x8eF\t\x13\xd0\x00\x
00\u07d4\x82HW(\xd0\xe2\x81V7X\xc7Z\xb2~\xd9\u80a0\x00-
\x89\a\xf8\b\xe9)\x1el\x00\x00\u07d4\x82K<<D>\x19)]~\xf6\xfa\xa7\xf3t\xa4y\x84\x86\xa8\x89\x01\
x15\x8eF\t\x13\xd0\x00\x00\u07d4\x82Q5\x8c\xa4\xe0`\u0775Y\xcaX\xbc\v\u077e\xb4\a\x02\x03\
x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x82Q5\xb1\xa7\xfc\x16\x05aL\x8a\xa4\u042cm\xba\u040fH\x
0e\x89M\x85<\x8f\x89\b\x98\x00\x00\u07d4\x82S\t\xa7\xd4]\x18\x12\xf5\x1en\x8d\xf5\xa7\xb9ol\x
90\x88\x87\x89\x804\xf7\u0671f\xd4\x00\x00\u07d4\x82Z\u007fN\x10\x94\x9c\xb6\xf8\x96Bh\xf1\x
fa_W\xe7\x12\xb4\u0109\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x82a\xfa#\f\x90\x1dC\xffW\x9fG\
x80\u04d9\xf3\x1e`v\xbc\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x82b\x16\x9baXp\x13N\xb4\xacl_G\
x1ck\xf2\xf7\x89\xfc\x89\x19\x12z\x13\x91\xea*\x00\x00\u07d4\x82c\xec\xe5\xd7\t\xe0\u05eeq\u0
328h\xed7\xcd/\xef\x80{\x895\xab\x02\x8a\xc1T\xb8\x00\x00\xe0\x94\x82l\xe5y\x052\xe0T\x8ca\x
02\xa3\r>\xac\x83k\xd68\x8f\x8a\x03\xcf\xc8.7\xe9\xa7@\x00\x00\u07d4\x82n\xb7\xcds\x19\xb8-
\xd0z\x1f;@\x90q\xd9n9g\u007f\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x82u1\xa6\u0141z\xe3_\x
82\xb0\v\x97T\xfc\xf7LU\xe22\x89\xc3(\t>a\xee@\x00\x00\u0794\x82u\xcdhL6y\u0548}\x03fN3\x8
3E\xdc<\xdd\xe1\x88\xdbD\xe0I\xbb,\x00\x00\u07d4\x82\x84\x92;b\u62ff|+\x9f4\x14\xd1>\xf6\xc8\
x12\xa9\x04\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\x82\x8b\xa6Q\u02d3\x0e\xd9xqV)\x9a
=\xe4L\u040br\x12\x89Hz\x9a0E9D\x00\x00\u07d4\x82\xa1\\\xef\x1dl\x82`\xea\xf1Y\xea?\x01\x80
\xd8g}\xce\x1c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x82\xa8\xb9kl\x9e\x13\xeb\xec\x1e\x9f\x18\xa
c\x02\xa6\x0e\xa8\x8aH\xff\x89lk\x8c@\x8es\xb3\x00\x00\u07d4\x82\xa8\u02ff\xdf\xf0+.8\xaeK\xb
f\xca\x15\xf1\xf0\xe8;\x1a\xea\x89\x04\x9b\x99\x1c'\xefm\x80\x00\u07d4\x82\xe4F\x1e\xb9\xd8I\xf
0\x04\x1c\x14\x04!\x9eBr\u0110\n\xb4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x82\xe5w\xb5\x15\xcb
+\b`\xaa\xfe\x1c\xe0\x9aY\xe0\x9f\xe7\xd0@\x89
\x86\xac5\x10R`\x00\x00\u07d4\x82\xea\x01\xe3\xbf.\x83\x83nqpN\"\xa2q\x93w\xef\xd9\u00c9\xa
4\xccy\x95c\u00c0\x00\x00\u07d4\x82\xf2\xe9\x91\xfd2L_]\x17v\x8e\x9fa3]\xb61\x9dl\x89\x1b\x1a
\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\x82\xf3\x9b'X\xaeB'{\x86\u059fu\xe6(\xd9X\xeb\u02b0\x8a\bxg
\x83&\xea\xc9\x00\x00\x00\xe0\x94\x82\xf8T\xc9\xc2\xf0\x87\xdf\xfa\x98Z\xc8
\x1ebl\xa5Fv\x86\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\x82\xffqo\xdf\x03>\xc7\xe9B\xc9\t\u0643\x18g\xb8\xb6\xe2\
xef\x89a\t=|,m8\x00\x00\u07d4\x83\b\xed\n\xf7\xf8\xa3\xc1u\x1f\xaf\xc8w\xb5\xa4*\xf7\xd3X\x82\x
8965\u026d\xc5\u07a0\x00\x00\u07d4\x83\x1cD\xb3\b@G\x18K*\xd2\x18h\x06@\x907P\xc4]\x89j
\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x83!\x05\x83\xc1jN\x1e\x1d\xac\x84\xeb\xd3~=\x0f|W\ub909lk\x
93[\x8b\xbd@\x00\x00\u07d4\x83,T\x17k\xdfC\xd2\u027c\u05f8\b\xb8\x95V\xb8\x9c\xbf1\x89\n\u0
5ce\xbcZ\xc6
\x00\x00\u07d4\x833\x16\x98]Gt+\xfe\xd4\x10`J\x91\x95<\x05\xfb\x12\xb0\x89lk\x93[\x8b\xbd@\x
00\x00\u07d4\x834vK{9zNW\x8fP6M`\xceD\x89\x9b\xff\x94\x89\x05\x03\xb2\x03\xe9\xfb\xa2\x00\
x00\xe0\x94\x83;j\x8e\xc8\xda@\x81\x86\xac\x8a}*m\xd6\x15#\xe7\u0384\x8a\x03c\\\x9a\xdc]\xea
\x00\x00\x00\u07d4\x83=?\xaeT*\xd5\xf8\xb5\f\xe1\x9b\xde+\xecW\x91\x80\u020c\x89\x12\xc1\xb
6\xee\xd0=(\x00\x00\xe0\x94\x83=\xb4,\x14\x16<{\xe4\u02b8j\u0153\xe0bf\u0599\u054a$\xe4\r+i
C\xef\x90\x00\x00\xe0\x94\x83V;\xc3d\ud060\xc6\xda;V\xffI\xbb\xf2g\x82z\x9c\x8a\x03\xab\x91\xd
1{ \xdeP\x00\x00\u07d4\x83zd]\xc9\\IT\x9f\x89\x9cN\x8b\u03c7S$\xb2\xf5|\x89

\x8c9J\xf1\u0208\x00\x00\u07d4\x83\x8b\xd5e\xf9\x9f\xdeH\x05?y\x17\xfe3<\xf8J\xd5H\xab\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x83\x90\x8a\xa7G\x8am\x1c\x9b\x9b\x02\x81\x14\x8f\x8f\x9f$+\x9f\u0709lk\x93[\x8b\xbd@\x00\x00\u07d4\x83\x92\xe57vq5x\x01[\xffI@\xcfC\x84\x9d}\u02e1\x89\bM\xf05]V\x17\x00\x00\xe0\x94\x83\x97\xa1\xbcG\xac\xd6GA\x81Y\xb9\x9c\xeaW\xe1\xe6S-
n\x8a\x01\xf1\x0f\xa8'\xb5P\xb4\x00\x00\u07d4\x83\x98\xe0~\xbc\xb4\xf7_\xf2\x11m\xe7|\x1c*\x99\xf3\x03\xa4\u03c9\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x83\xa3\x14\x883\xd9dI\x84\xf7\xc4u\xa7\x85\a\x16\ufd00\xff\x89\xb8Pz\x82\a(
\x00\x00\u07d4\x83\xa4\x02C\x8e\x05\x19w=TH2k\xfba\xf8\xb2\f\xf5-
\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d4\x83\xa9;[\xa4\x1b\xf8\x87
\xe4\x15y\f\xdc\vg\xb4\xaf4\u0109\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x83\xc2=\x8aP!$\xee\x15\x0f\b\xd7\x1d\xc6rt\x10\xa0\xf9\x01\x8a\a3\x1f;\xfef\x1b\x18\x00\x00\u07d4\x83\u0217\xa8Ki^\xeb\xe4fy\xf7\xda\x19\xd7vb\x1c&\x94\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x83\xd52\u04cdm\xee?`\xad\u018b\x93a3\u01e2\xa1\xb0\u0749\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x83\xdb\xf8\xa1(S\xb4\n\xc6\x19\x96\xf8\xbf\x1d\xc8\xfd\xba\xdd\xd3)\x894\x95tD\xb8@\xe8\x00\x00\u07d4\x83\xdb\xfd\x8e\xda\x01\xd0\u078e\x15\x8b\x16\u0413_\xc28\n]\u01c9
\x86\xac5\x10R`\x00\x00\u07d4\x83\xe4\x80U2|(\xb5\x93o\xd9\xf4D~s\xbd\xb2\xdd3v\x89\x90\xf54`\x8ar\x88\x00\x00\xe0\x94\x83\xfeZ\x1b2\x8b\xaeD\a\x11\xbe\xafj\xad`&\xed\xa6\xd2
\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x84\x00\x8ar\xf8\x03o?\xeb\xa5B\xe3Px\xc0W\xf3*\x88%\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x84\x0e\xc8>\xa96!\xf04\xe7\xbb7b\xbb\x8e)\xde\xd4\xc4y\x89\x87\x86x2n\xac\x90\x00\x00\xe0\x94\x84\x11E\xb4H@\xc9F\xe2\x1d\xbc\x19\x02d\xb8\xe0\xd5\x02\x93i\x8a?\x87\bW\xa3\xe0\xe3\x80\x00\x00\u07d4\x84#!\a\x93+\x12\xe01\x86X5%\xce\x02:p>\xf8\u0649lk\x93[\x8b\xbd@\x00\x00\u07d4\x84$O\xc9ZiW\xed|\x15\x04\xe4\x9f0\xb8\xc3^\xcaKy\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x841'}{\xdd\x10E}\xc0\x17@\x8c\x8d\xbb\xbdAJ\x8d\xf3\x89\x02\"\xc8\xeb?\xf6d\x00\x00\u07d4\x847Z\xfb\xf5\x9b:\x1da\xa1\xbe2\xd0u\xe0\xe1ZO\xbc\xa5\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x84;\xd3P/E\xf8\xbcM\xa3p\xb3#\xbd\xac?\xcf_\x19\xa6\x89P\x03\x9dc\xd1\x1c\x90\x00\x00\u07d4\x84P34c\rw\xf7AG\xf6\x8b.\bf\x13\xc8\xf1\xad\xe9\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4\x84R\x03u\x0fqH\xa9\xaa&)!\xe8mC\xbfd\x19t\xfd\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x84a\xec\u0126\xa4^\xb1\xa5\xb9G\xfb\x86\xb8\x80i\xb9\x1f\xcdo\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x84g^\x91wrmE\xea\xa4k9\x92\xa3@\xba\u007fq\f\x95\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x84hl{\xadv,T\xb6g\u055f\x90\x94<\xd1M\x11z&\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x84\x89\xf6\xad\x1d\x9a\x94\xa2\x97x\x91V\x89\x9d\xb6AT\xf1\u06f5\x89\x13t\a\xc0<\x8c&\x80\x00\u07d4\x84\x8c\x99Jy\x00?\xe7\xb7\xc2l\xc62\x12\xe1\xfc/\x9c\x19\xeb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x84\x8f\xbd)\xd6|\xf4\xa0\x13\xcb\x02\xa4\xb1v\xef$N\x9e\u6349\x01\x17*ck\xbd\xc2\x00\x00\u07d4\x84\x94\x9d\xbaU\x9ac\xbf\xc8E\xde\xd0n\x9f-
\x9b\u007f\x11\xef$\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x84\x9a\xb8\a\x90\xb2\x8f\xf1\xff\u05ba9N\xfctc\x10\\6\xf7\x89\x01\xe0+\xe4\xael\x84\x00\x00\u07d4\x84\x9b\x11oYc\x01\xc5\u063bb\xe0\xe9z\x82H\x12n9\xf3\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x84\xa7L\xee\xcf\xf6\\\xb9;/\x94\x9dw>\xf1\xad\u007f\xb4\xa2E\x89\x05\n\x9bDF\x85\xc7\x00\x00\u07d4\x84\xaa\xc7\xfa\x19\u007f\xf

8\\0\xe0;zS\x82\xb9W\xf4\x1f:\xfb\x89\b\x8b#\xac\xff\u0650\x00\x00\u07d4\x84\xaf\x1b\x15sB\xd5
Ch&\r\x17\x87b0\xa54\xb5K\x0e\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\x84\xb0\xeek\xb87\u04e4
\xc4\xc5\x01\x1c:\"\x8c\x0e\u06b4cJ\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x84\xb4\xb7Nf#\x
ba\x9d\x15\x83\xe0\u03feId?\x168AI\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x84\xb6\xb6\xad\
xbe/[>-
h,f\xaf\x1b\u0110S@\xc3\xed\x89!\x92\xf8\xd2\"\x15\x00\x80\x00\xe0\x94\x84\xb9\x1e.)\x02\xd0^
+Y\x1bA\b;\u05fe\xb2\xd5,t\x8a\x02\x15\xe5\x12\x8bE\x04d\x80\x00\u07d4\x84\xbc\xbf\"\xc0\x96\
a\xac\x844\x1d.\xdb\xc0;\xfb\x179\xd7D\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x84\xbf\xce
\xf0I\x1a\n\xe0iK7\u03ac\x02E\x84\xf2\xaa\x04g\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\x84\xcb}\xa0
P-
\xf4\\\xf5a\x81{\xbd#b\xf4Q\xbe\x02\u0689Hz\x9a0E9D\x00\x00\u07d4\x84\xccxx\xda`_\xdb\x01\x
9f\xab\x9bL\xcf\xc1Wp\x9c\u0765\x89Hy\x85\x13\xaf\x04\xc9\x00\x00\u07d4\x84\xdb\x14Y\xbb\x0
0\x81.\xa6~\xcb=\xc1\x89\xb7!\x87\xd9\xc5\x01\x89\b\x11\xb8\xfb\u0685\xab\x80\x00\u07d4\x84\
u9516\x80\xbe\xcehA\xb9\xa7\xe5%\r\b\xac\xd8}\x16\u0349\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x84\xe9\u03c1f\xc3j\xbf\xa4\x90S\xb7\xa1\xad@6
&\x81\xef\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x84\xec\x06\xf2G\x00\xfeBAL\xb9\x89|\x15L\x88\x
de/a2\x89Hz\x9a0E9D\x00\x00\xe0\x94\x84\xf5\"\xf0R\x0e\xbaR\xdd\x18\xad!\xfaK\x82\x9f+\x89\u
02d7\x8a\x01\fQ\x06\xd5\x13O\x13\x00\x00\u07d4\x85\v\x9d\xb1\x8f\xf8K\xf0\xc7\xdaI\xea7\x81\x
d9
\x90\xad~d\x89\x8c\xf2?\x90\x9c\x0f\xa0\x00\x00\u07d4\x85\x10\xee\x93O\f\xbc\x90\x0e\x10\a\xe
b8\xa2\x1e*Q\x01\xb8\xb2\x89\x05\xbf\v\xa6cOh\x00\x00\u07d4\x85\x16\xfc\xafw\u0213\x97\x0f\x
cd\x1a\x95\x8b\xa9\xa0\x0eI\x04@\x19\x89\n\xa3\xeb\x16\x91\xbc\xe5\x80\x00\u07d4\x85\x1a\xa
9\x1c\x82\xf4/\xad]\xd8\xe8\xbb^\xa6\x9c\x8f:Yw\u0449\b\x0eV\x1f%xy\x80\x00\u07d4\x85\x1c\rb\
xbeF5\xd4w~\x805\xe3~K\xa8Q|a2\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x85\x1d\u00ca\
xdbE\x93r\x9av\xf3:\x86\x16\u06b6\xf5\xf5\x9aw\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x852I\b\x97\xbb\xb4\u038b\u007fk\x83~L\xba\x84\x8f\xbe\x99v\x89\x05k\xc
7^-
c\x10\x00\x00\u07d4\x85>j\xba\xf4Di\xc7/\x15\x1dN\"8\x19\xac\xedN7(\x89lk\x93[\x8b\xbd@\x00\
x00\u07d4\x85F\x91\xceqO2\\\xedU\xceY(\u039b\xa1/\xac\u0478\x89\xedp\xb5\xe9\xc3\xf2\xf0\x0
0\x00\u07d4\x85L\fF\x9c$k\x83\xb5\u0473\xec\xa4C\xb3\x9a\xf5\xee\x12\x8a\x89V\xbcu\xe2\xd6
1\x00\x00\x00\u07d4\x85]\x9a\xef,9\xc6#\r\t\u025e\xf6II\x89\xab\u61c5\x89\b\xbaR\xe6\xfcE\xe4\x
00\x00\u07d4\x85c\u0113a\xb6%\xe7hw\x1c\x96\x15\x1d\xbf\xbd\x1c\x90iv\x89lk\x93[\x8b\xbd@\
x00\x00\u07d4\x85fa\t\x01\xaa\xce8\xb82D\xf3\xa9\xc810jg\xb9\u0709\xb0\x82\x13\xbc\xf8\xff\xe
0\x00\x00\xe0\x94\x85j\xa2<\x82\xd7![\xec\x8dW\xf6\n\xd7^\xf1O\xa3_D\x8a\x04<3\xc1\x93ud\x8
0\x00\x00\u07d4\x85nZ\xb3\xf6L\x9a\xb5k\x00\x93\x93\xb0\x16d\xfc\x03$\x05\x0e\x89a\t=|,m8\x0
0\x00\u07d4\x85n\xb2\x04$\x1a\x87\x83\x0f\xb2)\x03\x13C\xdc0\x85OX\x1a\x8965\u026d\xc5\u0
7a0\x00\x00\u07d4\x85s,\x06\\\xbdd\x11\x99A\xae\xd40\xacYg\vlQ\u0109'\xa5sb\xab\n\x0e\x80\x
00\xe0\x94\x85x\xe1\x02\x12\xca\x14\xff\a2\xa8$\x1e7F}\xb8V2\xa9\x8a\x01EB\xba\x12\xa37\xc0
\x00\x00\u07d4\x85y\xda\xdf\x1a9Z4q\xe2\vov=\x9a\x0f\xf1\x9a?o\x89\xd8\xd7&\xb7\x17z\x80\x0
0\x00\u07d4\x85\u007f\x10\v\x1aY0\"^\xfc~\x90
\u05c3'\xb4\x1c\x02\u02c9lk\x93[\x8b\xbd@\x00\x00\u07d4\x85\x94mV\xa4\xd3q\xa93hS\x96\x90
\xb6\x0e\xc8%\x10tT\x89]\u0212\xaa\x111\xc8\x00\x00\xe0\x94\x85\x99\xcb\u0566\xa9\xdc\u053

9f\xbe8}iw]\xa5\xe3<o\x8a\fQ\xf1\xb1\xd5&\"\x90\x00\x00\u07d4\x85\x9c`\f\xf1=\x1d\x02s\xd5\xd1\xda<\u05c9\u4549\x9f'\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\x85\xa2\xf6\xea\x94\xd0^\x8c\x1d\x9a\xe2\xf4\x91\x038\xa3X\xe9\x8d\xed\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x85\xb1o\v\x8b4\xdf\xf3\x80Oi\xe2\x16\x8aO{$\xd1\x04+\x89\x11/B<vF\xd4\x00\x00\xe0\x94\x85\xb2\x99\x8d\fs0,\xb2\xba\x13\xf4\x8913\x01\xe0S\xbe\x15\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x85\xbbQ\xbc;\xfe\x9a\x1b*/k\x1c\u0695\xbc\xa8\xb3\x8c\x8d^\x89\x11q-
\xa0K\xa1\xef\x00\x00\u07d4\x85\xc8\xf3\xccz5O\xea\u025a^{\xfe|\u07e3Q\xcf\xe3U\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u0794\x85\xca\x1er~\x9d\x1a\x87\x99\x1c\xc2\xc4\x18@\xeb\xb9\xed\xf2\x1d\x1b\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\x85\u028b\xc6\xda(\x03\xd0r_^\x1aEl\x89\xf9\xbcwN/\x89
\x86\xac5\x10R`\x00\x00\u07d4\x85\xd0\u0607T\xac\x84\xb8\xb2\x1b\xa9=\u04bf\xecrbo\xab\xa8\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x85\xeb%kQ\xc8\x19\xd6\x0e\xa6\x1a\x82\xd1,\x93X\u055c\x1c\xae\x89\x18\xef\xc8J\xd0\u01f0\x00\x00\u07d4\x85\xf0\xe7\xc1\xe3\xaf\xf8\x05\xa6'\xa2\xaa\xf2\xcf\xf6\xb4\xc0\xdb\xe9\u02c9\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x86\x02l\xad?\xe4\xea\x1c\xe7\xfc\xa2`\xd3\xd4^\xb0\x9e\xa6\xa3d\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x86\x0f_\xfc\x10\xdev}\xed\x80\u007fq\xe8a\xd6G\xdf\xd2\x19\xb1\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x86\x150c\xa1\xae\u007f\x02\xf1\xa8\x816\xd4\u059c|^>C'\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x86$_Yf\x91\t>\xce?=<\xa2&>\xac\xe8\x19A\u0649\n1\x06+\xee\xedp\x00\x00\u07d4\x86%i!\x1e\x8cc'\xb5A^:g\xe5s\x8b\x15\xba\xafn\x89\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4\x86)}s\x0f\xe0\xf7\xa9\xee$\xe0\x8f\xb1\b{1\xad\xb3\x06\xa7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x86D\xcc(\x1b\xe32\xcc\xce\xd3m\xa4\x83\xfb*\aF\u067a.\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x86I\x9a\x12(\xff-
~\xe3\au\x93dPo\x8e\x8c\x83\a\xa5\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x86K\xecPi\xf8U\xa4\xfdX\x92\xa6\xc4I\x1d\xb0|\x88\xff|\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x86W\n\xb2Y\u0271\xc3,\x97) /w\xf5\x90\xc0}\xd6\x12\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x86c\xa2A\xa0\xa8\x9ep\xe1\x82\xc8E\xe2\x10\\\x8a\xd7&K\u03ca\x03#\xb1=\x83\x98\xf3#\x80\x00\u07d4\x86g\xfa\x11U\xfe\xd72\u03f8\u0725\xa0\xd7e\xce\r\a\x05\xed\x89\x04n\xc9e\u00d3\xb1\x00\x00\u07d4\x86h\xaf\x86\x8a\x1e\x98\x88_\x93\u007f&\x15\xde\xd6u\x18\x04\xeb-
\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\x86t\nFd\x8e\x84Z]\x96F\x1b\x18\t\x1f\xf5{\xe8\xa1o\x8a\x14\xc0\x974\x85\xbf9@\x00\x00\xe0\x94\x86~\xbaVt\x8aY\x045\r,\xa2\xa5\u039c\xa0\vg\n\x9b\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\x86\x80dt\xc3X\x04}\x94\x06\xe6\xa0\u007f@\x94[\xc82\x8eg\x8a\x01u.\xb0\xf7\x01=\x10\x00\x00\u07d4\x86\x88=T\xcd9\x15\xe5I\tU0\xf9\xab\x18\x05\xe8\xc5C-
\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x86\x8c#\xbe\x874f\xd4\xc7L\"\n\x19\xb2E\xd1x~\x80\u007f\x89J\x13\xbb\xbd\x92\u020e\x80\x00\xe0\x94\x86\x92O\xb2\x11\xaa\xd2<\xf5\xce`\x0e\n\xae\x80c\x96D@\x87\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x86\x93\u9e3e\x94B^\xefyi\xbci\xf9\xd4/|\xadg\x1e\x8967\tlK\xcci\x00\x00\xe0\x94\x86\x9f\x1a\xa3\x0eDU\xbe\xb1\x82
\x91\xde\\\xad\xecy\xa8\xf9F\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x86\xa1\xea\xde\xeb0F\x13E\xd9\xefk\xd0R\x16\xfa$|\r\f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x86\xa5\xf8%\x9e\u0570\x9e\x18\x8c\xe3F\xee\x92\xd3J\xa5\u0753\xfa\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x86\xb7\xbdV<\uad86\xf9bD\xf9\xdd\xc0*\u05f0\xb1K\u008a\x02\x1e\x19\xe0\u0

27a\xb2@\x00\x00\u07d4\x86\u008bVx\xaf7\xd7'\xec\x05\xe4Dw\x90\xf1_q\xf2\xea\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x86\xc4\xce\x06\u066c\x18[\xb1H\xd9o{z\xbes\xf4A\x00m\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x86\xc8\xd0\u0642\xb59\xf4\x8f\x980\xf9\x89\x1f\x9d`z\x94&Y\x8a\x02\xce\xd3wa\x82O\xb0\x00\x00\u07d4\x86\xc94\xe3\x8eS\xbe;3\xf2t\xd0S\x9c\xfc\xa1Y\xa4\xd0\u04494\x95tD\xb8@\xe8\x00\x00\xe0\x94\x86\xca\x01E\x95~k\r\xfe6\x87_\xbez\r\xecU\xe1z(\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x86\u02af\xac\xf3*\xa01|\x03*\xc3k\xab\xed\x97G\x91\xdc\x03\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\x86\u0377\xe5\x1a\xc4Gr\xbe6\x90\xf6\x1d\x0eYvn\x8b\xfc\x18\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x86\xdfs\xbd7\u007f,\t\xdec\xc4]g\xf2\x83\xea\xef\xa0\xf4\xab\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x86\xe3\xfe\x86\xe9=\xa4\x86\xb1Bf\xea\xdf\x05l\xbf\xa4\xd9\x14C\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x86\xe8g\x0e'Y\x8e\xa0\x9c8\x99\xabw\x11\u04f9\xfe\x90\x1c\x17\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x86\xefd&!\x19I\xcc7\xf4\xc7^xP6\x9d\f\xf5\xf4y\x8a\x02\xd6_2\xea\x04Z\xf6\x00\x00\u07d4\x86\xf0]\x19\x06>\x93i\xc6\x00N\xb3\xf1#\x94:|\xffN\xab\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\x86\xf2>\x9c\n\xaf\u01cb\x9c@M\xcd`3\x9a\x92[\xff\xa2f\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x86\xf4\xf4\n\u0644\xfb\xb8\t3\xaebn\x0eB\xf93?\xddA\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x86\xf9\\[\x11\xa2\x93\x94\x0e5\xc0\xb8\x98\u0637_\b\xaa\xb0m\x8a\x06D\xe3\xe8u\xfc\xcft\x00\x00\u07d4\x86\xff\xf2
\xe5\x93\x05\xc0\x9fH8`\xd6\xf9N\x96\xfb\xe3/W\x89\x02S[j\xb4\xc0B\x00\x00\u07d4\x87\a\x96\xab\xc0\u06c4\xaf\x82\xdaR\xa0\xedhsM\xe7\xe66\xf5\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x87\x0f\x15\xe5\u07cb\x0e\xab\xd0%iSz\x8e\xf9;Vx\\B\x89\x15\b\x94\xe8I\xb3\x90\x00\x00\u07d4\x87\x181`\xd1r\xd2\xe0\x84\xd3'\xb8k\xcb|\x1d\x8eg\x84\xef\x89\xd8\xd8X?\xa2\xd5/\x00\x00\xe0\x94\x87\x1b\x8a\x8bQ\u07a1\x98\x9aY!\xf1>\xc1\xa9U\xa5\x15\xadG\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x87%\xe8\xc7S\xb3\xac\xbf\u0725_<b\xdf\u1954T\x96\x8a\x8967\tlK\xcci\x00\x00\u07d4\x877\xda\xe6q\x82:\x8dY\x17\xe0\x15z\u039cCF\x8d\x94k\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\x87;\u007fxm<\x99\xff\x01,J|\xae&w'\x02@\xb9\u0149]\u0212\xaa\x111\xc8\x00\x00\u07d4\x87<op\ufdb1\xd0\xf2\xbb\xc5~\xeb\xcdpa|l\xe6b\x896\xf0\xd5']\tW\x00\x00\u07d4\x87>I\x13\\3\x91\x99\x10`)\n\xa7\xf6\u0338\xf8Zx\u06c9\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x87Pa\xee\x12\xe8\x04\x1a\x01\x94,\xb0\xe6[\xb4'\xb0\x00`\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\x87XJ?a;\xd4\xfa\xc7L\x1ex\v\x86\xd6\xca\xeb\x89\f\xb2\x89\\(=A\x03\x94\x10\x00\x00\u07d4\x87d\xd0'\"\x00\t\x96\xec\xd4u\xb43)\x8e\x9fT\v\x05\xbf\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x87l?!\x8bGv\xdf<\xa9\xdb\xfb'\r\xe1R\xd9N\xd2R\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x87u\xa6\x10\xc5\x02\xb9\xf1\xe6\xadL\xda\u06cc\xe2\x9b\xffu\xf6\xe4\x89\x86\xac5\x10R`\x00\x00\u07d4\x87vN6w\xee\xf6\x04\xcb\u015a\xed$\xab\xdcVk\t\xfc%\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94\x87\x87\xd1&w\xa5\xec)\x1eW\xe3\x1f\xfb\xfa\xd1\x05\xc32K\x87\x8a\x02\xa2N\xb52\b\xf3\x12\x80\x00\u07d4\x87\x94\xbfG\xd5E@\xec\xe5\xc7\"7\xa1\xff\xb5\x11\u0777Gb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x87\xa5>\xa3\x9fY\xa3[\xad\xa85%!dU\x94\xa1\xa7\x14\u02c9g\x8a\x93 b\xe4\x18\x00\x00\u07d4\x87\xa7\xc5\b\xefqX-\u0665Cr\xf8\x9c\xb0\x1f%/\xb1\x80\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x87\xaf%\xd3\xf6\xf8\xee\xa1S\x13\xd5\xfeEW\xe8\x10\xc5$\xc0\x83\x8a\x04+\xf0kx\xed;P\x00\x00\u07d4\x87\xb1\x0f\x9c(\x00\x98\x17\x9a+v\xe9\u0390\xbea\xfc\x84M\r\x89Hz\x9a0E9D\x00\x00\u07d4\x87\xbf|\xd5\u0629)\xe1\u01c5\xf9\xe5D\x91\x06\xac#$c\u0249\x047\xb

1\x1f\xccEd\x00\x00\u07d4\x87\u0118\x17\t4\xb8#=\x1a\xd1\xe7i1}\\G_/@\x897\b\xba\xed=h\x90\x00\x00\u07d4\x87\xcf6\xad\x03\xc9\xea\xe9\x05:\xbbRB\u0791\x17\xbb\x0f*\v\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\x87\u05ec\x06S\xcc\xc6z\xa9\xc3F\x9e\xefCR\x19?}\xbb\x86\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\xe0\x94\x87\xe3\x06+#!\xe9\u07f0\x87\\\u311c\x9b.5\"\xd5\n\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x87\xe6\x03N\xcf#\xf8\xb5c\x9d_\x0e\xa7\n\"S\x8a\x92\x04#\x89\x11\xc7\xea\x16.x
\x00\x00\u07d4\x87\xefm\x8bj|\xbf\x9b\\\x8c\x97\xf6~\xe2\xad\u00a7;?w\x89\n\xdd\x1b\xd2<<H\x00\x00\u07d4\x87\xfb&\xc3\x1eHdMi14
\\\xaeC\xb2\x1f\x18aK\x89JD\x91\xbdm\xcd(\x00\x00\u07d4\x87\xfcF5&9D\xce\x14\xa4lu\xfaJ\x82\x1f9\xce\u007fr\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x87\xfc\xbe|A\x93\xff\xcb\b\x147y\u027e\xc8?\xe7\xfd\xa9\xfc\x89\x05o\x98]8dK\x80\x00\xe0\x94\x88\x01]r\x03\xc5\xe0\"J\xed\xa2\x86\xed\x12\xf1\xa5\x1bx\x933\x8a\x01\x0f\b\xed\xa8\xe5U\t\x80\x00\u07d4\x88\x10l\"\xd2\vt\xb4\xb9\x8c\xa6+#+\xd5\xc9t\x11\x17\x1f\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94\x88\x120\x04|!\x1d-[\x00\xd8\xdeLQ9\xde^2'\u01ca\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x88*\xa7\x98\xbfA\xdf\x17\x9f\x85R\x010\xf1\\\xcd\xf5\x9b^X\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x88+\u04e2\xe9\xd7A\x10\xb2Ia\xc57w\xf2/\x1fF\xdc]\x8a\x02\xd4\xca\x05\xe2\xb4<\xa8\x00\x00\xe0\x94\x88,\x8f\x81\x87,y\xfe\xd5!\xcb_\x95\r\x8b\x03#\"\xeai\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\x88/up\x83\x86e<\x80\x17\x1d\x06c\xbf\xe3\v\x01~\u042d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x884I\tdLz\u0513\x0f\xd8s\xca\x1c\r\xa2\xd44\xc0\u007f\x89\a's\x9f\xcb\x00M\x00\x00\u07d4\x884\xb2E4q\xf3$\xfb&\xbe[%\x16k[W&\x02]\x89\x1f\x0f\xf8\xf0\x1d\xaa\xd4\x00\x00\u07d4\x88:x\xae\xab\xaaP\xd8\xdd\xd8W\v\xcd4&_\x14\xb1\x93c\x89\xd2U\"\xfd\xa3y\xa1\x80\x00\u07d4\x88E\xe9\xf9\x0e\x963k\xac<ak\xe9\u0604\x02h>\x00L\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x88F\x92\x8dh2\x89\xa2\xd1\x1d\xf8\xdbz\x94t\x98\x8e\xf0\x13H\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x88I\x80\xebEe\xc1\x04\x83\x17\xa8\xf4\u007f\u06f4a\x96[\u4049\xd8\xd6\x11\x9a\x81F\x05\x00\x00\xe0\x94\x88Jz9\u0411n\x05\xf1\xc2B\xdfU`\u007f7\u07cc_\u068a\x04\xf4\x84<\x15|\x8c\xa0\x00\x00\u07d4\x88T\x93\xbd\xa3j\x042\x97eF\xc1\xdd\xceq\xc3\xf4W\x00!\x89\v\xbfQ\r\xdf\xcb&\x00\x00\xe0\x94\x88`\x9e\nF[n\x99\xfc\xe9\a\x16mW\xe9\xda\b\x14\xf5\u020a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x88m\n\x9e\x17\xc9\xc0\x95\xaf.\xa25\x8b\x89\xecpR\x12\ue509\x01\x84\x93\xfb\xa6N\xf0\x00\x00\u07d4\x88y~Xg^\xd5\xccL\x19\x98\a\x83\xdb\xd0\xc9V\bQS\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x88|\xacA\xcdpo3E\xf2\xd3J\xc3N\x01u*nY\t\x89
F\\\ue7617\x00\x00\u07d4\x88\x88\x8aW\xbd\x96\x87\xcb\xf9P\xae\xea\u03d7@\xdc\xc4\xd1\xefY\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u0794\x88\x89D\x83\x16\xcc\xf1N\xd8m\xf8\xe2\xf4x\xdcc\xc43\x83@\x88\xd2\xf1?w\x89\xf0\x00\x00\u07d4\x88\x8c\x16\x14I3\x19|\xac&PM\xd7n\x06\xfdf\x00\u01c9\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x88\x8e\x94\x91p\x83\xd1R
+S\x1699\x86\x9d'\x11u\xb4\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x88\x90\x87\xf6o\xf2\x84\xf8\xb5\xef\xbd)I;pg3\xab\x14G\x8a\x02\x15\xf85\xbcv\x9d\xa8\x00\x00\u07d4\x88\x95\xebrb&\xed\xc3\xf7\x8c\u01a5\x15\a{2\x96\xfd\xb9^\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x88\x97Z_\x1e\xf2R\x8c0\v\x83\xc0\xc6\a\xb8\xe8}\u0593\x15\x89\x04\x86\u02d7\x99\x19\x1e\x00\x00\u07d4\x88\x9d\xa4\x0f\xb1\xb6\x0f\x9e\xa9\xbdzE>XL\xf7\xb1\xb4\xd9\xf7\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\x88\x9d\xa6b\xebJ\n*\x06\x9d+\xc2K\x05\xb4\xee.\x92\xc4\x1b\x89Z,\x8cTV\xc9\xf2\x80\x00\u07d4\x88\xa1\"\xa28,R91\xfbQ\xa0\u032d;\xeb[rY\u00c9lk\x93[\x8b\xbd@\x00\x00\u07d4\x88\xa2\x15D0\xc0\xe4\x11G\xd3\xc1\xfe\u3cf0\x06\xf8Q\xed\xbd\x8965f3\xeb\xd8\xea\x00\x00\u

07d4\x88\xb2\x17\u0337\x86\xa2T\xcfM\xc5\u007f]\x9a\xc3\xc4U\xa3\x04\x83\x892$\xf4'#\xd4T\x
00\x00\xe0\x94\x88\xbcC\x01.\xdb\x0e\xa9\xf0b\xacCxC%\n9\xb7\x8f\xbb\x8a\x04<3\xc1\x93ud\x
80\x00\x00\u07d4\x88\xc2Qj|\xdb\t\xa6'mr\x97\xd3\x0fZM\xb1\xe8K\x86\x89\xd8\xd7&\xb7\x17z\x
80\x00\x00\xe0\x94\x88\xc3ad\rki7;\b\x1c\xe0\xc43\xbdY\x02\x87\xd5\xec\x8a\n\x96\x81c\xf0\xa5{
@\x00\x00\u07d4\x88\xd5A\xc8@\xceC\xce\xfb\xafm\x19\xafk\x98Y\xb5s\xc1E\x89\t79SM(h\x00\
x00\u07d4\x88\xde\x13\xb0\x991\x87|\x91\rY1e\xc3d\u0221d\x1b\u04c9\xa2\xa1]\tQ\x9b\xe0\x00\
x00\u07d4\x88\xde\u017d?N\xba-
\x18\xb8\xaa\xce\xfa{r\x15H\xc3\x19\xba\x89JD\x91\xbdm\xcd(\x00\x00\u07d4\x88\xe6\xf9\xb2G\x
f9\x88\xf6\xc0\xfc\x14\xc5o\x1d\xe5>\u019dC\u0309\x05k\xc7^-
c\x10\x00\x00\u07d4\x88\xee\u007f\x0e\xfc\x8fw\x8ckh~\xc3+\xe9\xe7\xd6\xf0
\xb6t\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x88\xf1\x04_\x19\xf2\xd3\x19\x18\x16\xb1\xdf\x18\xbbn
\x145\xad\x1b8\x89\r\x02\xabHl\xed\xc0\x00\x00\xe0\x94\x89\x00\x9e<d\x88\xbd^W\r\x1d\xa3N\x
ab\xe2\x8e\xd0$\xde\x1b\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x89\x05D0\xdc\xdc(\xac\x15\xf
ac^\xf8|\x10^`+\xf7\f\x89\x05\xda\xcd\x13\u029e0\x00\x00\xe0\x94\x89\bv\f\u04db\x9c\x1e\x81\x8
4\xe6\xa7R\ue20e?\vpE\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\x89\x0f\xe1\x1f<$\u06c72\
xd6\xc2\xe7r\xe2)|~e\xf19\x8a\rV'\x13}\xa8\xb5\x90\x00\x00\u07d4\x89\x14\xa6\x80\xa5\xae\xc5\"
mK\xaa\xec.UR\xb4M\xd7\xc8t\x89\x05l\xd5_\xc6M\xfe\x00\x00\u07d4\x89\x1c\xb8#\x8c\x88\xe9:\
x1b\xcfa\xdbI\xbd\x82\xb4z\u007fO\x84\x89\x91Hx\xa8\xc0^\xe0\x00\x00\u07d4\x89%\xdaEI\xe1
QU\xe5zb\x85\"\u03a9\xdd\xdfb}\x81\x8966\xc2^f\xec\xe7\x00\x00\u07d4\x890\x17\xff\x1a\xda\u0
519\xaa\x06T\x01\xb4#l\xe6\xe9+bZ\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94\x893I\x17`\xc8\xf0\xb4
\u07cc\xaa\u01ce\xd85\xca\xee!\x04m\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x896\bu\x1dh\xd
0F\xe8X\x02\x92fs\xcd\xf2\xf5\u007f|\xb8\x89\x01\x11du\x9f\xfb2\x00\x00\u0794\x898\u0474\xda\
xeeU\xa5Ms\x8c\xf1~Dw\xf6yNF\xf7\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x89:I.\xb8\xb4\n\xb0
\x96\xb4\xf6~t\xa8\x97\xb8@tn\x86\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4\x89<\xdd\xdfSw\xf3\
xc7Q\xbf.T\x11 \x04ZG\u02e1\x01\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x89V\x13#o5\x84!j\xd7\\]>\a\xe3\xfahc\xa7x\x89lk\x93[\x8b\xbd@\x00\x00\u0
7d4\x89Wr~r\xcfb\x90
\xf4\xe0^\xdfy\x9a\xa7E\x80b\u0409wC\"\x17\xe6\x83`\x00\x00\u07d4\x89]iN\x88\v\x13\xcc\u0404
\x8a\x86\xc5\xceA\x1f\x88Gk\xbf\x89\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4\x89^\xc5TVD\u0dc30\
xff\xfa\xb8\xdd\xea\xc9\xe83\x15l\x89
\x86\xac5\x10R`\x00\x00\u07d4\x89`\tRj,{\f\t\xa6\xf6:\x80\xbd\U0009d707\u079c\x89\xbb\xb8k\x82
#\xed\xeb\x00\x00\u07d4\x89g\u05f9\xbd\xb7\xb4\xae\xd2.e\xa1]\xc8\x03\xcbz!?\x10\x89\x15\xaf\
x1dx\xb5\x8c@\x00\x00\u07d4\x89n3\\\xa4z\xf5yb\xfa\x0fM\xbf>E\xe6\x88\u02e5\x84\x89J/\xc0\x
ab`R\x12\x00\x00\u07d4\x89s\xae\xfd^\xfa\xee\x96\t]\x9e(\x8fj\x04l\x977KC\x89\a\xa4\u0120\xf32
\x14\x00\x00\u07d4\x89\x8cr\xddseX\xef\x9eK\xe9\xfd\xc3O\xefT\xd7\xfc~\b\x8965\u026d\xc5\u0
7a0\x00\x00\u07d4\x89\x9b<$\x9f\fK\x81\xdfu\xd2\x12\x00M=m\x95/\xd2#\x89lk\x93[\x8b\xbd@\x
00\x00\xe0\x94\x89\xab\x13\xee&mw\x9c5\xe8\xbb\x04\u034a\x90\xcc!\x03\xa9[\x8a\f\xb4\x9bD\x
ba`-
\x80\x00\x00\u07d4\x89\xc43\xd6\x01\xfa\xd7\x14\xdaci0\x8f\xd2l\x1d\u0254+\xbf\x89lk\x93[\x8b\
xbd@\x00\x00\u07d4\x89\xd7[\x8e\b1\xe4o\x80\xbc\x17A\x88\x18N\x00o\xde\x0e\xae\x8965\u02
6d\xc5\u07a0\x00\x00\u07d4\x89\u3d5a\x15\x86G7\u0513\xc1\xd2<\xc5=\xbf\x8d\xcb\x13b\x89\x
d8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x89\xfc\x8eM8k\r\v\xb4\xa7\a\xed\xf3\xbdV\r\xf1\xad\x8fN\x

89\xa00\xdc\xeb\xbd/L\x00\x00\u07d4\x89\xfe\xe3\r\x17(\xd9l\xec\xc1\u06b3\xda.w\x1a\xfb\u03ea
A\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94\x8a\x1c\u016c\x11\x1cI\xbf\xcf\xd8H\xf3}\xd7h\xaae\u0208\x02\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x8a
\xe5\xb5\xce\xe7\xcd\x1fU\x15\xba\xce;\xf4\xf7\u007f\xfd\xe5\xcc\a\x89\x04V9\x18$O@\x00\x00\xe0\x94\x8a!}\xb3\x8b\xc3_!_\xd9)\x06\xbeBCo\xe7\xe6\xed\x19\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x8a$:\n\x9f\xeaI\xb89TwE\xff-
\x11\xaf?K\x05\"\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\x8a$}\x18e\x10\x80\x9fq\xcf\xfcEYG\x1c9\x10\x85\x81!\x89a\t=|,m8\x00\x00\u07d4\x8a4p(-
^**\xef\u05e7P\x94\xc8\"\xc4\xf5\xae\uf289\r(\xbc`dx\xa5\x80\x00\u07d4\x8a6\x86\x9a\xd4x\x99|\xbfm\x89$\xd2\n<\x80\x18\xe9\x85[\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x8aC\x14\xfba\u0353\x8f\xc3>\x15\xe8\x16\xb1\x13\U000ac267\xfb\x89\x17vNz\xede\x10\x00\x00\u07d4\x8aOJ\u007fR\xa3U\xba\x10_\xca
r\xd3\x06_\xc8\xf7\x94K\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x8aX1(,\xe1Jezs\r\xc1\x88&\xf7\xf9\xb9\x9d\xb9h\x89\uaf8a[A\xc16\x00\x00\u07d4\x8a_\xb7W\x93\xd0C\xf1\xbc\xd48\x85\xe07\xbd0\xa5(\xc9'\x89\x13Snm.\x9a\xc2\x00\x00\u07d4\x8af\xab\xbc-
0\xce!\xa83\xb0\u06ceV\x1dQ\x05\xe0\xa7,\x89%\xf1\xde\\v\xac\xdf\x00\x00\u07d4\x8atl]g\x06G\x11\xbf\xcah[\x95\xa4\xfe)\x1a'\x02\x8e\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\x8ax\n\xb8z\x91E\xfe\x10\xed`\xfaGjt\n\xf4\u02b1\u0489\x12\x1b.^ddx\x00\x00\u07d4\x8az\x06\xbe\x19\x9a:X\x01\x9d\x84j\xc9\xcb\xd4\xd9]\xd7W\u0789\xa2\xa4#\x94BV\xf4\x00\x00\u07d4\x8a\x81\x01\x14\xb2\x02]\xb9\xfb\xb5\x00\x99\xa6\xe0\u02de.\xfak\u0709g\x8a\x93
b\xe4\x18\x00\x00\u07d4\x8a\x86\xe4\xa5\x1c\x01;\x1f\xb4\xc7k\xcf0f|x\xd5.\xed\xef\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8a\x9e\u029cZ\xba\x8e\x13\x9f\x80\x03\xed\xf1\x16:\xfbp\xaa:\xa9\x89#\xc7W\a+\x8d\xd0\x00\x00\u07d4\x8a\xb89\xae\xaf*\xd3|\xb7\x8b\xac\xbb\xb63\xbc\xc5\xc0\x99\xdcF\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8a\u021b\u06780\x1ek\x06w\xfa%\xfc\xf0\xf5\x8f\f\u01f6\x11\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\x8a\xdcS\xef\x8c\x18\xed0Qx]\x88\xe9\x96\xf3\xe4\xb2\x0e\xcdQ\x8a\b\xe4\xd3\x16\x82v\x86@\x00\x00\u07d4\x8a\xe6\xf8\vp\xe1\xf2<\x91\xfb\u0569f\xb0\xe4\x99\xd9]\xf82\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\x8a\xe9\uf30a\x8a\u07e6\xaby\x8a\xb2\xcd\xc4\x05\b*\x1b\xbbp\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8a\xf6&\xa5\xf3'\xd7Pe\x89\xee\xb7\x01\x0f\xf9\xc9D`
\u0489K\xe4\xe7&{j\xe0\x00\x00\xe0\x94\x8b\x01\xda4\xd4p\xc1\xd1\x15\xac\xf4\xd8\x11<M\u0628\xc38\xe4\x8a\x05W-
\xce\xfa\xb6\x97\x90\x00\x00\u07d4\x8b\a\xd0PuM\u027a#\r\xb0\x1c1\n\xfd\xb59Z\xa1\xb3\x89\x06f\xb0nb\xa6 \x00\x00\u07d4\x8b
\xad;\x94em\xbd\xc0\xdd!\xa3\x93\u0627\xd9\xe0!8\u02c9\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x8b'9\"\x06\xb9X\xcd7]~\xf8\xaf,\xf8\xef\x05\x98\xc0\xbc\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x8b0\xc0@\x98\u05e7\xe6B\f5~\xa7\xbf\xa4\x9b\xac\x9a\x8a\x18\x8a\x01\xb1\xb1\x13\xf9\x1f\xb0\x14\x00\x00\u07d4\x8b3\x84\x11\xf2l\xcf7e\x8c\xc7U!\xd7v)\t\x9eF}\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8b6\"LsV\xe7Q\xf0\xc0f\xc3^;D\x86\x03d\xbf\u00896'\xba\u01e3\xd9'\x80\x00\xe0\x94\x8b6\x96\xf3\xc6\r\xe3$2\xa2\xe4\u00d5\xef\x03\x03\xb7\xe8\x1eu\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4\x8b9?\xb0\x81>\xe1\x01\xdb\x1e\x14\xec\xc7\xd3\"\xc7+\x8c\x04s\x89\x18\xb2j1>\x8a\xe9\x00\x00\xe0\x94\x8bH\xe1\x9d9\xdd5\xb6nn\x1b\xb6\xb9\xc6W\xcb,\xf5\x9d\x04\x8a\x03\xc7U\xac\x9c\x02J\x01\x80\x00\xe0\x94\x8bP^(q\xf7\u07b7\xa68\x95

\x8e\x82'\u072a\x1b\xff\x05\x8a\f\xf6\x8e\xfc0\x8dy\xbc\x00\x00\u07d4\x8bW\xb2\xbc\x83\u030d
M\xe31
N\x89?/;\x1d\xb1\a\x9a\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\x8b\\\x91K\x12\x8b\xf1i\\\b\x89#
\xfaF~y\x11\xf3Q\xfa\x89\x05V\xf6L\x1f\xe7\xfa\x00\x00\xe0\x94\x8b_)\xcc/\xaa&,\xde\xf3\x0e\xf5
T\xf5\x0e\xb4\x88\x14n\xac\x8a\x01;hp\\\x97
\x81\x00\x00\u07d4\x8bpV\xf6\xab\xf3\xb1\x18\xd0&\xe9D\xd5\xc0sC<\xa4Q\u05c965\xc6
G9\u0640\x00\u07d4\x8bqE\"\xfa(9b\x04p\xed\xcf\fD\x01\xb7\x13f=\xf1\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x8bt\xa7\xcb\x1b\xb8\u014f\xce&tf\xa3\x03X\xad\xafR\u007fa\x8a\x02\xe2WxN
%\xb4P\x00\x00\u07d4\x8b~\x9fo\x05\xf7\xe3dv\xa1n>q\x00\xc9\x03\x1c\xf4\x04\xaf\x8965\u026d
\xc5\u07a0\x00\x00\u07d4\x8b\x81\x15ni\x869\x94<\x01\xa7Rr\xad=5\x85\x1a\xb2\x82\x89\x12\x
b3\x16_e\xd3\xe5\x00\x00\u07d4\x8b\x95w\x92\x00S\xb1\xa0\x01\x890M\x88\x80\x10\xd9\xef,\xb
4\xbf\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x8b\x98A\x86.w\xfb\xbe\x91\x94p\x93U\x83\x
a9<\xf0'\xe4P\x89llS4B\u007f\x1f\x00\x00\u07d4\x8b\x99}\xbc\a\x8a\xd0)a5]\xa0\xa1Y\xf2\x92~\x
d4=d\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94\x8b\x9f\xda}\x98\x1f\xe9\xd6B\x87\xf8\\\x94\xd8
?\x90t\x84\x9f\u030a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4\x8b\xb0!/2\x95\xe0)\u02b1\xd9a\x
b0A3\xa1\x80\x9e{\x91\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8b\xbe\xac\xfc)\xcf\xe94\x02\xdb<A\
u065a\xb7Yf.s\xec\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x8b\xc1\xff\x87\x14\x82\x8b\xf2\x86\xff~
\x8aw\t\x10eH\xed\x1b\x18\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x8b\u0436ZP\xef\\
\xef\x84\xfe\xc4
\xbe{\x89\xed\x14p\u03b9\x8a\x02\x8aw\x93n\x92\xc8\x1c\x00\x00\u07d4\x8b\u05b1\xc6\xd7M\x0
1\r\x10\b\u06e6\xef\x83]D0\xb3\\2\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\x8b\xd8\xd4\x
c4\xe9C\xf6\xc8\a9!\xdc\x17\xe3\xe8\u05e0v\x16'\x89\x9f\x04!\x9d\x8d4\x95\x00\x00\u07d4\x8b\x
df\xdal!W
\xed\xa2\x13o\x91\x05#!\xafN\x93l\x1f\x8965\xe6\x19\xbb\x04\xd4\x00\x00\u07d4\x8b\xea@7\x93
G\xa5\u0211\u055acc1V@\xf5\xa7\xe0z\x89lkv\uf597\f\x00\x00\xe0\x94\x8b\xf0+\xd7Hi\x0e\x1f\x
d1\xc7m'\b3\x04\x8bf\xb2_\u04ca\x02\u007f\xad\xe5h\xeb\xa9`\x00\x00\u07d4\x8b\xf2\x97\xf8\xf4
SR>\xd6j\x1a\xcbvv\x85c7\xb9;\xf0\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8b\xf3s\xd0v\x81
L\xbcW\xe1\xc6\xd1j\x82\u017e\x13\xc7=7\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x8c\x10#\xfd\xe1WM\xb8\xbbT\xf1s\x96p\x15|\xa4}\xa6R\x8a\x01y\u03da\u00e1
\xb1w\x00\x00\u07d4\x8c\x1f\xbe_\n\xea5\x9cZ\xa1\xfa\b\u0209T\x12\u028e\x05\xa6\x8965\u026
d\xc5\u07a0\x00\x00\u07d4\x8c\"B`U\xb7o\x11\xf0\xa2\xde\x1a\u007f\x81\x9aa\x96\x85\xfe`\x89k
V\x05\x15\x82\xa9p\x00\x00\u07d4\x8c+}\x8b`\x8d(\xb7\u007f\\\xaa\x9c\xd6E$*\x82>L\u0649b\xa
9\x92\xe5:\n\xf0\x00\x00\u07d4\x8c/\xbe\ue3ac\xc5\xc5\xd7|\x16\xab\xd4b\ue701E\xf3K\x89i*\xe8
\x89p\x81\xd0\x00\x00\u07d4\x8c:\x9e\xe7\x1fr\x9f#l\xba8g\xb4\u05dd\x8c\xee\xe2]\xbc\x89\x05k\
xc7^-
c\x10\x00\x00\u07d4\x8cP\xaa*\x92\x12\xbc\xdeVA\x8a\xe2a\xf0\xb3^z\x9d\xbb\x82\x89\x15\xaf\x
1dx\xb5\x8c@\x00\x00\u07d4\x8cT\xc7\xf8\xb9\x89nu\xd7\xd5\xf5\xc7`%\x86\x99\x95qB\xad\x89\
x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\x8c]\x16\xede\xe3\xed~\x8b\x96\u0297+\xc8as\xe3P\v\x03\
x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8cj\xa8\x82\xee2,\xa8HW\x8c\x06\xcb\x0f\xa9\x11\xd3`\x83
\x05\x89 \x86\xac5\x10R`\x00\x00\xe0\x94\x8cj\xe7\xa0Z\x1d\xe5u\x82\xae'h
Bv\xc0\xffG\xed\x03\x8a,\v\xb3\xdd0\xc4\xe2\x00\x00\x00\u07d4\x8co\x9fN[z\xe2v\xbfXI{\u05ff*}%
$_d\x89\x93\xfe\\W\xd7\x10h\x00\x00\u07d4\x8cu\x95n\x8f\xedP\xf5\xa7\xdd|\xfd'\xda

\x0fgF\xae\xa6\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x8c|\xb4\xe4\x8b%\x03\x1a\xa1\xc4\xf9)%
\xd61\xa8\xc3\xed\xc7a\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x8c\u007f\xa5\xca\xe8/\xed\xb6\x
9a\xb1\x89\xd3\xff'\xae
\x92\x93\xfb\x93\x89\x15\xaf\x88\r\x8c\u06c3\x00\x00\xe0\x94\x8c\x81A\x0e\xa85L\xc5\xc6\\A\xb
e\x8b\xd5\xdes<\v\x11\x1d\x8a\x02\x05\xb4\u07e1\xeetx\x00\x00\u07d4\x8c\x83\xd4$\xa3\xcf$\xd
5\x1f\x01\x92=\xd5J\x18\u05b6\xfe\xde{\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8c\x90\n\x82
6\xb0\x8c+e@]9\xd7_
\x06*ua\xfd\x89X\xe7\x92n\xe8X\xa0\x00\x00\u07d4\x8c\x93\xc3\xc6\u06dd7q}\xe1e\u00e1\xb4\xf
eQ\x95,\b\u0789\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x8c\x99\x95\x91\xfdr\xefq\x11\xef\xcaz\x
9e\x97\xa25k;\x00\n\x89\xddd\xe2\xaa\ngP\x00\x00\u07d4\x8c\xa6\x98\x97F\xb0n2\xe2Hta\xb1\u
0399j':\xcf\u05c9\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x8c\xb3\xaa?\xcd!(T\xd7W\x8f\xcc0\xfd\x
ed\xe6t*1*\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x8c\xc0\xd7\xc0\x16\xfaz\xa9P\x11J\xa1\xdb\tH\x
82\xed\xa2t\xea\x89\b\xa9\xab\xa5W\xe3l\x00\x00\u07d4\x8c\xc6R\xdd\x13\xe7\xfe\x14\u06bb\xb
3m]2\r\xb9\xff\xee\x8aT\x89a\t=|,m8\x00\x00\u07d4\x8c\u02bf%\a\u007f:\xa4\x15E4MS\xbe\x1b+\
x9c3\x90\x00\x89[\xe8f\xc5b\xc5D\x00\x00\u07d4\x8c\xcf:\xa2\x1a\xb7BWj\xd8\xc4\"\xf7\x1b\xb1\
x88Y\x1d\ua28965\u026d\xc5\u07a0\x00\x00\u07d4\x8c\xd0\xcd\"\xe6
\xed\xa7\x9c\x04a\xe8\x96\xc9<D\x83~)h\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8c\u078bs.`#\x87\
x8e\xb2>\xd1b)\x12K_z\xfb\xec\x89\a?u\u0460\x85\xba\x00\x00\u07d4\x8c\xe2/\x9f\xa3rD\x9aB\x
06\x10\xb4z\xe0\xc8\xd5eH\x122\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x8c\u451d\x8a\x16T-
B<\x17\x98Ng9\xfar\u03b1w\x8a\x05K@Y&\xf4\xa6=\x80\x00\u07d4\x8c\xe5\xe3\xb5\xf5\x91\xd5\
uc8ca\xbf\"\x8f.<5\x13K\xda\xc0\x89}\xc3[\x84\x89|8\x00\x00\xe0\x94\x8c\xee8\xd6YW\x88\xa5n?
\xb9F4\xb3\xff\xe1\xfb\xdb&\u058a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x8c\xee\xa1^\xec;\xda\xd
8\x02?\x98\xec\xf2[+\x8f\xef'\xdb)\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8c\xf3To\xd1\u0363=X\x8
4_\xc8\xfc\xfe\u02bc\xa7\xc5d*\x89\x1f\x1e9\x93,\xb3'\x80\x00\u07d4\x8c\xf6\xda\x02\x04\xdb\u0
106\vF\xad\x97?\xc1\x11\x00\x8d\x9e\fF\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x8c\xfe\xde\xf1\x98\xdb\n\x91C\xf0\x91)\xb3\xfdd\u073b\x9bIV\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4\x8d\x04\xa5\xeb\xfb]\xb4\t\xdb\x06\x17\xc9\xfaV1\xc1\x92\x86\x1fJ\x894\x95tD\
xb8@\xe8\x00\x00\u07d4\x8d\x06\xe4d$\\\xadaI9\xe0\xaf\bE\xe6\xd70\xe2\x03t\x89\n\u070a(\xf3\
xd8}\x80\x00\u07d4\x8d\a\xd4-\x83\x1c-
|\x83\x8a\xa1\x87+:\xd5\xd2w\x17h#\x89\x12\xee\x1f\x9d\xdb\xeeh\x00\x00\u07d4\x8d\v\x9e\xa5?
\xd2cA^\xac\x119\x1f|\xe9\x12<Dpb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8d\x17\x94\xdaP\x9c\x
b2\x97\x056a\xa1J\xa8\x92321\xe3\xc1\x89\n\xd2\x01\xa6yO\xf8\x00\x00\u07d4\x8d\x1a\xbd\x89}
\xac\xd41.\x18\b\f\x88\xfb\x96G\xea\xb4@R\x89\v\xb5\x9a'\x95<`\x00\x00\u07d4\x8d#\x034\x1e\x
1e\x1e\xb5\xe8\x18\x9b\xde\x03\xf7:`\xa2\xa5Ha\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x8d#\x8e\x03e\x96\x98vC\xd71s\xc3{\n\xd0`U\xb9l\x89qH\xbf\n*\xf0f\x00\x0
0\xe0\x94\x8d.1\xb0\x88\x03\xb2\xc5\xf1=9\x8e\xca\u0605(
\x9f`W\x8a\x02\x1d\xb8\xbb\xca\xd1\x1e\x84\x00\x00\u07d4\x8d7\x8f\x0e\xdc\v\xb0\xf0hmj
\xbejv\x92\xc4\xfa$\xb8\x89\x05k\xc7^-
c\x10\x00\x00\u0794\x8dK`<]\xd4W\f4f\x95\x15\xfd\xccfX\x90\x84\fw\x88\xfc\x93c\x92\x80\x1c\x0
0\x00\u07d4\x8dQ\xa4\xccb\x01\x13\"\u0196\xfdr[\x9f\xb8\xf5?\xea\xaa\a\x8965\u026d\xc5\u07a0
\x00\x00\u07d4\x8dTL2\xc0\u007f\u0404,v\x1dS\xa8\x97\xd6\xc9P\xbbu\x99\x89\n\u05ce\xbcZ\xc
6

\x00\x00\u07d4\x8d^\xf1r\xbfw1^\xa6N\x85\xd0\x06\x19\x86\u01d4\xc6\xf5\x19\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x8dak\x1e\xeew\xee\xf6\xf1v\xe0i\x8d\xb3\xc0\xc1A\xb2\xfc\x8f\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x8dap\xfff\x97\x8ew;\xb6!\xbfr\xb1\xba{\xe3\xa7\xf8~\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x8db\v\xde\x17\"\x8fl\xbb\xa7M\xf6\xbe\x87&M\x98\\\xc1y\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x8db\x9c
`\x815I\x1bP\x13\xf1\x00%\x86\xa0810\xe5\x89JD\x91\xbdm\xcd(\x00\x00\u07d4\x8dfW\xf5\x97\x11\xb1\xf8\x03\xc6\xeb\xefh/\x91[b\xf9-
\u0249lk\x93[\x8b\xbd@\x00\x00\u07d4\x8dfv7\xe2\x9e\xca\x05\xb6\xbf\xbe\xf1\xf9mF\x0e\uffd9\x84\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8dm\xf2\tHM{\x94p+\x03\xa5>V\xb9\xfb\x06`\xf6\xf0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8dy_JV\x89\xadb\u0696\x16q\xf0(\x06R\x86\xd5T\x89o\x05\xb5\x9d; \x00\x00\x00\u07d4\x8d\u007f>a)\x9c-
\xb9\xb9\xc0H|\xf6'Q\x9e\xd0\n\x91#\x89^t\xa8P^\x80\xa0\x00\x00\xe0\x94\x8d\x89\x17\v\x92\xb2\xbe,\b\xd5|H\xa7\xb1\x90\xa2\xf1Fr\x0f\x8a\x04+\xf0kx\xed;P\x00\x00\u07d4\x8d\x93\xda\u01c5\xf8\x8f\x1a\x84\xbf\x92}Se+E\xa1T\xcc\u0749\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u07d4\x8d\x99R\u043bN\xbf\xa0\xef\xd0\x1a:\xa9\xe8\xe8\u007f\x05%t.\x89\xbb\x91%T\"c\x90\x00\x00\u07d4\x8d\x9a\fp\xd2& B\xdf\x10\x17\xd6\xc3\x03\x13
$w'\x12\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8d\x9e\xd7\xf4U0X\xc2ox6\xa3\x80-
0d\xeb\x1b6=\x89\x04\xe1\x00;(\xd9(\x00\x00\u07d4\x8d\xa1\x17\x8fU\xd9wr\xbb\x1d$\x11\x1a@
JO\x87\x15\xb9]\x89/\x9a\xc3\xf6\xde\x00\x80\x80\x00\u07d4\x8d\xa1\xd3Y\xbal\xb4\xbc\xc5}zCw\xd5]\xb2\xf0\x1cr\x89\x04V9\x18$O@\x00\x00\u07d4\x8d\xab\x94\x8a\xe8\x1d\xa3\x01\xd9r\xe3\xf6\x17\xa9\x12\xe5\xa7Sq.\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x8d\xad\xdfR\xef\xbdt\u0695\xb9i\xa5GoO\xbb\xb5c\xbf\u0489-
C\xf3\xeb\xfa\xfb,\x00\x00\u07d4\x8d\xb1\x85\xfe\x1bp\xa9Jj\b\x0e~#\xa8\xbe\xdcJ\xcb\xf3K\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\x8d\xb5\x8e@n -
\xf9\xbcp<H\v\xd8\xed$\x8dR\xa02\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8d\xbc>l\xb43\xe1\x94\xf4\x0f\x82\xb4\x0f\xaa\xdb\x1f\x8b\x85a\x16\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\x8d\xc1\xd5\x11\x1d\t\xaf%\xfd\xfc\xacE\\|\xec(>mgu\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8d\u0504\xff\x8a0sd\xebf\xc5%\xa5q\xaa\xc7\x01\xc5\xc3\x18\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8d\u05a9\xba\xe5\u007fQ\x85I\xad\xa6wFo\ua2b0O\u0674\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8d\xde<\xb8\x11\x85h\xefE\x03\xfe\x99\x8c\xcd\xf56\xbf\x19\xa0\x98\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8d\xde`\xeb\b\xa0\x99\xd7\u06a3V\u06aa\xb2G\r{\x02Zk\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94\x8d\xf39!Kj\u0472Fc\xceq`4t\x9dn\xf88\u064a\x02TO\xaaw\x80\x90\xe0\x00\x00\xe0\x94\x8d\xf5=\x96\x19\x14q\xe0Y\xdeQ\xc7\x18\xb9\x83\xe4\xa5\x1d*\xfd\x8a\x06\u01b95\xb8\xbb\xd4\x00\x00\x00\u07d4\x8d\xfb\xaf\xbc\x0e[\\\x86\xcd\x1a\u0597\xfe\xea\x04\xf41\x88\u0796\x89\x15%+\u007f_\xa0\xde\x00\x00\u07d4\x8e\a;\xad%\xe4\"\x18a_J\x0ek.\xa8\xf8\xde\"0\xc0\x89\x82=b\x9d\x02k\xfa\x00\x00\u07d4\x8e\x0f\xee8hZ\x94\xaa\xbc\xd7\u0385{k\x14\t\x82Ou\xb8\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x8e#\xfa\xcd\x12\xc7e\xc3j\xb8\x1am\xd3M\x8a\xa9\xe6\x89\x18\xae\x89\t\x11\u418d\xba\x9b\x00\x00\xe0\x94\x8e/\x904\xc9%G\x19\u00ceP\u026ad0^\u0596\xdf\x1e\x8a\x01\x00N.E\xfb~\xe0\x00\x00\u07d4\x8e2@\xb0\x81\x0e\x1c\xf4\a\xa5\x00\x80G@\u03cdad2\xa4\x89\x02/fU\xef\v8\x80\x00\u07d4\x8eHj\x04B\xd1q\xc8`[\xe3H\xfe\xe5~\xb5\b^\xff\r\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8e

aV3k\xe2\u037e2\x14\r\xf0\x8a+\xa5_\u0425\x84c\x89\x04\t\x9e\x1dcW\x18\x00\x00\u07d4\x8eg\
b\x15\xfbg\xae\xae\xa5{\x86SN\xdc\x00\xcd\xf5d\xfe\u5272\xe4\xb3#\xd9\xc5\x10\x00\x00\u07d4\
x8emt\x85\xcb\u942c\xc1\xad\x0e\xe9\xe8\xcc\xf3\x9c\f\x93D\x0e\x893\xc5I\x901r\f\x00\x00\xe0\x
94\x8et\xe0\u0477~\xbc\x82:\xca\x03\xf1\x19\x85L\xb1
'\xf6\u05ca\x16\xb3R\xda^\x0e\xd3\x00\x00\x00\u07d4\x8ex\xf3QE}\x01oJ\xd2u^\xc7BN\\!\xbamQ
\x89\a\xea(2uw\b\x00\x00\u07d4\x8ey6\u0552\x00\x8f\xdcz\xa0N\xde\xebuZ\xb5\x13\u06f8\x9d\x
89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x8e\u007f\xd28H\xf4\xdb\a\x90j}\x10\xc0K!\x80;\xb0\x8
2'\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x8e\x92\xab\xa3\x8er\xa0\x98\x17\v\x92\x95\x92FSz.U
V\xc0\x89\x0e~\xeb\xa3A\vt\x00\x00\u07d4\x8e\x98ve$\xb0\xcf'G\xc5\r\xd4;\x95gYM\x971\u0789l
D\xb7\xc2a\x82(\x00\x00\u07d4\x8e\x9b5\xadJ\n\x86\xf7XDo\xff\xde4&\x9d\x94\f\xea\u0349\xd8\x
d7&\xb7\x17z\x80\x00\x00\u07d4\x8e\x9c\b\xf78f\x1f\x96v#n\xff\x82\xbaba\xdd?H\"\x89\x05k\xc7^
-
c\x10\x00\x00\u07d4\x8e\x9cB\x92f\xdf\x05~\xfax\xdd\x1d_w\xfc@t*\xd4f\x89\x10D.\u0475l|\x80\x
00\u07d4\x8e\xa6V\xe7\x1e\xc6Q\xbf\xa1|ZWY\xd8`1\xcc5\x99w\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\x8e\xae)CU\x98\xba\x8f\x1c\x93B\x8c\xdb>+M1\a\x8e\x00\x89lk\x93[\x8b\xb
d@\x00\x00\u07d4\x8e\xb1\xfb\xe4\xe5\xd3\x01\x9c\xd7\xd3\r\xae\x9c\r[Lv\xfbc1\x89lk\x93[\x8b\x
bd@\x00\x00\u07d4\x8e\xb5\x17t\xaf
k\x96k\x89\t\xc4Z\xa6r'H\x80,\f\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x8e\xb8\xc7\x19\x82
\xa0\x0f\xb8Bu)2S\xf8\x04ED\xb6kI\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\x8e\xcb\u03ec\x
bf\xaf\xe9\xf0\f9\"\xa2N,\xf0\x02gV\xca \x8a\x011\xbe\xb9%\xff\xd3
\x00\x00\u07d4\x8e\u03b2\xe1$Sl[_\xfcd\x0e\xd1O\xf1^\u0668\xcbq\x89lk\x93[\x8b\xbd@\x00\x00
\u07d4\x8e\u042f\x11\xff(p\xda\x06\x81\x00J\xfe\x18\xb0\x13\xf7\xbd8\x82\x89\xd8\xd7&\xb7\x17
z\x80\x00\x00\u0794\x8e\xd1Cp\x1f/r(\x0f\xd0J{Ad(\x19y\xea\x87\u0248\xc2I\xfd\xd3'x\x00\x00\u0
7d4\x8e\xd1R\x8bD~\xd4)y\x02\xf69\xc5\x14\u0414J\x88\xf8\u0209\n\xc6\xe7z\xb6c\xa8\x00\x00\
u07d4\x8e\xd4(L\x0fGD\x9c\x15\xb8\u0673$]\u8fb6\u0380\xbf\x89+^:\xf1k\x18\x80\x00\x00\xe0\x
94\x8e\xde~=\xc5\aI\xc6\xc5\x0e.(\x16\x84x\xc3M\xb8\x19F\x8a\x04<0\xfb\b\x84\xa9l\x00\x00\u0
7d4\x8e\xe5\x843}\xdb\xc8\x0f\x9e4\x98\xdfU\xf0\xa2\x1e\xac\xb5\u007f\xb1\x89\x01\x15\x8eF\t\x
13\xd0\x00\x00\u07d4\x8e\xeb\xec\x1ab\xc0\x8b\x05\xa7\xd1\u0551\x80\xaf\x9f\xf0\u044e?6\x89\
x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x8e\xf4\u0622\xc2<Ry\x18{d\xe9ot\x14\x04\bS\x85\xf3\x
89\x10=\xc1\xe9\xa9i{\x00\x00\u07d4\x8e\xf7\x11\xe4:\x13\x91\x8f\x13\x03\xe8\x1d\x0e\xa7\x8c\x
9e\xef\xd6~\xb2\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8e\xfe\xc0X\xccTaWvjc'u@J3J\xaa\
u0687\x89l]\xb2\xa4\xd8\x15\xdc\x00\x00\u07d4\x8f\x02\xbd\xa6\xc3i\"\xa6\xbejP\x9b\xe5\x19\x0
6\u04d3\xf7\xb9\x9b\x897I\r\xc1.\xbe\u007f\x80\x00\u07d4\x8f\x058\xedq\xda\x11U\xe0\xf3\xbd\x
e5f|\xeb\x841\x8a\x1a\x87\x89i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94\x8f\x06||\x1b\xbdWx\v{\x9e\xe
b\x9e\xc0\x03/\x90\xd0\xdc\xf9\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\x8f\n\xb8\x94\xbd?Ni}\
xbc\xfb\x85\x9dIz\x9b\xa1\x95\x99J\x8a\b]c\x8beG*\xa2\x00\x00\u07d4\x8f\n\xf3uf\xd1R\x80/\x1a\
xe8\xf9(\xb2Z\xf9\xb19\xb4H\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x8f\x19R\xee\xd1\xc5H\xd9\ue6d7\xd0\x16\x9a\a\x93;\xe6\x9fc\x8965\u026d\xc5\
u07a0\x00\x00\xe0\x94\x8f\x1f\xcc<Q\xe2R\xb6\x93\xbc[\x0e\xc3\xf65)\xfei(\x1e\x8a\x01EB\xba\x
12\xa37\xc0\x00\x00\u07d4\x8f\"`\x96\xc1\x84\xeb\xb4\x01\x05\xe0\x8d\xacM\"\xe1\xc2\xd5M0\x8
9\x10\x9eC{\xd1a\x8c\x00\x00\xe0\x94\x8f)\xa1J\x84Z\xd4X\xf2\xd1\b\xb5h\xd8\x13\x16k\xcd\xf4
w\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x8f1\xc7\x00Q\x97\xec\x99z\x87\xe6\x9b\xec

Hd\x9a\xb9K\xb2\xa5\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8fA\xb1\xfb\xf5B\x98\xf5\u043c-
\x12/N\xb9]\xa4\xe5\xcd=\x89\x133\x83/^3\\\x00\x00\u07d4\x8fG2\x8e\xe02\x01\xc9\xd3^\u04b5A+%\xde\u0305\x93b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8fG=\n\xb8v\u076a\x15`\x86!\xd7\x01>o\xf7\x14\xb6u\x89\x19\x80\x1c\x83\xb6\xc7\xc0\x00\x00\u07d4\x8fM\x1dAi>F,\xf9\x82\xfd\x81\u042ap\x1d:St\u0249\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8fM\x1e~Ea(J4\xfe\xf9g<\r4\xe1*\xf4\xaa\x03\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8fO\xb1\xae\xa7\xcd\x0fW\x0e\xa5\xe6\x1b@\xa4\xf4Q\vbd\xe4\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x8fV\x1bA\xb2\t\xf2H\u0229\x9f\x85\x87\x887bP`\x9c\xf3\x89\\(=A\x03\x94\x10\x00\x00\xe0\x94\x8fX\xd84\x8f\xc1\xdcN\r\xd84;eC\xc8W\x04^\xe9@\x8a\x02\xe3\x03\x8d\xf4s\x03(\x00\x00\u07d4\x8f`\x89_\xbe\xbb\xb5\x01\u007f\xcb\xff<\u0763\x97)+\xf2[\xa6\x89\x17D\x06\xff\x9fo\u0480\x00\u07d4\x8fd\xb9\xc1$m\x85x1d1\a\xd3U\xb5\xc7_\xef]O\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94\x8ff\x0f\x8b.L|\u00b4\xac\x9cG\xed(P\x8d_\x8f\x86P\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x8fi\xea\xfd\x023\xca\xdb@Y\xabw\x9cF\xed\xf2\xa0PnH\x89`\xf0f
\xa8IE\x00\x00\xe0\x94\x8fq~\xc1U/LD\x00\x84\xfb\xa1\x15J\x81\xdc\x00>\xbd\xc0\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\x8f\x8a\xcb\x10v\a8\x84y\xf6K\xaa\xab\xea\x8f\xf0\a\xad\xa9}\x8a\x05\xc6\xf3\b\n\xd4#\xf4\x00\x00\u07d4\x8f\x8c\xd2n\x82\xe7\xc6\xde\xfd\x02\u07ed\a\x97\x90!\xcb\xf7\x15\f\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x8f\x8f7\u042d\x8f3]*q\x01\xb4\x11V\xb6\x88\xa8\x1a\x9c\xbe\x89\x03\xcbq\xf5\x1f\xc5X\x00\x00\u07d4\x8f\x92\x84O(*\x92\x99\x9e\u5d28\xd7s\xd0kiM\xbd\x9f\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\x8f\xact\x8fxJ\x0f\xedh\u06e43\x19\xb4*u\xb4d\x9cn\x891T\xc9r\x9d\x05x\x00\x00\u07d4\x8f\u0665\xc3:}\x9e\xdc\xe0\x99{\xdfw\xab0d$\xa1\x1e\xa9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x8f\xef\xfa\xdb8z\x15G\xfb(M\xa9\xb8\x14\u007f>|m\xc6\u0689-
b{\xe4S\x05\b\x00\x00\u07d4\x8f\xf4`Ehw#\xdc3\xe4\u0419\xa0i\x04\xf1\ubd44\u0709lk\x93[\x8b\xbd@\x00\x00\u07d4\x8f\xfa\x06!\"\xac0t\x18\x82\x1a\u06d3\x11\aZ7\x03\xbf\xa3\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x8f\xfe2)\x97\xb8\xe4\x04B-
\x19\xc5J\xad\xb1\x8f[\xc8\u9dc9\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x90\x01\x94\u0131\aC\x05\u045d\xe4\x05\xb0\xacx(\x0e\xca\xf9g\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x90\x03\xd2p\x89\x1b\xa2\xdfd=\xa84\x15\x83\x195E\xe3\xe0\x00\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x90\x05z\xf9\xaaf0~\xc9\xf03\xb2\x97$\u04f2\xf4\x1e\xb6\xf9\x8a\x19\xd1\u05aa\xdb,R\xe8\x00\x00\u07d4\x90\x0f\v\x8e5\xb6h\xf8\x1e\xf2R\xb18U\xaaP\a\xd0\x12\xe7\x89\x17\n\x0fP@\xe5\x04\x00\x00\u07d4\x90\x18\xcc\x1fH\xd20\x8e%*\xb6\b\x9f\xb9\x9a|\x1dV\x94\x10\x89\n\u05ce\xbcZ\xc6\x00\x00\u07d4\x90\x1d\x99\xb6\x99\xe5\u0191\x15\x19\xcb
v\xb4\xc7c0\xc5M\"\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x90-
t\xa1W\xf7\u04b9\xa37\x8b\x1fVp70\xe0:\x17\x19\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x904\x13\x87\x8a\xea;\xc1\bc\t\xa3\xfev\x8beU\x9e\x8c\xab\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x90If\xcc\"\x13\xb5\xb8\xcb[\xd6\b\x9e\xf9\xcd\xdb\xef~\xdf\u0309lk\x93[\x8b\xbd@\x00\x00\u07d4\x90L\xaaB\x9ca\x9d\x94\x0f\x8egA\x82j\r\xb6\x92\xb1\x97(\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x90R\xf2\xe4\xa3\xe3\xc1-
\xd1\xc7\x1b\xf7\x8aN\xc3\x04=\u020b~\x89\x0e~\xeb\xa3A\vt\x00\x00\u0794\x90U&V\x8a\xc1#\xaf\xc0\xe8J\xa7\x15\x12O\xeb\xe8=\xc8|\x88\xf8i\x93)g~\x00\x00\u07d4\x90\x92\x91\x87\a\xc6!\xfd\xbd\x1d\x90\xfb\x80\xebx\u007f\xd2osP\x89\x85[[\xa6\\\x84\xf0\x00\x00\u07d4\x90\x9b^v:9\xdc\

u01d5\"=s\xa1\u06f7\xd9L\xa7Z\u0209lk\x93[\x8b\xbd@\x00\x00\u07d4\x90\xac\xce\xd7\xe4\x8c\
b\u01b94dm\xfa\n\xdf)\u0714\aO\x89\x03\vK\x15{\xbdI\x00\x00\u07d4\x90\xb1\xf3p\xf9\xc1\xeb\v\
xe0\xfb\x8e+\x8a\xd9jAcq\u074a\x890\xca\x02O\x98{\x90\x00\x00\u07d4\x90\xb6/\x13\x1a_)\xb4
UqQ>\xe7\xa7J\x8f\v#\"\x02\x89\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u07d4\x90\xbdb\xa0P\x84Ra\xf
aJ\x9f|\xf2A\xeac\v\x05\ufe09\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x90\xc4\x1e\xba\x00\x8e
\xcb\xe9'\xf3F`?\u0206\x98\x12Yi\x89\x02F\xdd\xf9yvh\x00\x00\u07d4\x90\u0480\x9a\xe1\xd1\xff\
xd8\xf6>\xda\x01\xdeI\xddU-
\xf3\u047c\x89\u063beI\xb0+\xb8\x00\x00\u07d4\x90\xdc\t\xf7\x17\xfc*[i\xfd`\xba\b\xeb\xf4\v\xf4\x
e8$l\x89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4\x90\xe3\x00\xacqE\x1e@\x1f\x88\u007fnw(\x85\x16
G\xa8\x0e\a\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x90\xe3Z\xab\xb2\xde\xef@\x8b\xb9\xb5\
xac\xefqDW\xdf\xdebr\x89\x05l\xd5_\xc6M\xfe\x00\x00\u07d4\x90\xe7\a\x0fM\x03?\xe6\x91\f\x9e\
xfeZ'\x8e\x1f\xc6#M\xef\x89\x05q8\b\x19\xb3\x04\x00\x00\u07d4\x90\xe9>M\xc1q!HyR36\x14\x00
+\xe4#VI\x8e\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\x90\u9a68.\u06a8\x14\u0084\xd22\xb6\u9e90p\x1dIR\x89\x05k\xe0<\xa
3\xe4}\x80\x00\u07d4\x90\xf7t\xc9\x14}\u0790\x85=\xdcC\xf0\x8f\x16\xd4U\x17\x8b\x8c\x89\xd8\x
d7&\xb7\x17z\x80\x00\x00\u07d4\x90\xfcS{!\x06Xf\n\x83\xba\xa9\xacJ\x84\x02\xf6WF\xa8\x89e\x
ea=\xb7UF`\x00\x00\u07d4\x91\x05\n\\\xff\xad\xed\xb4\xbbn\xaa\xfb\xc9\xe5\x014(\xe9l\x80\x89\\(
=A\x03\x94\x10\x00\x00\u07d4\x91\x05\x17d\xafk\x80\x8eB\x12\xc7~0\xa5W.\xaa1pp\x8965\u026
d\xc5\u07a0\x00\x00\xe0\x94\x91\v}Wz~9\xaa#\xac\xf6*\xd7\xf1\xef4)4\xb9h\x8a\x02\x1e\x19\xe0
\u027a\xb2@\x00\x00\u07d4\x91\x0e\x99eC4Lh\x15\xfb\x97\u0367\xafK\x86\x98vZ[\x89\x05\x9a\
xf6\x98)\xcfd\x00\x00\u07d4\x91\x1f\xee\xa6\x1f\xe0\xedP\u0179\xe5\xa0\xd6`q9\x9d(\xbd\u0189\
x03@\xaa\xd2\x1b;p\x00\x00\u07d4\x91\x1f\xf23\xe1\xa2\x11\xc0\x17,\x92\xb4l\xf9\x97\x03\x05\x
82\xc8:\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x91
\xe7\x11s\xe1\xba\x19\xba\x8f\x9fO\xdb\u072a4\xe1\u05bbx\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\
x91!\x17\x12q\x9f+\bM;8u\xa8Pi\xf4f61A\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x91#\x04\x11\x8
b\x80G=\x9e\x9f\xe3\xeeE\x8f\xbea\x0f\xfd\xa2\xbb\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x91Tky\xec\xf6\x9f\x93kZV\x15\b\xb0\xd7\xe5\f\u0159/\x89\x0e~\xeb\xa3A\vt\x00\
x00\u07d4\x91V\u0440)5\x0eG\x04\b\xf1_\x1a\xa3\xbe\x9f\x04\ng\u018965\u026d\xc5\u07a0\x00
\x00\u07d4\x91b\x0f>\xb3\x04\xe8\x13\u048b\x02\x97Ume\xdcN]\u5a89\xcf\x15&@\xc5\xc80\x00\
x00\xe0\x94\x91k\xf7\xe3\xc5E\x92\x1d2\x06\xd9\x00\xc2O\x14\x12|\xbd^p\x8a\x03\xd0\u077c}\xf
2\xbb\x10\x00\x00\u0794\x91l\xf1}qA(\x05\xf4\xaf\xc3DJ\v\x8d\xd1\xd93\x9d\x16\x88\xc6s\xce<@
\x16\x00\x00\u07d4\x91{\x8f\x9f:\x8d\t\xe9
,R\u009erA\x96\xb8\x97\xd3^\x89\b\xbaR\xe6\xfcE\xe4\x00\x00\u07d4\x91\x89g\x91\x8c\u0617\x
dd\x00\x05\xe3m\xc6\u0203\xefC\x8f\xc8\u01c9\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4\x91\x89\x
8e\xab\x8c\x05\xc0\"(\x83\xcdM\xb2;w\x95\xe1\xa2J\u05c9lk\x93[\x8b\xbd@\x00\x00\u0794\x91\x
91\xf9F\x98!\x05\x16\xcfc!\xa1B\a\x0e
Yvt\xed\x88\xee\x9d[\xe6\xfc\x11\x00\x00\u07d4\x91\xa4\x14\x9a,{\x1b:g\xea(\xaf\xf3G%\u0fcdu$\
x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\x91\xa7\x87\xbcQ\x96\xf3HW\xfe\f7/M\xf3v\xaa\xa7f\x13\x
89lk\x93[\x8b\xbd@\x00\x00\u07d4\x91\xa8\xba\xae\xd0\x12\xea.c\x80;Y=\r\f*\xabL[\n\x89QP\xae
\x84\xa8\xcd\xf0\x00\x00\u07d4\x91\xac\\\xfeg\xc5J\xa7\xeb\xfb\xa4HflF\x1a;\x1f\xe2\xe1\x89\x15\
xc94\x92\xbf\x9d\xfc\x00\x00\u07d4\x91\xbb?y\x02+\xf3\xc4S\xf4\xff%n&\x9b\x15\xcf,\x9c\xbd\x8
9RX\\\x13\xfe:\\\x00\x00\u07d4\x91\xc7^<\xb4\xaa\x89\xf3F\x19\xa1d\xe2\xa4x\x98\xf5gM\x9c\x89

lk\x93[\x8b\xbd@\x00\x00\u07d4\x91\xc8\f\xaa\b\x1b85\x1d*\x0e\x0e\x00\xf8\n4\xe5dt\xc1\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x91\xccF\xaa7\x9f\x85jf@\xdc\xcdZd\x8ay\x02\xf8I\u0649\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x91\u04a9\xee\x1am\xb2\x0fS\x17\u0327\xfb\xe218\x95\u06ce\xf8\x8a\x01\xcc\u00e5/0n(\x00\x00\u07d4\x91\xd6n\xa6(\x8f\xaaK=`l*\xa4\\{k\x8a%'9\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x91\u06f6\xaa\xad\x14\x95\x85\xbeG7\\]m\xe5\xff\t\x19\x15\x18\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x91\xe8\x81\x06R\xe8\xe6\x16\x15%\xd6;\xb7u\x1d\xc2\x0fg`v\x89'Mej\xc9\x0e4\x00\x00\u07d4\x91\xf5\x16\x14l\xda
(\x17\x19\x97\x80`\u01beAI\x06|\x88\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x91\xf6$\xb2J\x1f\xa5\xa0V\xfeW\x12)\xe77\x9d\xb1K\x9a\x1e\x8a\x02\x8a\x85\x17\xc6i\xb3W\x00\x00\xe0\x94\x91\xfe\x8aLad\u07cf\xa6\x06\x99]k\xa7\xad\xca\xf1\u0213\u038a\x03\x99\x92d\x8a#\u0220\x00\x00\u07d4\x92\x1fRa\xf4\xf6\x12v\a\x06\x89&%\xc7^{\u0396\xb7\b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x92!\xc9\xce\x01#&et\x10\x96\xac\a#Y\x03\xad\x1f\xe2\xfc\x89\x06\xdbc3U\"b\x80\x00\u07d4\x92%\x988`\xa1\xcbF#\xc7$\x80\xac\x16'+\f\x95\xe5\xf5\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x92%\xd4jZ\x80\x949$\xa3\x9e[\x84\xb9m\xa0\xacE\x05\x81\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\x92*
\u01da\x1d:&\xdd8)g{\xf1\xd4\\\x8fg+\xb6\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x92C\x8eR\x03\xb64o\xf8\x86\xd7\xc3b\x88\xaa\xcc\xccx\xce\u028965\u026d\xc5\u07a0\x00\x00\u07d4\x92C\xd7v-
w({\x12c\x86\x88\xb9\x85N\x88\xa7i\xb2q\x8965\u026d\xc5\u07a0\x00\x00\u0794\x92K\xcez\x85<\x97\v\xb5\xec{\xb7Y\xba\xeb\x9ct\x10\x85{\x88\xbe -
j\x0e\xda\x00\x00\u07d4\x92N\xfam\xb5\x95\xb7\x93\x13'~\x881\x96%\akX\n\x10\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x92U\x82&\xb3\x84bl\xadH\xe0\x9d\x96k\xf19^\xe7\xea]\x89\x12\x1e\xa6\x8c\x11NQ\x00\x00\u07d4\x92`\x82\xcb~\xedK\x19\x93\xad$ZGrg\xe1\xc3<\xd5h\x89\x14Jt\xba\u07e4\xb6\x00\x00\u07d4\x92b\t\xb7\xfd\xa5N\x8d\u06dd\x9eM=\x19\xeb\u070e\x88\u009f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x92h\xd6&FV6\x11\xdc;\x83*0\xaa#\x94\xc6F\x13\xe3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x92i\x8e4Sx\xc6-
\x8e\xda\x18M\x946j\x14K\f\x10[\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\x92y:\u0173rhwJq0\xde+\xbd3\x04\x05f\x17s\x89\x02,\xa3X|\xf4\xeb\x00\x00\xe0\x94\x92y\xb2\"\x8c\xec\x8f{M\xda?2\x0e\x9a\x04f\xc2\xf5\x85\u028a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\x92|\xb7\xdc\x18p6\xb5B{\xc7\xe2\x00\xc5\xecE\f\x1d'\u0509\v\xb5\x9a'\x95<`\x00\x00\u07d4\x92|\u00bf\xda\x0e\b\x8d\x02\xef\xf7\v8\xb0\x8a\xa5<\xc3\tA\x89do`\xa1\xf9\x866\x00\x00\xe0\x94\x92\x84\xf9m\xdbG\xb5\x18n\xe5X\xaa12M\xf56\x1c\x0fs\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07d4\x92\x9d6\x8e\xb4j-
\x1f\xbd\xc8\xff\xa0`~\xdeK\xa8\x8fY\xad\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x92\xa7\u0166Cb\xe9\xf8B\xa2=\xec\xa2\x105\x85\u007f\x88\x98\x00\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\x92\xa8\x98\xd4o\x19q\x9c8\x12j\x8a<'\x86z\xe2\xce\u5589lk\x93[\x8b\xbd@\x00\x00\u07d4\x92\xa9q\xa79y\x9f\x8c\xb4\x8e\xa8G]r\xb2\xd2GAr\xe6\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x92\xaa\xe5\x97h\xed\xdf\xf8<\xfe`\xbbQ.s\n\x05\xa1a\u05c9\\\x97xA\fv\u0440\x00\u07d4\x92\xad\x1b=u\xfb\xa6}Tf=\xa9\xfc\x84\x8a\x8a\xde\x10\xfag\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x92\xae[|~\xb4\x92\xff\x1f\xfa\x16\xddB\xad\x9c\xad@\xb7\xf8\u0709.\xe4IU\b\x98\xe4\x00\x00\u07d4\x92\xc0\xf5s\xec\xcfb\xc5H\x10\xeek\xa8\xd1\xf1\x13T+0\x1b\x89\xb7ro\x16\u0331\xe0\x00\x00\u07d4\x92\xc1?\x

e0\xd6\u0387\xfdP\xe0=\uf7e6@\x05\t\xbdps\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\x92\xc9L(
\xdf\xcfqV\xe6\xf10\x88\xec\u754b6v\xfd\x89\x05-
T(\x04\xf1\xce\x00\x00\u07d4\x92\xcf\xd6\x01\x88\xef\u07f2\xf8\xc2\xe7\xb1i\x8a\xbb\x95&\xc1Q\
x1f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x92\u062d\x9aMah;\x80\u0526g.\x84\xc2\rbB\x1e\x80\x8
9\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x92\u0725\xe1\x02\xb3\xb8\x1b`\xf1\xa5\x04cIG\xc3t\xa
8\x8c\u02c9lk\x93[\x8b\xbd@\x00\x00\u07d4\x92\xe454\x0e\x9d%<\x00%c\x89\xf5+\x06}U\x97Nv
\x89\x0e\x87?D\x13<\xb0\x00\x00\xe0\x94\x92\xe49(\x16\xe5\xf2\xef_\xb6X7\xce\xc2\xc22\\\xc6I\"
\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x92\xe6X\x1e\x1d\xa1\xf9\xb8F\xe0\x93G3=\xc
8\x18\xe2\u04acf\x89\xc5S%\xcat\x15\xe0\x00\x00\u07d4\x93\x1d\xf3M\x12%\xbc\xd4\"Nch\r\\L\t\
xbc\xe75\xa6\x89\x03\xaf\xb0\x87\xb8v\x90\x00\x00\u07d4\x93\x1f\xe7\x12\xf6B\a\xa2\xfdP\"r\x8
8CT\x8b\xfb\x8c\xbb\x05\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x93#_4\r(c\xe1\x8d/LR\x99e\x16\x1
3\x8d\"\x02g\x89\x04\x00.D\xfd\xa7\xd4\x00\x00\u07d4\x93%\x82U\xb3|\u007fX\xf4\xb1\x06s\xa9
2\xdd:\xfd\x90\xf4\xf2\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x93(\xd5\\\xcb?\xceS\x1f\x19\x93\x8
23\x9f\x0eWn\xe8@\xa3\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x93)\xff\xdc&\x8b\xab\u0788t
\xb3f@l\x81D[\x9b-
5\x89\x16\xe6/\x8cs\f\xa1\x80\x00\u07d4\x93+\x9c\x04\xd4\r*\xc80\x83\xd9B\x98\x16\x9d\xae\x81
\xab.\u0409lk\x93[\x8b\xbd@\x00\x00\u07d4\x9346\xc8G&U\xf6L:\xfa\xaf|Lb\x1c\x83\xa6+8\x8965
\u026d\xc5\u07a0\x00\x00\u0794\x93;\xf3?\x82\x99p+:\x90&B\xc3>\v\xfa\xea\\\x1c\xa3\x88\xd2\xf
1?w\x89\xf0\x00\x00\u07d4\x93@4\\\xa6\xa3\uaf77sc\xf2X`C\xf2\x948\xce\v\x89\x1c\xc8\x05\xda\
r\xff\xf1\x00\x00\xe0\x94\x93@\xb5\xf6x\xe4^\xe0^\xb7\b\xbbz\xbbn\xc8\xf0\x8f\x1bk\x8a\x01EB\x
ba\x12\xa37\xc0\x00\x00\xe0\x94\x93J\xf2\x1b~\xbf\xa4g\xe2\xce\xd6Z\xa3N\xdd:\x0e\xc7\x132\x
8a\a\x80\x1f>\x80\xcc\x0f\xf0\x00\x00\xe0\x94\x93PiDJj\x98M\xe2\bNFi*\xb9\x9fg\x1f\xc7'\x8a\x01
\xe7\xe4\x17\x1b\xf4\u04e0\x00\x00\xe0\x94\x93P~\x9e\x81\x19\xcb\xce\u068a\xb0\x87\xe7\xec\
xb0q8=i\x81\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4\x93g\x8a<W\x15\x1a\xebh\xef\xdcC\xe
fM6\xcbY\xa0\t\xf3\x89\x01\xa1*\x92\xbc<>\x00\x00\xe0\x94\x93m\xcf\x00\x01\x94\xe3\xbf\xf5\n\u
0174$:;\xa0\x14\xd6a\u060a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x93o8\x13\xf5\xf6\xa1;\
x8eO\xfe\xc8?\xe7\xf8&\x18jq\u0349\x1c0s\x1c\xec\x03
\x00\x00\u07d4\x93t\x86\x9dJ\x99\x11\xee\x1e\xafU\x8b\xc4\u00b6>\xc6:\xcf\u074965\u026d\xc5\
u07a0\x00\x00\u07d4\x93uc\u0628\x0f\u05657\xb0\xe6m
\xa0%%\xd5\u0606`\x89\x87\x86x2n\xac\x90\x00\x00\u07d4\x93v\xdc\xe2\xaf.\xc8\xdc\xdat\x1b~
sEfF\x81\xd96h\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x93\x86\x8d\xdb*yM\x02\xeb\xda/\xa4\x8
0|v\xe3`\x98X\u0709m\xee\x15\xfc|$\xa7\x80\x00\xe0\x94\x93\x9cC\x13\xd2(\x0e\xdf^\a\x1b\xce\x
d8F\x06?\n\x97]T\x8a\x19i6\x89t\xc0[\x00\x00\x00\xe0\x94\x93\xa6\xb3\xabB0\x10\xf9\x81\xa7H\
x9dJ\xad%\xe2b\\WA\x8a\x04F\x80\xfej\x1e\xdeN\x80\x00\u07d4\x93\xaa\x8f\x92\xeb\xff\xf9\x91\
xfc\x05^\x90ne\x1a\xc7h\xd3+\u02092\xf5\x1e\u06ea\xa30\x00\x00\u07d4\x93\xb4\xbf?\xdf\xf6\xd
e?NV\xbamw\x99\xdcK\x93\xa6T\x8f\x89\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\u07d4\x93\xbc}\x9aJ\
xbdD\u023b\xb8\xfe\x8b\xa8\x04\xc6\x1a\xd8\xd6Wl\x89\xd8\xd6\x11\x9a\x81F\x05\x00\x00\u07d
4\x93\xc2\xe6N]\xe5X\x9e\xd2P\x06\xe8C\x19n\xe9\xb1\xcf\v>\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x0
0\xe0\x94\x93\u020e-
\x88b\x1e0\xf5\x8a\x95\x86\xbe\xd4\t\x89\x99\xebg\u074a\x06\x9bZ\xfa\xc7P\xbb\x80\x00\x00\u0
7d4\x93\xe0\xf3~\xcd\xfb\x00\x86\xe3\xe8b\xa9p4D{\x1eM\xec\x1a\x89\x01\xa0Ui\r\x9d\xb8\x00\x
00\xe0\x94\x93\xe3\x03A\x1a\xfa\xf6\xc1\a\xa4A\x01\u026c[6\xe9\xd6S\x8b\x8a\r\xf9\xdd\xfe\xcd\

x03e@\x00\x00\u07d4\x93\xf1\x8c\xd2R`@v\x14\x88\xc5\x13\x17M\x1eycv\x8b,\x89\x82\xff\xac\x9a\u0553r\x00\x00\u07d4\x94\x0fqQ@P\x9f\xfa\xbf\x97EF\xfa\xb3\x90\"\xa4\x19R\u0489K\xe4\xe7&{j\xe0\x00\x00\u07d4\x94,k\x8c\x95[\xc0\u0608\x12g\x8a#g%\xb3'9\xd9G\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\x94=7\x86JJS}5\xc8\u0657#\xcdd\x06\xce%b\xe6\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x94C\x9c\xa9\xcc\x16\x9ay\u0520\x9c\xae^gvJo\x87\x1a!\x89\r\x02\xabHl\xed\xc0\x00\x00\xe0\x94\x94D\x9c\x01\xb3*\u007f\xa5Z\xf8\x10OB\xcd\xd8D\xaa\x8c\xbc@\x8a\x03\x81\x11\xa1\xf4\xf0<\x10\x00\x00\xe0\x94\x94E\xba\\0\xe9\x89a\xb8`$a\xd08]@\xfb\xd8\x03\x11\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x94O\a\xb9o\x90\xc5\xf0\xd7\xc0\u0140S1I\xf3\xf5\x85\xa0x\x89\x04\x02\xf4\xcf\xeeb\xe8\x00\x00\u07d4\x94T\xb3\xa8\xbf\xf9p\x9f\xd0\u1407~l\xb6\u0219t\xdb\u0589\x90\xf54`\x8ar\x88\x00\x00\u07d4\x94]\x96\xeaW>\x8d\xf7&+\xbf\xa5r\"\x9bK\x16\x01k\x0f\x89\vX\x9e\xf9\x14\xc1B\x00\x00\u07d4\x94^\x18v\x9d~\xe7'\xc7\x01?\x92\xde$\xd1\x17\x96\u007f\xf3\x17\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x94a'\x81\x03;W\xb1F\xeet\xe7S\xc6r\x01\u007fS\x85\xe4\x89\xc3(\t>a\xee@\x00\x00\xe0\x94\x94dJ\xd1\x16\xa4\x1c\xe2\xca\u007f\xbe\xc6\t\xbd\xefs\x8a*\xc7\u01ca\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\x94p\xcc6YE\x86\x82\x18!\xc5\u0256\xb6\xed\xc8;mZ2\x89\x01M\x11
\u05f1`\x00\x00\xe0\x94\x94u\xc5\x10\xec\x9a&\x97\x92GtL=\x8c;\x0e\v_D\u04ca\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x94~\x11\xe5\xea)\ro\u00f3\x80H\x97\x9e\f\xd4N\xc7\xc1\u007f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x94\x83\u064f\x14\xa3?\xdc\x11\x8d@9U\u00995\xed\xfc_p\x89\x18\xea;4\xefQ\x88\x00\x00\u07d4\x94\x911\xf2\x89C\x92\\\xfc\x97\xd4\x1e\f\xea\v&)s\xa70\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\x94\x9f\x84\xf0\xb1\xd7\u0127\xcfI\xee\u007f\x8b,J\x13M\xe3(x\x89%\"H\u07b6\xe6\x94\x00\x00\u07d4\x94\x9f\x8c\x10{\xc7\xf0\xac\xea\xa0\xf1pR\xaa\xdb\xd2\xf9s+.\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x94\xa7\u0368\xf4\x81\xf9\u061dB\xc3\x03\xae\x162\xb3\xb7\t\xdb\x1d\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x94\xa9\xa7\x16\x911|
d'\x1bQ\xc95?\xbd\xed5\x01\xa8\x89\xb5\x0f\u03ef\xeb\xec\xb0\x00\x00\u07d4\x94\xadK\xad\x82K\xd0\ub7a4\x9cX\u03bc\xc0\xff^\b4k\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\x94\xbb\xc6}\x13\xf8\x9e\xbc\xa5\x94\xbe\x94\xbcQp\x92\f0\xd9\xf3\x89\x04X\xff\xa3\x15\nT\x00\x00\u07d4\x94\xbe:\xe5Ob\xd6c\xb0\xd4\u031e\x1e\xa8\xfe\x95V\ua7bf\x89\x01C\x13,\xa8C\x18\x00\x00\xe0\x94\x94\xc0U\xe8X5z\xaa0\xcf
A\xfa\x90Y\xce\x16J\x1f\x91\x8a\x04<%\xe0\xdc\xc1\xbd\x1c\x00\x00\xe0\x94\x94\xc7B\xfdz\x8by\x06\xb3\xbf\xe4\xf8\x90O\xc0\xbe\\v\x803\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x94\xcaV\xdew\u007f\xd4S\x17\u007f^\x06\x94\xc4x\xe6j\xff\x8a\x84\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\x94\xd8\x10t\xdbZ\xe1\x97\u04bb\x13s\xab\x80\xa8}\x12\x1cK\u04ca\x01\xfd\x934\x94\xaa_\xe0\x00\x00\u07d4\x94\u06c0xs\x86\n\xac=Z\xea\x1e\x88^R\xbf\xf2\x86\x99T\x89\xae\x8ez\v\xb5u\xd0\x00\x00\u07d4\x94\xe1\xf5\u02db\x8a\xba\xce\x03\xa1\xa6B\x82VU;i\f#U\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x94\xef\x8b\xe4Pw\xc7\xd4\xc5e'@\u0794jbbOq?\x89\x05l\xf5Y:\x18\xf8\x80\x00\u07d4\x94\xf1?\x9f\b6\xa3\xee$7\xa8I\"\u0498M\xc0\xf7\xd5;\x89\xa2\xa02\x9b\u00ca\xbe\x00\x00\u07d4\x94\xf8\xf0W\xdb~`\xe6u\xad\x94\x0f\x15X\x85\u0464w4\x8e\x89\x15\xbeat\xe1\x91.\x00\x00\xe0\x94\x94\xfc\u03ad\xfe\\\x10\x9c^\xae\xafF-
C\x871B\u020e\"\x8a\x01\x045a\xa8\x82\x93\x00\x00\x00\u07d4\x95\x03N\x16!\x86Q7\xcdG9\xb3F\xdc\x17\xda:'\xc3N\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\x95\fh\xa4\t\x88\x15M#\x93\xff\xf8\xda|\u0369\x96\x14\xf7,\x89\xf9AF\xfd\x8d\xcd\xe5\x80\x00\xe0\x94\x95\x0f\xe9\xc6\xca\xd5\f\x18\xf1\x1a\x9e\xd9\xc4W@\xa6\x18\x06\x12\u040a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x95!\

x83\xcf\u04ce5.W\x9d6\xde\xce\u0171\x84P\xf7\xfb\xa0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x95'
\x8b\b\xde\xe7\xc0\xf2\xc8\xc0\xf7\"\xf9\xfc\xbb\xb9\xa5$\x1f\u0689\x82\x93\t\xf6O\r\xb0\x00\x00\
u07d4\x95,W\xd2\xfb\x19Q\a\xd4\xcd\\\xa3\x00wA\x19\u07ed/x\x89lk\x93[\x8b\xbd@\x00\x00\u07
d4\x955r\xf0\xeam\xf9\xb1\x97\xca\xe4\x0eK\x8e\xcc\x05lCq\u014965\u026d\xc5\u07a0\x00\x00\
u07d4\x95>\xf6R\xe7\xb7i\xf5=nxjX\x95/\xa9>\xe6\xab\u725b\ny\x1f\x12\x110\x00\x00\u07d4\x95
DpF1;/:^\x19\xb9H\xfd;\x8b\xed\xc8,q|\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\x95]\xb3\xb
7C`\xb9\xa2hg~s\u03a8!f\x8a\xf6\xfa\u038a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4\x95`\xe8\xacg\x
18\xa6\xa1\xcd\xcf\xf1\x89\xd6\x03\xc9\x06>A=\xa6\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x
95g\xa0\u0781\x1d\xe6\xff\t[~\xe6N\u007f\x1b\x83\xc2a[\x80\x89\x0e~\xeb\xa3A\vt\x00\x00\u07d4
\x95h\x1c\xda\xe6\x9b
I\xce\x10\x1e2\\u\x98\x92\xca\xc3\xf8\x11\x89\x9a\xe9*\x9b\xc9L@\x00\x00\xe0\x94\x95h\xb7\xd
euV(\xaf5\x9a\x84T=\xe25\x04\xe1^A\xe6\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\x95i\xc6:\x92
\x84\xa8\x05bm\xb3\xa3.\x9d#c\x93GaQ\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x95\x80\x9e\x8d\x
a3\xfb\xe4\xb7\xf2\x81\xf0\xb8\xb1q_B\x0f}}c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x95\x9fW\xfd\x
edj\xe3y\x13\xd9\x00\xb8\x1e_H\xa7\x93\"\xc6'\x89\r\xdb&\x10GI\x11\x80\x00\u07d4\x95\x9f\xf1\u
007f\x1dQ\xb4s\xb4@\x10\x05'U\xa7\xfa\x8cu\xbdT\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x95\xa
5w\xdc.\xb3\xael\xb9\xdf\xc7z\xf6\x97\xd7\xef\xdf\xe8\x9a\x01\x89\a_a\x0fp\xed
\x00\x00\u07d4\x95\xcbm\x8acy\xf9J\xba\x8b\x88ViV,MD\x8eV\xa7\x89lk\x93[\x8b\xbd@\x00\x00\
u07d4\x95\xd5PB{ZQLu\x1ds\xa0\xf6\u049f\xb6]\"\xed\x10\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x
95\u064d\f\x10i\x90\x8f\x06zR\xac\xac+\x8bSM\xa3z\xfd\x89oY\xb60\xa9)p\x80\x00\xe0\x94\x95\
xdfN4E\xd7f&$\u010e\xbat\u03de\nS\xe9\xf72\x8a\v\xdb\xc4\x1e\x03H\xb3\x00\x00\x00\u07d4\x9
5\xe6\xa5K-
_g\xa2JHu\xafu\x10|\xa7\xea\x9f\xd2\xfa\x89Hz\x9a0E9D\x00\x00\xe0\x94\x95\xe6\xf9=\xac\"\x8b
\xc7XZ%sZ\xc2\xd0v\xcc:@\x17\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x95\xe7ad$\xcd\ta\x
a7\x17'$t7\xf0\x06\x92r(\x0e\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x95\xe8\n\x82\xc2\f\xbe=
`$,\xb9-sX\x10\xd04\xa2\x89\x01\xc3.F?\u0539\x80\x00\u07d4\x95\xf6-
\x02C\xed\xe6\x1d\xad\x9a1e\xf59\x05'\rT\xe2B\x89WG=\x05\u06ba\xe8\x00\x00\u07d4\x95\xfbZ\
xfb\x14\xc1\uf6b7\xd1y\xc5\xc3\x00P?\xd6j^\xe2\x89\x01\xda\xf7\xa0+\r\xbe\x80\x00\u07d4\x96\x
10Y\"\x02\u0082\xab\x9b\u0628\x84Q\x8b>\v\xd4u\x817\x89\x0e\x87?D\x13<\xb0\x00\x00\xe0\x9
4\x96\x1cY\xad\xc7E\x05\u0446M\x1e\xcf\u02ca\xfa\x04\x12Y<\x93\x8a\bxg\x83&\xea\xc9\x00\x0
0\x00\u07d4\x96,\r\xec\x8a=FK\xf3\x9b\x12\x15\xea\xfd&H\n\xe4\x90\u0349l\x82\xe3\xea\xa5\x13
\xe8\x00\x00\u07d4\x96,\xd2*\x8e\xdf\x1eONU\xb4\xb1]\xdb\xfb]\x9dT\x19q\x89lk\x93[\x8b\xbd@\
x00\x00\u07d4\x963K\xfe\x04\xff\xfaY\x02\x13\xea\xb3e\x14\xf38\xb8d\xb76\x89\x15\xaf\x1dx\xb5
\x8c@\x00\x00\u07d4\x967\xdc\x12r=\x9cxX\x85B\uac02fO?\x03\x8d\x9d\x8965\u026d\xc5\u07a0
\x00\x00\u07d4\x96N\xabK'kL\u0618>\x15\xcar\xb1\x06\x90\x0f\xe4\x1f\u0389\x1b\x1a\xe4\xd6\x
e2\xefP\x00\x00\u07d4\x96b\xee\x02\x19&h+1\xc5\xf2\x00\xceEz\xbe\xa7ll\xe9\x89$Y\x0e\x85\x8
9\xebj\x00\x00\xe0\x94\x96l\x04x\x1c\xb5\xe6}\xde25\xd7\xf8b\x0e\x1a\xb6c\xa9\xa5\x8a\x10\r
P\xdacQ`\x00\x00\u07d4\x96pv\xa8w\xb1\x8e\xc1ZA[\xb1\x16\xf0n\xf3&E\u06e3\x89lk\x93[\x8b\x
bd@\x00\x00\u07d4\x96{\xfa\xf7bC\u0379@<g\xd2\xce\xef\xde\xe9\n?\xebs\x894\x9d\x87\xf2\xa2
\xdc/\x00\x00\u07d4\x96}AB\xafw\x05\x15\xddpb\xaf\x93l\x8d\xbf\xdf\xf2\x9f
\x89\x01\x18T\xd0\xf9\xce\xe4\x00\x00\u07d4\x96\x8b\x14d\x8f\x01\x833h|\xd2\x13\xfad\n\xec\x0
4\xcec#\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x96\x8d\xea`\xdf>\t\xae<\x8d5\x05\xe9\xc0\x80

EK\xe0\xe8\x19\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x96\x92A\x91\xb7\xdfe[3\x19\xdcma
7\xf4\x81\xa7:\x0f\xf3\x89\xd9\xec\xb4\xfd
\x8eP\x00\x00\u07d4\x96\x96\x05!83\x8cr/\x11@\x81\\\xf7t\x9d\r;:t\x89\x1b\x1a\xe4\xd6\xe2\xefP\
x00\x00\u07d4\x96\xa5_\x00\xdf\xf4\x05\xdcM\xe5\xe5\x8cW\xf6\xf6\xf0\xca\xc5]/\x89jf\x167\x9c\
x87\xb5\x80\x00\u07d4\x96\xaaW?\xed/#4\x10\u06eeQ\x80\x14[#\xc3\x1a\x02\xf0\x89]\u0212\xaa
\x111\xc8\x00\x00\u07d4\x96\xadW\x9b\xbf\xa8\u06ce\xbe\xc9\u0486\xa7.Fa\xee\xd8\xe3V\x89:\
v\xa4+\xeca\x83\x00\x00\u07d4\x96\xb44\xfe\x06W\xe4*\u0302\x12\xb6\x86Q9\xde\xde\x15\x97\x
9c\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x96\xb9\x06\xear\x9fFU\xaf\xe3\xe5}5'|\x96}\xfa\x1
5w\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x96\xd6-
\xfdF\b\u007fb@\x9d\x93\xdd`a\x88\xe7\x0e8\x12W\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x96\xd9\
u0328\xf5^\xea\x00@\xecn\xb3H\xa1wK\x95\xd9>\xf4\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\
x96\xe7\xc0\xc9\u057f\x10\x82\x1b\xf1@\xc5X\xa1E\xb7\xca\xc2\x13\x97\x899>\xf1\xa5\x12|\x80\
x00\x00\u07d4\x96\xeaj\u021a+\xac\x954{Q\u06e6=\x8b\xd5\xeb\xde\xdc\xe1\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4\x96\xea\xfb\xf2\xfboM\xb9\xa46\xa7LE\xb5eDR\xe28\x19\x89\x01\x15\x8eF\t\x
13\xd0\x00\x00\u07d4\x96\xebR>\x83/P\n\x01}\xe1>\xc2\u007f]6lV\x0e\xff\x89\x10\xac\u03baC\xe
e(\x00\x00\u07d4\x96\xf0F*\xe6\xf8\xb9`\x88\xf7\xe9\u018ct\xb9\u062d4\xb3G\x89a\t=|,m8\x00\x0
0\u07d4\x96\xf8 P\vp\xf4\xa3\xe3#\x9da\x9c\xff\x8f\" u\xb15\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x96\xfeY\xc3\u06f3\xaa|\xc8\xcbbH\fe\xe5nb\x04\xa7\xe2\x8a\x04<3\xc1\x93ud\
x80\x00\x00\u07d4\x96\xffoP\x99h\xf3l\xb4,\xbaH\xdb2\xf2\x1fVv\xab\xf8\x89j\xcb=\xf2~\x1f\x88\x
00\x00\u07d4\x97\t8R*\xfb^\x8f\x99Hs\xc9\xfb\xdc&\xe3\xb3~1L\x8965\u026d\xc5\u07a0\x00\x00\
u07d4\x97\n\xbdS\xa5O\xcaJd) |\x18-
MW\xbb9\u0520\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x97\r\x8b\x8a\x00\x16\xd1C\x05O\x14\x9f\x
b3\xb8\xe5P\xdc\a\x97\u01c965\u026d\xc5\u07a0\x00\x00\u07d4\x97,/\x96\xaa\x00\u03ca/
Z\xbc\xf8\x93|\fu\xf5\xd8\u0649\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x97?N6\x1f\xe5\xde\u0358\x9dL\x8f}|\xc9y\x908]\xaf\x89\x15\x0f\x85C\xa3\x87B\x
00\x00\u07d4\x97M\x05A\xabJG\xec\u007fu6\x9c\x00i\xb6J\x1b\x81w\x10\x89\x15\xaf\x1dx\xb5\x
8c@\x00\x00\u0794\x97M/\x17\x89_)\x02\x04\x9d\xea\xae\xcf\t\xc3\x04e\a@-
\x88\xcc\x19\u00947\xab\x80\x00\u07d4\x97R\xd1O^\x10\x93\xf0qq\x1c\x1a\xdb\xc4\xe3\xeb\x1e\
\W\xf3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x97V\xe1v\xc9\xefi>\xe1\xee\u01b9\xf8\xb1Q\xd3\x13
\xbe\xb0\x99\x89A\rXj
\xa4\xc0\x00\x00\u07d4\x97_7d\xe9{\xbc\xcfv|\xbd;y[\xa8m\x8b\xa9\x84\x0e\x89\x12\xc1\xb6\xee\
xd0=(\x00\x00\xe0\x94\x97j\x18Sj\xf4\x18tBc\b\x87\x1b\xcd\x15\x12\xa7u\xc9\xf8\x8a\x02\x1e\x1
9\xe0\u027a\xb2@\x00\x00\u07d4\x97n<\xea\xf3\xf1\xafQ\xf8\u009a\xff]\u007f\xa2\x1f\x03\x86\xd
8\xee\x89\r\x02\xabHl\xed\xc0\x00\x00\xe0\x94\x97w\xcca\xcfuk\xe3\xb3\xc2\f\xd4l\x1ci\xd2u\xe7\
xa1
\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x97\x81\v\xaf\xc3~\x840c2\xaa\xcb5\xe9*\xd9\x
11\xd2=$\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x97\x8cC\f\xe45\x9b\x06\xbc,\xdf\\)\x85\xfc\x95\
x0eP\xd5\u0209\x1a\x05V\x90\xd9\u06c0\x00\x00\u07d4\x97\x95\xf6C\x19\xfc\x17\xdd\x0f\x82a\x
f9\xd2\x06\xfbf\xb6L\xd0\u0249\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x97\x99\xca!\xdb\xcfi\xbf\xa1\xb3\xf7+\xacQ\xb9\xe3\xcaX|\xf9\x89\\(=A\x03\x94\x
10\x00\x00\u07d4\x97\x9c\xbf!\xdf\xec\x8a\xce?\x1c\x19m\x82\u07d6%4\xdf9O\x89\x99\x91\xd4x\
xddM\x16\x00\x00\u07d4\x97\x9dh\x1ca}\xa1o!\xbc\xac\xa1\x01\xed\x16\xed\x01Z\xb6\x96\x89e\x

ea=\xb7UF`\x00\x00\u07d4\x97\x9f0\x15\x8bWK\x99\x9a\xab4\x81\a\xb9\xee\xd8[\x1f\xf8\xc1\x894\x95tD\xb8@\xe8\x00\x00\u07d4\x97\xa8o\x01\xce?|\xfdDA3\x0e\x1c\x9b\x19\xe1\xb1\x06\x06\xef\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x97\xb9\x1e\xfesP\xc2\xd5~~@k\xab\x18\xf3a{\xcd\xe1J\x8a\x02\x1e\x19\x99\xbb\xd5\u04be\x00\x00\u07d4\x97\xd0\xd9r^;p\xe6u\x841s\x93\x8e\xd3q\xb6,\u007f\xac\x89\t79SM(h\x00\x00\u07d4\x97\xd9\xe4jv\x04\u05f5\xa4\xeaN\xe6\x1aB\xb3\xd25\x0f\xc3\xed\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x97\xdc&\xecg\n1\xe0\"\x1d*u\xbc]\xc9\xf9\f\x1fo\u0509\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94\x97\xde!\xe4!\xc3\u007f\xe4\xb8\x02_\x9aQ\xb7\xb3\x90\xb5\xdfx\x04\x8a\x10\xf0\xcf\x06M\u0552\x00\x00\x00\u07d4\x97\xe2\x89s\xb8`\xc5g@(\x00\xfb\xb6<\xe3\x9a\x04\x8a=y\x89\x05B%:\x12l\xe4\x00\x00\u07d4\x97\xe5\xcca'\xc4\xf8\x85\xbe\x02\xf4KB\xd1\u0230\xac\x91\u44c9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x97\xf1\xfeL\x80\x83\xe5\x96!*\x18w(\xdd\\\xf8\n1\xbe\u0149\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\x97\xf7v\x06W\xc1\xe2\x02u\x90\x86\x96>\xb4!\x1c_\x819\xb9\x8a\n\x8a\t\u007f\xcb=\x17h\x00\x00\xe0\x94\x97\xf9\x9bk\xa3\x13F\u0358\xa9\xfeL0\x8f\x87\u0165\x8cQQ\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x98\n\x84\xb6\x86\xfc1\xbd\xc8<\"\x10XTjq\xb1\x1f\x83\x8a\x89*AUH\xaf\x86\x81\x80\x00\u07d4\x98\x10\xe3J\x94\xdbn\xd1V\xd08\x9a\x0e+\x80\xf4\xfdk\n\x8a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x98\x1d\xdf\x04\x04\xe4\xd2-\xdaUj\a&\xf0\v-
\x98\xab\x95i\x8965f3\xeb\xd8\xea\x00\x00\xe0\x94\x98\x1fq'u\xc0\xda\xd9u\x18\xff\xed\xcbG\xb9\xad\x1dl'b\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\x984h!\x80\xb9\x82\xd1f\xba\u06dd\x9d\x1d\x9b\xbf\x01m\x87\xee\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x986\xb4\xd3\x04sd\x1a\xb5j\xee\xe1\x92Bv\x1drrQx\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x989sB\xec_=L\xb8w\xe5N\xf5\xd6\xf1\xd3fs\x1b\u050a\x01@a\xb9\xd7z^\x98\x00\x00\xe0\x94\x98Fd\x886\xa3\a\xa0W\x18O\xd5\x1fb\x8a_\x8c\x12B|\x8a\x04\vi\xbfC\xdc\xe8\xf0\x00\x00\xe0\x94\x98Jy\x85\xe3\xcc~\xb5\xc96\x91\xf6\xf8\xcc{\x8f$]\x01\xb2\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\x98]p\xd2\a\x89+\xed9\x85\x90\x02N$!\xb1\xcc\x11\x93Y\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\x98m\xf4~v\xe4\u05e7\x89\xcd\xee\x91<\u0243\x16P\x93l\x9d\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\x98t\x80?\xe1\xf3\xa06^y\"\xb1Bp\xea\xeb\x03,\xc1\xb5\x89<\xf5\x92\x88$\xc6\xc2\x00\x00\u07d4\x98ub4\x95\xa4l\xdb\xf2YS\x0f\xf88\xa1y\x9e\u00c9\x91\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x98v\x18\xc8VV
|{\xac\x15\a\xc0\xff\xef\xa2\xfbd\xb0\x92\x89\x03}\xfeC1\x89\xe3\x80\x00\u07d4\x98|\x9b\xcdn?9\x90\xa5+\xe3\xed\xa4q\f'Q\x8fOr\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x98\x82\x96|\xeeh\u04a89\xfa\u062bJ|=\xdd\xf6\xc0\xad\u0209Hx\xbe\x1f\xfa\xf9]\x00\x00\u07d4\x98\x85\\}\xfb\xee3SD\x90J\x12\xc4\fs\x17\x95\xb1:T\x899\xfb\xae\x8d\x04-
\xd0\x00\x00\u07d4\x98\x9c\f\xcf\xf6T\xda\x03\xae\xb1\x1a\xf7\x01\x05Ea\xd6)~\x1d\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x98\xa0\xe5Lm\x9d\u023e\x96'l\xeb\xf4\xfe\xc4`\xf6#]\x85\x89j\u0202\x10\tR\u01c0\x00\u07d4\x98\xb7i\xcc0\\\xec\xfbb\x9a\x00\xc9\a\x06\x9d~\xf9\xbc:\x12\x89\x01h\u048e?\x00(\x00\x00\xe0\x94\x98\xbaN\x9c\xa7/\xdd\xc2\fi\xb49ov\xf8\x18?z*N\x8a\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\x00\u07d4\x98\xbeimQ\xe3\x90\xff\x1cP\x1b\x8a\x0fc1\xb6(\xdd\u016d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x98\xbe\u04e7.\xcc\xfb\xaf\xb9#H\x92\x93\xe4)\xe7\x03\xc7\xe2[\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x98\xbfJ\xf3\x81\v\x84#\x87\xdbp\xc1MF\t\x96&\x00=\x10\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x98\xc1\x0e\xbf,O\x97\u02e5\xa1\xab?*\xaf\xe1\xca\xc4#\xf8\u02c9\x10CV\x1a\x88)0\x00\x00\u07d4\x98\xc1\x9d\xba\x81\v\xa6\x11\xe6\x8f/\x83\xee\x16\xf6\xe7tO\f\x1f\x89\n\u05ce\xbcZ\xc6

\x00\x00\xe0\x94\x98\xc5IJ\x03\xac\x91\xa7h\xdf\xfc\x0e\xa1\xdd\u0b3f\x88\x90\x19\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4\x98\xd2\x04\xf9\b_\x8c\x8e}\xe2>X\x9bd\xc6\xef\xf6\x92\xccc\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x98\xd3s\x19\x92\xd1\xd4\x0e\x12\x11\xc7\xf75\xf2\x18\x9a\xfa\a\x02\xe0\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x98\xe2\xb6\xd6\x06\xfd-
i\x91\xc9\xd6\xd4\a\u007f\xdf?\xddE\x85\u06890\xdf\x1ao\x8a\xd6(\x00\x00\u07d4\x98\xe3\xe9\v(
\xfc\xca\ue087y\xb8\xd4\nUh\xc4\x11n!\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\x98\xe6\xf5G\u06c8\xe7_\x1f\x9c\x8a\xc2\xc5\xcf\x16'\xbaX\v>\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x98\xf4\xaf:\xf0\xae\xde_\xaf\xdcB\xa0\x81\xec\xc1\xf8\x9e<\xcf
\x8a\x01\xfd\x934\x94\xaa_\xe0\x00\x00\u07d4\x98\xf6\xb8\xe6!=\xbc\x9aU\x81\xf4\xcc\xe6e_\x95%+\xdb\a\x89\x11Xr\xb0\xbc\xa40\x00\x00\u07d4\x99\te\r\u05719{\x8b\x8b\x0e\xb6\x94\x99\xb2
\x91\xb0\xad\x12\x13\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x99\x11s`\x19G\xc2\bJb\xd69R~\x96\x15\x12W\x9a\xf9\x89
\x86\xac5\x10R`\x00\x00\u07d4\x99\x12\x9d[<\f\xdeG\xea\r\xefM\xfc\a\r\x1fJY\x95'\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x99\x17\u058dJ\xf3A\xd6Q\xe7\xf0\a\\m\xe6\xd7\x14Nt\t\x8a\x012\xd4Gl\b\xe6\xf0\x00\x00\u07d4\x99\x1a\xc7\xcap\x97\x11_&
^\xee\x0e\xf7\xd4\x1e\xb4\xe3\x11\xae\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u0794\x99#e\xd7d\xc5\xce5@9\xdd\xfc\x91.\x02:u\xb8\xe1h\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x99&F\xac\x1a\u02ab\xf5\u076b\xa8\xf9B\x9a\xa6\xa9Nt\x96\xa7\x8967Pz0\xab\xeb\x00\x00\u07d4\x99&\x83'\xc3s3.\x06\xc3\xf6\x16B\x87\xd4U\xb9\xd5\xfaK\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x99(\xffqZ\xfc:+
`\xf8\xebL\u013aN\xe8\u06b6\u5749\x17\xda:\x04\u01f3\xe0\x00\x00\u07d4\x992\xef\x1c\x85\xb7Z\x9b*\x80\x05}P\x874\xc5\x10\x85\xbe\u0309\x02\xb8?\xa50\x1dY\x00\x00\xe0\x94\x99?\x14ax`
^f\xd5\x17\xbex.\xf0\xb3\xc6\x1aN\x19%\x8a\x01|\x1f\x055\u05e5\x83\x00\x00\xe0\x94\x99A7\x04\xb1\xa3.p\xf3\xbc\ri\u0748\x1c8VkT\u02ca\x05\xcckiF1\xf7\x12\x00\x00\u07d4\x99AR\xfc\x95\xd5\xc1\u028b\x88\x11:\xbb\xadMq\x0e@\xde\xf6\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x99D\xfe\xe9\xd3JJ\x88\x00#\u01c92\xc0\vY\xd5\xc8*\x82\x89(\xa8\xa5k6\x90\a\x00\x00\u07d4\x99L\u00b5\"~\xc3\xcf\x04\x85\x12F|A\xb7\xb7\xb7H\x90\x9f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x99q\xdf`\xf0\xaef\xdc\xe9\xe8\xc8N\x17\x14\x9f\t\xf9\xc5/d\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x99v\x94~\xff_j\xe5\xda\b\xddT\x11\x92\xf3x\xb4(\xff\x94\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x99}e\x92\xa3\x15\x89\xac\xc3\x1b\x99\x01\xfb\xeb<\xc3\xd6[2\x15\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x99\x82\xa5\x89\x0f\xfbT\x06\u04ec\xa8\u04bf\xc1\xddp\xaa\xa8\n\xe0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x99\x87\x8f\x9dn\n~\u066e\u01c2\x97\xb78y\xa8\x01\x95\xaf\xe0\x89\xd7\xc1\x98q\x0ef\xb0\x00\x00\u07d4\x99\x8c\x1f\x93\xbc\xdbo\xf2<\x10\xd0\u0712G(\xb7;\xe2\xff\x9f\x896[\xf3\xa43\xea\xf3\x00\x00\u07d4\x99\x91aL[\xaaG\xddl\x96\x87FE\xf9z\xdd,=\x83\x80\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x99\x92J\x98\x16\xbb}\xdf?\xec\x18D\x82\x8e\x9a\xd7\xd0k\xf4\xe6\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\x99\x99vh\xf7\xc1\xa4\xff\x9e1\xf9\x97z\xe3\"K\u02c8z\x85\x89\x0f\xc969(\x01\xc0\x00\x00\u07d4\x99\x9cI\xc1t\xca\x13\xbc\x83l\x1e\n\x92\xbf\xf4\x8b'\x15C\u0289\xb1\xcf$\xdd\u0431@\x00\x00\u07d4\x99\xa4\xde\x19\xde\u05d0\b\xcf\xdc\xd4]\x01M.XK\x89\x14\xa8\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4\x99\xa9k\xf2$.\xa1\xb3\x9e\xceo\xcc\r\x18\xae\xd0\f\x01y\xf3\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x99\xb0\x18\x93+\xca\xd3U\xb6y+%]\xb6p-
\xec\x8c\xe5\u0749\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4\x99\xb7C\xd1\xd9\xef\xf9\r\x9a\x194\xb4\xdb!\xd5\x19\u061bJ8\x89\x05k\xc7^-

c\x10\x00\x00\u07d4\x99\xb8\xc8$\x86\x9d\xe9\xed$\xf3\xbf\xf6\x85L\xb6\xddE\xcc?\x9f\x89e\xea
=\xb7UF`\x00\x00\u07d4\x99\xc0\x17L\xf8N\a\x83\xc2
\xb4\xebj\xe1\x8f\xe7\x03\x85J\u04c9py\xa2W=\fx\x00\x00\u07d4\x99\xc1\xd9\xf4\fj\xb7\xf8\xa9/\x
ce/\xdc\xe4zT\xa5\x86\xc5?\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\x99\xc26\x14\x1d\xae\xc87\xe
c\xe0O\xda\xee\x1d\x90\u03cb\xbd\xc1\x04\x89ve\x16\xac\xac\r
\x00\x00\u07d4\x99\xc3\x1f\xe7HX7\x87\xcd\xd3\xe5%\xb2\x81\xb2\x18\x96\x179\xe3\x897\b\xba
\xed=h\x90\x00\x00\u07d4\x99\xc4u\xbf\x02\xe8\xb9!J\xda_\xad\x02\xfd\xfd\x15\xba6\\\f\x89
\t\xc5\u023fo\xdc\x00\x00\u07d4\x99\u0203%\x85F\xcc~N\x97\x1fR.8\x99\x18\xda^\xa6:\x89\xd8\
xd7&\xb7\x17z\x80\x00\x00\u07d4\x99\xc9\xf9>E\xfe<\x14\x18\xc3S\xe4\u016c8\x94\xee\xf8\x12\
x1e\x89\x05\x85\xba\xf1E\x05\v\x00\x00\xe0\x94\x99\xd1W\x9c\xd4&\x82\xb7dN\x1dOq(D\x1e\xe
f\xfe3\x9d\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\x99\u0475\x85\x96_@jB\xa4\x9a\x1c\xa7\x0
fv\x9evZ?\x98\x8a\x03\x89O\x0eo\x9b\x9fp\x00\x00\u07d4\x99\xdf\xd0PL\x06\xc7C\xe4e4\xfd{U\x
f1\xf9\xc7\xec3)\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x99\xf4\x14|\xcck\u02c0\u0304.i\xf6\xd0\x0e
0\xfaA3\u0649\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x99\xf7\u007f\x99\x8b
\xe0\xbc\xdc\xd9\xfc\x83\x86ARl\xf2Y\x18\xef\x89a\t=|,m8\x00\x00\u07d4\x99\xfa\xd5\x008\xd0\x
d9\xd4\xc3\xfb\xb4\xbc\xe0V\x06\xec\xad\xcdQ!\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x99\xfe\r
\x12(\xa7S\x14VU\xd4(\xeb\x9f\xd9I\x85\xd3m\x89i
\xbf\xf3QZ:\x00\x00\u07d4\x9a\a\x9c\x92\xa6)\xca\x15\xc8\xca\xfa.\xb2\x8d[\xc1z\xf8(\x11\x89\x1
b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x9a\r<\xee=\x98\x92\xea;7\x00\xa2\u007f\xf8A@\xd9\x02T
\x93\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\x9a$\u038dH\\\xc4\xc8nI\u07b3\x90\"\xf9,t0\xe6~\x8
9Fy\x1f\xc8N\a\xd0\x00\x00\u07d4\x9a,\xe4;]\x89\u0593k\x8e\x8c5G\x91\xb8\xaf\xff\x96$%\x89lk\
x93[\x8b\xbd@\x00\x00\u07d4\x9a9\x01bS^9\x88w\xe4\x16x}b9\xe0uN\x93|\x8965\u026d\xc5\u0
7a0\x00\x00\u07d4\x9a=\xa6P#\xa10
\xd2!E\xcf\xc1\x8b\xab\x10\xbd\x19\xceN\x89\x18\xbfn\xa3FJ:\x00\x00\xe0\x94\x9a>+\x1b\xf3F\xd
d\a\v\x02sW\xfe\xacD\xa4\xb2\xc9}\xb8\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9aL\xa
8\xb8!\x17\x89NC\xdbr\xb9\xfax\xf0\xb9\xb9:\xce\t\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d
4\x9aR.R\xc1\x95\xbf\xb7\xcf_\xfa\xae\u06d1\xa3\xbath\x16\x1d\x8965\u026d\xc5\u07a0\x00\x00\
u07d4\x9aZ\xf3\x1c~\x063\x9a\u0234b\x8d|M\xb0\xce\x0fE\u0224\x89\x1b\x1a\xe4\xd6\xe2\xefP\
x00\x00\u0794\x9ac?\xcd\x11,\xce\xebv_\xe0A\x81ps*\x97\x05\u7708\xfc\x93c\x92\x80\x1c\x00\x
00\u07d4\x9ac\u0445\xa7\x91)\xfd\xab\x19\xb5\x8b\xb61\xea6\xa4
TN\x89\x02F\xdd\xf9yvh\x00\x00\u07d4\x9ag\b\u0778\x90<(\x9f\x83\xfe\x88\x9c\x1e\xdc\xd6\x1f\
x85D#\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9ao\xf5\xf6\xa7\xaf{z\xe0\xed\x9c
\xec\xecP#\u0481\xb7\x86\x89\x8a\x12\xb9\xbdjg\xec\x00\x00\xe0\x94\x9a\x82\x82m<)H\x1d\xcc
+\u0495\x00G\xe8\xb6\x04\x86\xc38\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x9a\x8e\xcaA\x89\
xffJ\xa8\xff~\u0536\xb7\x03\x9f\t\x02!\x9b\x15\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x9a\x95
;[\xccp\x93y\xfc\xb5Y\u05f9\x16\xaf\u06a5\f\xad\u0309\x05k\xc7^-
c\x10\x00\x00\u07d4\x9a\x99\v\x8a\xebX\x8d~\xe7\xec.\xd8\xc2\xe6Os\x82\xa9\xfe\xe2\x89\x01\x
d1'\xdbi\xfd\x8b\x00\x00\u07d4\x9a\x9d\x1d\xc0\xba\xa7}n
\xc3\xd8I\u01c8b\xdd\x1c\x05L\x87\x89/\xb4t\t\x8fg\xc0\x00\x00\xe0\x94\x9a\xa4\x8cf\xe4\xfbJ\u0
419\x93N2\x02.\x82t'\xf2w\xba\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9a\xa80\x8fB\x
91\x0eZ\xde\t\xc1\xa5\xe2\x82\xd6\xd9\x17\x10\xbd\xbf\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x9a\xaa\xfa\x00gd~\u0659\x06kzL\xa5\xb4\xb3\xf3\xfe\xaao\x8965\u026d\xc5\u07

a0\x00\x00\u07d4\x9a\xb9\x88\xb5\x05\xcf\xee\x1d\xbe\x9c\u044e\x9bTs\xb9\xa2\xd4\xf56\x89\x11X\xe4`\x91=\x00\x00\x00\u07d4\x9a\xb9\x8dm\xbb\x1e\xaa\xe1mE\xa0EhT\x1a\xd3\xd8\xfe\x06\u0309\x0e\xc5\x04d\xfe#\xf3\x80\x00\xe0\x94\x9a\xba+^'\xffx\xba\xaa\xb5\xcd\u0248\xb7\xbe\x85\\\xeb\xbd\u038a\x02\x1e\f\x00\x13\a\n\xdc\x00\x00\u07d4\x9a\xc4\xdaQ\xd2x\"\xd1\xe2\b\xc9n\xa6J\x1e[U)\x97#\x89\x05lUy\xf7\"\x14\x00\x00\u0794\x9a\xc8S\x97y*i\u05cf(k\x86C*\a\xae\u03b6\x0ed\x88\xc6s\xce<@\x16\x00\x00\xe0\x94\x9a\xc9\a\xee\x85\xe6\xf3\xe2#E\x99\x92\xe2V\xa4?\xa0\x8f\xa8\xb2\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9a\xd4\u007f\xdc\xf9\u0354-(\xef\xfd[\x84\x11[1\xa6X\xa1>\x89\xb2Y\xec\x00\xd5;(\x00\x00\u07d4\x9a\xdb\u04fc{\n\xfc\x05\xd1\xd2\xed\xa4\x9f\xf8c\x93\x9cH\xdbF\x89\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4\x9a\xdfE\x8b\xff5\x99\xee\xe1\xa2c\x98\x85<W[\u00ccc\x13\x89\x0f-\xc7\xd4\u007f\x15`\x00\x00\u07d4\x9a\xe1;\u0602\xf2Weu\x92\x1a\x94\x97L\xbe\xa8a\xba\r5\x89\xabM\xcf9\x9a:`\x00\x00\u07d4\x9a\xe9Gk\xfe\xcd5\x91\x96M\xd3%\u03cc*$\xfa\xed\x82\xc1\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9a\xf1\x00\xcc=\xae\x83\xa34\x02\x05\x1c\xe4Ik\x16aT\x83\xf6\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x9a\xf1\x13\x99Q\x1c!1\x81\xbf\xda:\x8b&L\x05\xfc\x81\xb3\u038a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u0794\x9a\xf5\u0249L3\xe4,,Q\x8e:\xc6p\xea\x95\x05\u0475>\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\x9a\xf9\xdb\xe4t\"\xd1w\xf9E\xbd\xea\xd7\xe6\xd8)05b0\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x9a\xfaSkLf\xbc8\xd8u\u0133\x00\x99\xd9&\x1f\xdb8\xeb\x89\v*\x8f\x84*w\xbc\x80\x00\u07d4\x9b\x06\xad\x84\x1d\xff\xbeL\xcfF\xf1\x03\x9f\u00c6\xf3\xc3!Dn\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9b\x11h\u078a\xb6KGU/3\x89\x80\n\x9c\xc0\x8bFf\u03c9]\u0212\xaa\x111\xc8\x00\x00\u07d4\x9b\x18\x11\xc3\x05\x1fF\xe6d\xaeK\xc9\xc8$\u0445\x92\xc4WJ\x89\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4\x9b\x18G\x86U\xa4\x85\x1c\xc9\x06\xe6`\xfe\xaca\xf7\xf4\u023f\xfc\x89\xe2G\x8d8\x90}\x84\x00\x00\u07d4\x9b\"\xa8\r\\{3t\xa0[D`\x81\xf9}\n4\a\x9e\u007f\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x9b+\xe7\xf5gT\xf5\x05\xe3D\x1a\x10\xf7\xf0\xe2\x0f\xd3\xdd\xf8I\x89\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\x9b2\xcfOQ\x15\xf4\xb3J\x00\xa6La}\xe0c\x875C#\x89\x05\xb8\x1e\u0608|\x80\x00\u07d4\x9bC\u0739_\xde1\x80u\xa5g\xf1\xe6\xb5v\x17\x05^\xf9\xe8\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\x9bDO\xd37\xe5\xd7R\x93\xad\xcf\xffp\xe1\xea\x01\xdb\x022\"\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x9bH$\xff\x9f\xb2\xab\xdaUM\xeeO\xb8\xcfT\x91eW\x061\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x9bL'\x15x\f\xa4\xe9\x9e`\xeb\xf2\x19\xf1Y\f\x8c\xadP\n\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4\x9bY\xeb!;\x1eue\xe4PG\xe0N\xa07O\x10v-\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9b\\9\xf7\xe0\xac\x16\x8c\x8e\xd0\xed4\x04w\x11}\x1bh.\xe9\x89\x05P\x05\xf0\xc6\x14H\x00\x00\u07d4\x9b^\xc1\x8e\x83\x13\x88}\xf4a\u0490.\x81\xe6z\x8f\x11;\xb1\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x9bd\xd3\u034d+s\xf6hA\xb5\xc4k\xb6\x95\xb8\x8a\x9a\xb7]\x89\x01:Ov\f\x16\x80\x00\u07d4\x9be\x8f\xb3a\xe0F\xd4\xfc\xaa\x8a\xefm\x02\xa9\x91\x11\"6%\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9bfA\xb1>\x17/\xc0r\xcaK\x83'\xa3\xbc(\xa1[f\xa9\x89\x06\x81U\xa46v\xe0\x00\x00\xe0\x94\x9bh\xf6t\x16\xa6;\xf4E\x1a1\x16L\x92\xf6r\xa6\x87Y\xe9\x8a\f\xb4\x9bD\xba`-\x80\x00\x00\u07d4\x9bw6i\xe8}v\x01\x8c\t\x0f\x82U\xe5D\t\xb9\u0728\xb2\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x9bw\xeb\xce\xd7\xe2\x15\xf0\x92\x0e\x8c+\x87\x00$\xf6\xec\xb2\xff1\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9b|\x88\x10\xcc|\u021e\x80Nm>8\x12\x18PG(w\xfe\x89lk\x93

[\x8b\xbd@\x00\x00\u07d4\x9b\xa5=\xc8\xc9^\x9aG/\xeb\xa2\xc4\xe3,\x1d\xc4\xdd{\xabF\x89Hz\x9a0E9D\x00\x00\xe0\x94\x9b\xac\xd3\xd4\x0f;\x82\xac\x91\xa2d\xd9\u060d\x90\x8e\xac\x86d\xb9\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x9b\xb7`\xd5\u0089\xa3\xe1\xdb\x18\xdb\tSE\xcaA;\x9aC\u0089\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\x9b\xb7b\x04\x18j\xf2\xf6;\xe7\x91h`\x16\x87\xfc\x9b\xadf\x1f\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x9b\xb9\xb0*&\xbf\xe1\xcc\xc3\xf0\xc6!\x9e&\x1c9\u007f\xc5\xcax\x89Hz\x9a0E9D\x00\x00\u07d4\x9b\xc5s\xbc\xda#\xb8\xb2o\x90s\xd9\f#\x0e\x8eq\xe0'\v\x896/u\xa40]\f\x00\x00\u07d4\x9b\xd7\u00caB\x100JMe>\xde\xff\x1b<\xe4_\xcexC\x89\x0fI\x89A\xe6d(\x00\x00\xe0\x94\x9b\u0600h\xe10u\xf3\xa8\xca\xc4d\xa5\xf9I\xd6\xd8\x18\xc0\xf6\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x9b\xd9\x05\xf1q\x9f\u01ec\xd0\x15\x9dM\xc1\xf8\xdb/!G#8\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9b\xdb\u071b\x9741\xd1<\x89\xa3\xf9u~\x9b;bu\xbf\u01c9\x1b\x1a}\u03caD\u04c0\x00\u07d4\x9b\xe3\xc3)\xb6*(\xb8\xb0\x88l\xbd\x8b\x99\xf8\xbc\x93\f\xe3\xe6\x89\x04\t\xe5+H6\x9a\x00\x00\xe0\x94\x9b\xf5\x8e\xfb\xea\a\x84\xeb\x06\x8a\xde\u03e0\xbb!P\x84\xc7:5\x8a\x01:k+VHq\xa0\x00\x00\u07d4\x9b\xf6r\xd9y\xb3fR\xfcR\x82Tzjk\xc2\x12\xaeCh\x89#\x8f\xd4,\\\xf0@\x00\x00\xe0\x94\x9b\xf7\x03\xb4\x1c6$\xe1_@T\x96#\x90\xbc\xba0R\xf0\xfd\x8a\x01H>\x01S<.<\x00\x00\u07d4\x9b\xf7\x1f\u007f\xb57\xacT\xf4\xe5\x14\x94\u007f\xa7\xffg(\xf1m/\x89\x01\u03c4\xa3\n\n\f\x00\x00\u07d4\x9b\xf9\xb3\xb2\xf2<\xf4a\xebY\x1f(4\v\xc7\x19\x93\x1c\x83d\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x9b\xfce\x9c\x9c`\x1e\xa4*k!\xb8\xf1p\x84\xec\x87\xd7\x02\x12\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9b\xff\xf5\r\xb3jxUU\xf0vR\xa1S\xb0\xc4+\x1b\x8bv\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9c\x05\xe9\xd0\xf0u\x8eyS\x03q~1\xda!<\xa1W\u618965\u026d\xc5\u07a0\x00\x00\u07d4\x9c\x1bw\x1f\t\xaf\x88*\xf0d0\x83\xde*\xa7\x9d\xc0\x97\xc4\x0e\x89\x86p\xe9\xece\x98\xc0\x00\x00\u07d4\x9c(\xa2\xc4\b`\x91\xcb]\xa2&\xa6W\xce2H\xe8\xea{o\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\u07d4\x9c/\xd5@\x89\xaff]\xf5\x97\x1ds\xb8\x04a`9dsu\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9c4@\x98\xbaaZ9\x8f\x11\xd0\t\x90[\x17|D\xa7\xb6\x02\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9c=\x06\x92\xce\xee\xf8\n\xa4\x96\\\xee\xd2b\xff\xc7\xf0i\xf2\u0709\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x9c@\\\xf6\x97\x95a8\x06^\x11\xc5\xf7U\x9eg$[\u0465\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x9cE
*%\xf6\xad\x00\x11\xf1\x15\xa5\xa7\"\x04\xf2\xf2\x19\x88f\x8a\x01\x0f\xcf:b\xb0\x80\x98\x00\x00\xe0\x94\x9cI\xde\xffG\b_\xc0\x97\x04\u02a2\u0728\u0087\xa9\xa17\u068a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\x9cK\xbc\xd5\xf1dJo\aX$\xdd\xfe\x85\xc5q\u05ab\xf6\x9c\x89a\x94\x04\x9f0\xf7\x00\x00\u07d4\x9cRj\x14\x06\x83\xed\xf1C\x1c\xfa\xa1(\xa95\xe2\xb6\x14\u060b\x89\x06\x04o7\xe5\x94\\\x00\x00\xe0\x94\x9cT\xe4\xedG\x9a\x85h)\u01bbB\u069f\vi*u\xf7(\x8a\x01\x97\xa8\xf6\xddU\x19\x80\x00\x00\xe0\x94\x9cX\x1a`\xb6\x10(\xd94\x16y)\xb2-
p\xb3\x13\xc3O\u040a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\x9c\\\xc1\x11\t,\x12!\x16\xf1\xa8_N\xe3\x14\bt\x1a}/\x89\x1a\xb2\xcf|\x9f\x87\xe2\x00\x00\u07d4\x9ck\u0264k\x03\xaeT\x04\xf0C\xdf\xcf!\x88>A\x10\xcc3\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x9cx\x96?\xbc&<\t\xbdr\xe4\xf8\xde\xf7J\x94u\xf7\x05\\\x8a\x02\ub3b1\xa1r\u0738\x00\x00\u07d4\x9cx\xfb\xb4\xdfv\x9c\xe2\xc1V\x92\f\xfe\xdf\xda\x03:\x0e%J\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x9c{m\xc5\x19\x0f\xe2\x91)c\xfc\xd5yh>\xc79Q\x16\xb0\x89*\x11)\u0413g
\x00\x00\u07d4\x9c\x80\xbc\x18\xe9\xf8\u0516\x8b\x18]\xa8\u01df\xa6\xe1\x1f\xfc>#\x89\r\x02\xabHl\xed\xc0\x00\x00\xe0\x94\x9c\x98\xfd\xf1\xfd\u034b\xa8\xf4\u0170L:\xe8X~\xfd\xf0\xf6\xe6\x8a

\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\x9c\x99\xa1\u0691\u0552\v\xc1N\f\xb9\x14\xfd\xf6+\x94\u02c3X\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\x9c\x99\xb6&\x06(\x1b\\\xef\xab\xf3aV\xc8\xfeb\x83\x9e\xf5\xf3\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x9c\x9a\a\xa8\xe5|1r\xa9\x19\xefdx\x94tI\x0f\r\x9fQ\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9c\x9d\xe4G$\xa4\x05M\xa0\xea\xa6\x05\xab\u0300&hw\x8b\xea\x89\n\xd7\xd5\xca?\xa5\xa2\x00\x00\u07d4\x9c\x9f;\x8a\x81\x1b!\xf3\xff?\xe2\x0f\xe9p\x05\x1c\xe6j\x82O\x89>\xc2\u07bc\a\u053e\x00\x00\xe0\x94\x9c\x9f\x89\xa3\x91\x0fj*\xe8\xa9\x10G\xa1z\xb7\x88\xbd\xde\xc1p\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9c\xa0B\x9f\x87O\x8d\xce\xe2\xe9\xc0b\xa9\x02\n\x84*Xz\xb9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9c\xa4.\u7838\x98\xf6\xa5\xcc`\xb5\xa5\u05f1\xbf\xa3\xc321\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9c\xb2\x8a\xc1\xa2\n\x10o\u007f76\x92\xc5\xceLs\xf172\xa1\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9c\xcd\u0732\xcf\u00b2[\br\x9a\n\x98\xd9\xe6\xf0 .\xa2\xc1\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x9c\xe2\u007f$^\x02\xd1\xc3\x12\xc1\xd5\x00x\x8c\x9d\xefv\x90E;\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x9c\xe56;\x13\xe8#\x8a\xa4\xdd\x15\xac\u0432\xe8\xaf\xe0\x872G\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\x9c\xf2\x92\x8b\xee\xf0\x9a@\xf9\xbf\xc9S\xbe\x06\xa2Q\x11a\x82\xfb\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x9d\x06\x91\x97\xd1\xdeP\x04Z\x18o^\xc7D\xac@\u8bd1\u0189lk\x93[\x8b\xbd@\x00\x00\u07d4\x9d\x0e}\x92\xfb0XS\u05d8&;\xf1^\x97\xc7+\xf9\xd7\xe0\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9d\x0f4~\x82k}\u03aa\xd2y\x06\n5\xc0\x06\x1e\xcf3K\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9d u\x17B,\xc0\xd6\r\xe7\xc27\tzMO\xce
\x94\f\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\x9d%\n\xe4\xf1\x10\xd7\x1c\xaf\u01f0\xad\xb5.\x8d\x9a\xcbfy\xb8\x8a\x02\x15mn\x99r\x13\xc0\x00\x00\xe0\x94\x9d+\xfc6\x10o\x03\x82P\xc0\x18\x01hW\x85\xb1l\x86\xc6\r\x8aPw\xd7]\xf1\xb6u\x80\x00\x00\xe0\x94\x9d0\xcb#{\xc0\x96\xf1p6\xfc\x80\xdd!\xcah\x99,\xa2\u064a\x06n\xe71\x8f\u070f0\x00\x00\u07d4\x9d2\x96.\xa9\x97\x00\xd92(\xe9\xdb\xda\xd2\xcc7\xbb\x99\xf0~\x89\xb4c+\xed\xd4\xde\xd4\x00\x00\u07d4\x9d4\xda\xc2[\xd1X(\xfa\xef\xaa\xf2\x8fq\aS\xb3\x9e\x89\u0709;\x1cV\xfe\xd0-
\xf0\x00\x00\u07d4\x9d6\x91e\xfbp\xb8\x1a:v_\x18\x8f\xd6\f\xbe^{\th\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9d@\xe0\x12\xf6\x04%\xa3@\xd8-
\x03\xa1\xc7W\xbf\xab\xc7\x06\xfb\x89\t4o:\xdd\u020d\x80\x00\u07d4\x9dAt\xaaj\xf2\x84v\xe2)\xda\xdbF\x18\b\b\xc6u\x05\xc1\x89B\x1a\xfd\xa4.\u0597\x00\x00\u07d4\x9dB\x133\x9a\x01U\x18avL\x87\xa9<\xe8\xf8_\x87\x95\x9a\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x9dF\f\x1b7\x9d\xdb\x19\xa8\xc8[LgG\x05\r\xdf\x17\xa8u\x89\xb5\x0f\u03ef\xeb\xec\xb0\x00\x00\u07d4\x9dG\xba[L\x85\x05\xad\x8d\xa4)4(\va\xa0\xe1\xe8\xb9q\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x9dM2\x11w%n\xbd\x9a\xfb\xda0A5\xd5\x17\xc3\xdcV\x93\x89!d\xb7\xa0J\u0220\x00\x00\u07d4\x9dO\xf9\x89\xb7\xbe\u066b\x10\x9d\x10\xc8\xc7\xe5_\x02\xd7g4\xad\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\x9dQ\x15C\xb3\xd9\xdc`\xd4\u007f\t\u051d\x01\xb6\u0118\xd8
x\x8a\x02a\x97\xb9Qo\u00d4\x00\x00\u07d4\x9dn\u03e0:\xf2\xc6\xe1D\xb7\xc4i*\x86\x95\x1e\x90.\x9e\x1f\x89\xa2\xa5\xaa`\xad$?\x00\x00\u07d4\x9dvU\xe9\xf3\xe5\xba]n\x87\xe4\x12\xae\xbe\x9e\xe0\u0512G\ue24e\t1\x1c\x1d\x80\xfa\x00\x00\u07d4\x9dx1\xe84\xc2\v\x1b\xaaiz\xf1\xd8\xe0\xc6!\u016f\xff\x9a\x89\x04\xb0m\xbb\xb4\x0fJ\x00\x00\u07d4\x9dx\xa9u\xb7\xdb^M\x8e(\x84\\\xfb\xe7\xe3\x14\x01\xbe\r\u0649H\xa4<T`/p\x00\x00\u07d4\x9dy\x9e\x94>0k\xa2\u5e5c\x8ahX\u02f5,\f

\xf75\x89\x10CV\x1a\x88)0\x00\x00\xe0\x94\x9d\u007f\xdapp\xbf>\xe9\xbb\u0664\x1fU\xca\u0505J\xe6\xc2,\x8a\x02U\u02e3\xc4o\xcf\x12\x00\x00\u07d4\x9d\x81\xae\xa6\x9a\xedj\xd0p\x89\xd6\x14E4\x8c\x17\xf3K\xfc[\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x9d\x91\x1f6\x82\xf3/\xe0y.\x9f\xb6\xff<\xfcG\xf5\x89\xfc\xa5\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9d\x91;]3\x9c\x95\xd8wEV%c\xfe\xa9\x8b#\xc6\f\u0109\tA0,\u007fM#\x00\x00\u07d4\x9d\x93\xfa\xb6\xe2(E\xf8\xf4Z\aIo\x11\xdeqS\r\xeb\u01c9lO\xd1\xee$nx\x00\x00\u07d4\x9d\x99\xb1\x89\xbb\u0664\x8f\xc2\xe1n\x8f\u03633;\xb9\x9a1{\xbb\x89=\x16\xe1\vm\x8b\xb2\x00\x00\u07d4\x9d\x9cN\xfe\x9fC9\x89\xe2;\xe9@I!S)\xfaU\xb4\u02c9\r\u3c89\x03\u01b5\x80\x00\u07d4\x9d\x9eW\xfd\xe3\x0ePh\xc0>I\x84\x8e\xdc\xe3C\xb7\x02\x83X\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4\x9d\xa30\"@\xaf\x05\x11\xc6\xfd\x18W\xe6\u07779Ow\xabk\x89\xa8\r$g~\xfe\xf0\x00\x00\u07d4\x9d\xa4\xec@pw\xf4\xb9p{-\x9d.\xde^\xa5(+\xf1\u07c9\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9d\xa6\t\xfa:~l\xf2\xcc\x0ep\u036b\xe7\x8d\xc4\xe3\x82\xe1\x1e\x89A\rXj
\xa4\xc0\x00\x00\xe0\x94\x9d\xa6\x1c\xcdb\xbf\x86\x06V\xe02]qW\xe2\xf1`\xd9;\xb5\x8a\x01\x0f\f\xa9V\xf8y\x9e\x00\x00\xe0\x94\x9d\xa6\xe0u\x98\x9ct\x19\tL\xc9\xf6\xd2\u44d3\xbb\x19\x96\x88\x8a\x02Y\xbbq\u056d\xf3\xf0\x00\x00\u07d4\x9d\xa8\xe2,\xa1\x0eg\xfe\xa4NR^GQ\xee\xac6\xa3\x11\x94\x89\x0e\x189\x8ev\x01\x90\x00\x00\u07d4\x9d\xb2\xe1\\\xa6\x81\xf4\xc6`H\xf6\xf9\xb7\x94\x1e\u040b\x1f\xf5\x06\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9d\xc1\x0f\xa3\x8f\x9f\xb0h\x10\xe1\x1f`\x17>\xc3\xd2\xfdju\x1e\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\x9d\xd2\x19f$\xa1\xdd\xf1J\x9d7^_\a\x15+\xaf\"\xaf\xa2\x89A\xb0^$c\xa5C\x80\x00\u07d4\x9d\xd4k\x1cm?\x05\u279co\x03~\xed\x9aYZ\xf4\xa9\xaa\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x9d\xdd5^cN\xe9\x92~K\u007fl\x97\xe7\xbf:/\x1ehz\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\x9d\xe2\n\xe7j\xa0\x82c\xb2\x05\xd5\x14$a\x96\x1e$\b\xd2f\x89\r\xa93\xd8\xd8\xc6p\x00\x00\u07d4\x9d\xe2\v\xc3~\u007fH\xa8\x0f\xfdz\xd8O\xfb\xf1\xa1\xab\xe1s\x8c\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x9d\xe78m\xde@\x1c\xe4\xc6{q\xb6U?\x8a\xa3N\xa5\xa1}\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\x9d\xeb9\x02z\xf8w\x99+\x89\xf2\xecJ\x1f\x82.\xcd\xf1&\x93\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x9d\xef\xe5j\x0f\xf1\xa1\x94}\xba\t#\xf7\xdd%\x8d\x8f\x12\xfaE\x8a\x05\xb1*\ufbe8\x04\x00\x00\x00\u07d4\x9d\xf0W\xcd\x03\xa4\xe2~\x8e\x03/\x85y\x85\xfd\u007f\x01\xad\xc8\u05c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x9d\xf3*P\x1c\vx\x1c\x02\x81\x02/B\xa1)?\xfd{\x89*\x8a\x01\xe7\xe4\x17\x1b\xf4\u04e0\x00\x00\u07d4\x9e\x01vZ\xff\b\xbc\"\x05P\xac\xa5\xea.\x1c\xe8\u5c19#\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9e
\xe5\xfd6\x1e\xab\xcfc\x89\x1f[\x87\xb0\x92h\xb8\xeb7\x93\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x9e#,\b\xc1M\xc1\xa6\xed\v\x8a;(h\x97{\xa5\xc1}\x10\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x9e#\xc5\u4dc2\xb0\n_\xad\U0006eb47\xda\xcf[\x03g\xa1\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x9e59\x90q\xa4\xa1\x01\xe9\x19M\xaa?\t\xf0J\v_\x98p\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9e>\xb5\t'\x8f\xe0\xdc\xd8\xe0\xbb\xe7\x8a\x19N\x06\xb6\x809C\x892\xf5\x1e\u06ea\xa30\x00\x00\u07d4\x9eBrrQk>g\xd4\xfc\xbf\x82\xf5\x93\x90\xd0L\x8e(\xe5\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\x9eL\xec5:\xc3\u3043^<\t\x91\xf8\xfa\xa5\xb7\u0428\xe6\x8a\x02\x1e\x18\xb9\xe9\xabE\xe4\x80\x00\u07d4\x9eX\x11\xb4\v\xe1\xe2\xa1\xe1\u048c;\at\xac\xde\n\t`=\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\x9eZ1\x1d\x9fi\x89\x8a|j\x9dc`h\x048\xe6z{/\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4\x9e|
P\xa2'\xbb\xfd`\x93~&\x8c\xea>h\xfe\xa8\xd1\xfe\x89\x05k\xc7^-c\x10\x00\x00\u07d4\x9e\u007fe\xa9\x0e\x85\b\x86{\xcc\xc9\x14%j\x1e\xa5t\xcf\a\xe3\x89C8t\xf6

2\xcc`\x00\x00\xe0\x94\x9e\x81D\xe0\x8e\x89dx\x11\xfekr\xd4E\u05a5\xf8\n\xd2D\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9e\x8fd\xdd\xcd\u9e34Q\xba\xfa\xa25\xa9\xbfQ\x1a%\xac\x91\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\x9e\x95\x1fm\xc5\xe3R\xaf\xb8\xd0B\x99\xd2G\x8aE\x12Y\xbfV\x89\x03\xe7A\x98\x81\xa7:\x00\x00\u07d4\x9e\x96\r\xcd\x03\u057a\x99\xcb\x11]\x17\xffL\t$\x8a\xd4\u043e\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x9e\xafj2\x8a@v\x02N\xfakg\xb4\x8b!\xee\xdc\xc0\xf0\xb8\x89\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u07d4\x9e\xb1\xffqy\x8f(\xd6\xe9\x89\xfa\x1e\xa0X\x8e'\xba\x86\xcb}\x89\a\xa1\xfe\x16\x02w\x00\x00\x00\u07d4\x9e\xb2\x81\xc3'\x19\xc4\x0f\xdb>!m\xb0\xf3\u007f\xbcs\xa0&\xb7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x9e\xb3\xa7\xcb^g&Bz:6\x1c\xfa\x8dad\xdb\u043a\x16\x89+\x95\xbd\xcc9\xb6\x10\x00\x00\u07d4\x9e\xb7\x83N\x17\x1dA\xe0i\xa7yG\xfc\xa8v\"\xf0\xbaNH\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\x9e\xc0>\x02\u51f7v\x9d\xefS\x84\x13\xe9\u007f~U\xbeq\u060a\x04+\xf0kx\xed;P\x00\x00\u07d4\x9e\u02eb\xb0\xb2'\x82\xb3uD)\xe1uz\xab\xa0K\x81\x18\x9f\x89,\xa7\xbb\x06\x1f^\x99\x80\x00\u07d4\x9e\xce\x14\x00\x80\t6\xc7\xc6H_\xcd\xd3b`\x17\u041a\xfb\xf6\x89\x10\xce\x1d=\x8c\xb3\x18\x00\x00\u07d4\x9e\xd4\xe6?ReB\xd4O\xdd\xd3MY\xcd%8\x8f\xfdk\u0689\u049b4\xa4cH\x94\x00\x00\u07d4\x9e\xd8\x0e\xda\u007fU\x05M\xb9\xfbR\x82E\x16\x88\xf2k\xb3t\xc1\x89\x10CV\x1a\x88)0\x00\x00\u07d4\x9e\u0710\xf4\xbe!\be!J\xb5\xb3^Z\x8d\xd7t\x15'\x9d\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9e\u07acL\x02k\x93\x05M\u0171\xd6a\fo9`\xf2\xads\x89A\rXj
\xa4\xc0\x00\x00\u07d4\x9e\xe9?3\x9eg&\xece\xee\xa4O\x8aK\xfe\x10\xda=2\x82\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\x9e\xe9v\f\xc2s\xd4pj\xa0\x83u\xc3\xe4o\xa20\xaf\xf3\u054a\x01\xe5.3l\xde\"\x18\x00\x00\u07d4\x9e\xeb\a\xbd+x\x90\x19^}F\xbd\xf2\a\x1bf\x17QM\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4\x9e\xefD-
)\x1aD}t\xc5\xd2S\u011e\xf3$\xea\xc1\xd8\xf0\x89\xb9f\b\xc8\x10;\xf0\x00\x00\u07d4\x9e\xf1\x89k\x00|2\xa1Q\x14\xfb\x89\xd7=\xbdG\xf9\x12+i\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9f\x01w\x06\xb80\xfb\x9c0\ufc20\x9fPk\x91WEu4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9f\x10\xf2\xa0F;e\xae0\xb0p\xb3\xdf\x18\xcfF\xf5\x1e\x89\xbd\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\x9f\x19\xfa\u0223$7\xd8\n\u0183z\v\xb7\x84\x17)\xf4\x97.\x89#=\xf3)\x9far\x00\x00\u07d4\x9f\x1a\xa8\xfc\xfc\x89\xa1\xa52\x8c\xbdcD\xb7\x1f'\x8a,\xa4\xa0\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\x9f!0,\xa5\tk\xeat\x02\xb9\x1b\x0f\xd5\x06%O\x99\x9a=\x89C\x97E\x1a\x00=\xd8\x00\x00\u07d4\x9f'\x1d(U\x00\xd78F\xb1\x8fs>%\u074bO]J\x8b\x89'#\xc3F\xae\x18\b\x00\x00\u07d4\x9f4\x97\xf5\xef_\xe60\x95\x83l\x00N\xb9\xce\x02\xe9\x01;K\x89\"V\x86\x1b\xf9\xcf\b\x00\x00\xe0\x94\x9f:t\xfd^~\xdc\xc1\x16)\x93\x17\x13\x81\u02f62\xb7\xcf\xf0\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\x9fF\xe7\xc1\xe9\a\x8c\xae\x860Z\xc7\x06\v\x01F}f\x85\xee\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\x9fll\xb2\x06\x95c\x14M\b\x11g{\xa0\xe4q:\nAC\x89<\xd2\xe0\xbfc\xa4H\x00\x00\u07d4\x9fJq\x95\xac|\x15\x1c\xa2X\xca\xfd\xa0\u02b0\x83\xe0I\xc6\x02\x89SS\x8c2\x18\\\xee\x00\x00\u07d4\x9fJ\xc9\xc9\xe7\xe2L\xb2DJ\x04T\xfa[\x9a\xd9\xd9-8S\x89-C\xf3\xeb\xfa\xfb,\x00\x00\u07d4\x9f_D\x02kWjJ\xdbA\xe9YaV\x1dA\x03\x9c\xa3\x91\x89\r\x8drkqw\xa8\x00\x00\u07d4\x9f`{?\x12F\x9fDa!\u03bf4u5kq\xb42\x8c\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\x9fa\xbe\xb4o^\x85=\n\x85!\xc7Dnh\xe3L}\ts\x89\x1e[\x8f\xa8\xfe*\xc0\x00\x00\u07d4\x9fd\xa8\xe8\xda\xcfJ\xde0\xd1\x0fMY\xb0\xa3\u056b\xfd\xbft\x8966\x9e\xd7t}&\x00\x00\u07d4\x9ff.\x95'A!\xf1wVncm#\x96L\xf1\xfdho\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x9fj2*mF\x99\x81Bj\xe8

D\x86]~\xe0\xbb\x15\u01f3\x89\x02\xb5\xeeW\x92\x9f\u06c0\x00\u07d4\x9fy\x86\x92J\xeb\x02h|\
\xd6A\x89\x18\x9f\xb1g\xde\xd2\xdd\\\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\x9fz\x03\x92\xf8Ws.0
\x04\xa3u\xe6\xb1\x06\x8dI\xd801\x89Ik\x93[\x8b\xbd@\x00\x00\u07d4\x9f\x82E\u00eb}\x171d\x8
6\x1c\u04d9\x1b\x94\xf1\xba@\xa9:\x89\x9b\ny\x1f\x12\x110\x00\x00\u07d4\x9f\x83\xa2\x93\xc3$
\xd4\x10l\x18\xfa\xa8\x88\x8fd\u0499\x05L\xa0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\x9f\x86\xa0f\xed\xb6\x1f\xcbXV\u0793\xb7\\\x8cy\x18d\xb9{\x89Ik\x93[\x8b\xbd@\
x00\x00\u07d4\x9f\x98\xeb4\xd4iy\xb0\xa6\u078b\x05\xaaS:\x89\xb8%\xdc\xf1\x89\x04\xb0m\xbb\
xb4\x0fJ\x00\x00\xe0\x94\x9f\x9f\xe0\xc9_\x10\xfe\xe8z\xf1\xaf
r6\xc8\xf3aN\xf0/\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\x9f\xae\xa1<s4\x12\xdcKI\x04\x02\
xbf\xef'\xa09z\x9b\u00c9\x10\xce\x1d=\x8c\xb3\x18\x00\x00\u07d4\x9f\xbe\x06m\xe5r6\u0703\a%
\xd3*\x02\xae\xf9$ll^\x89Ik\x93[\x8b\xbd@\x00\x00\u07d4\x9f\xd1\x05*`Pk\u0469\xef\x00:\xfd\x9d\
x03<&}\x8e\x99\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9f\xd6Cs\xf2\xfb\u035c\x0f\xac\xa6\x05
G\xca\xd6.&\u0645\x1f\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9f\xe5\x01\xaaW\xea\u05d2x\x93
|\xd60\x8c\\\xfazV)\xfe\x89\x02\xb5\xeeW\x92\x9f\u06c0\x00\u07d4\x9f\xfc_\xe0o3\xf5\xa4\x80\xb
7Z\xa9N\xb8Um\x99z\x16\xc0\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x9f\xfc\xf5\xefF\xd93\xa
5\x19\xd1\xd1Ik\xa3\x18\x9b'Ib$\x8965\u026d\xc5\u07a0\x00\x00\u07d4\x9f\xfe\xdc\xc3k|\xc3\x12
\xad*\x9e\xdeC\x1aQO\u0334\x9b\xa3\x89$OW\x9f?\\\xa4\x00\x00\u07d4\xa0\x06&\x84Fd>\xc5\x
e8\x1ez\xcb?\x17\xf1\xc3Q\xee.\u0649\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xa0\b\x01\x98c\xc1
\xa7|\x14\x99\xeb9\xbb\u05ff-
\u05e3\x1c\xb9\x89\amA\xc6$\x94\x84\x00\x00\u07d4\xa0\t\xbf\ao\x1b\xa3\xfaW\u04a7!r\x18\xbe\
xd5VZzz\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xa0\x1e\x94v\u07c4C\x18%\xc86\xe8\x80:\x97
\xe2/\xa5\xa0\u034a\x01EB\xba\x12\xa37\xc0\x00\x00\u0794\xa0\x1f\x12\xd7\x0fD\xaa{\x11;(\\\"\x
dc\xdbE\x874T\xa7\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xa0\x1f\u0450j\x90\x85\x06\xde\xda\
xe1\xe2\b\x12\x88r\xb5n\u7489\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xa0\"\x82@\xf9\x9e\x1d\xe9\x
cb2\xd8,\x0f/\xa9\xa3\xd4K\v\xf3\x89V\xbcu\xe2\xd61\x00\x00\x00\xe0\x94\xa0+\xdedahn\x19\xac
e\f\x97\r\x06r\xe7m\xcbO\u008a\x01\xe0\x92\x96\xc37\x8d\xe4\x00\x00\u07d4\xa0,\x1e4\x06O\x0
4u\xf7\xfa\x83\x1c\xcb%\x01L:\xa3\x1c\xa2\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\xa0-
\u01aa2\x8b\x88\r\u97acTh#\xfc\xcfw@G\xfb\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xa0.?\x8fYY\x
a7\xaa\xb7A\x86\x12\x12\x9bp\x1c\xa1\xb8\x00\x10\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\x
a04\u007f\n\x98wc\x90\x16\\\x16m2\x96;\xf7M\xcd\n/\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa0
5\xa3e$\x\xf8-\xbdm\x11_\xaa\x8c\xa9F\xec\x9eh\x1d\x89\x05\xf4\xe4-
\u052f\xec\x00\x00\u07d4\xa0:=\xc7\xc53\xd1tB\x95\xbe\x95]a\xaf?R\xb5\x1a\xf5\x89\x02+\x1c\x8
c\x12'\xa0\x00\x00\u07d4\xa0E\x9e\xf3i:\xac\xd1d|\xd5\u0612\x989
L\xefS\xbe\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xa0O*\xe0*\xdd\x14\xc1/\xafe\xcb%\x90\"\u0
403\n\x8e&\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xa0l\xd1\xf3\x969l\ndFFQ\xd7\xc2\x05\xef\xaf8|\xa3\x89lj\xc
cg\u05f1\xd4\x00\x00\u07d4\xa0ri\x1c\x8d\xd7\xcdB7\xffr\xa7\\\x1a\x95\x06\xd0\xce[\x9e\x89\x14\
x0e\xc8\x0f\xa7\xee\x88\x00\x00\u07d4\xa0r\u03beb\xa9\xe9\xf6\x1c\xc3\xfb\xf8\x8a\x9e\xfb\xfe>\
x9a\x8dp\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xa0v\x82\x00\v\x1b\xcf0\x02\xf8\\\x80\xc0\xf
a)I\xbd\x1e\x82\xfd\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xa0z\xa1mt\xae\u8a63(\x8dR\xdb\
x15Q\u0553\x882\x97\x89
\x86\xac5\x10R`\x00\x00\u07d4\xa0\x8d![[j\xacHa\xa2\x81\xac~@\vx\xfe\xf0L\xbf\x89\x01\x15\x8e

F\t\x13\xd0\x00\x00\u07d4\xa0\x95\x19p\xdf\u0403/\xb8;\xda\x12\xc25E\xe7\x90Aul\x89
\x86\xac5\x10R`\x00\x00\u07d4\xa0\x9fM^\xaae\xa2\xf4\xcbu\nI\x924\x01\xda\u5410\xaf\x89\a\x9
6\xe3\xea?\x8a\xb0\x00\x00\xe0\x94\xa0\xa0\xe6R\x04T\x1f\u029b/\xb2\x82\u0355\x13\x8f\xae\x
16\xf8\t\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa0\xaa_\x02\x01\xf0M;\xbe\xb8\x98\x1
3/|\x11g\x94f\xd9\x01\x89\x01\xfb\xedR\x15\xbbL\x00\x00\u07d4\xa0\xaa\xdb\xd9P\x97\"p_m#X\x
a5\u01df7\x97\x0f\x00\xf6\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa0\xb7q\x95\x1c\xe1\xde\xee6:\xe2\xb7q\xb7>\a\u0135\xe8\x00\x89K\xe4\xe7&{j
\xe0\x00\x00\u07d4\xa0\xde\\`\x1eif5\u0198\xb7\xae\x9c\xa4S\x9f\u01f9A\xec\x89\x12\xc3\xcb\xd
7\x04\xc9w\x00\x00\u07d4\xa0\xe8\xbaf\x1bH\x15L\xf8C\xd4\u00a5\xc0\xf7\x92\xd5(\xee)\x89\x1
5\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xa0\xfc~S\xc5\xeb\xd2z*\xbd\xacE&\x1f\x84\xab;Q\xae\xfb\x
89\xa3\x13\xda\xec\x9b\xc0\xd9\x00\x00\xe0\x94\xa0\xff[L\xf0\x16\x02~\x83#I}D(\xd3\xe5\xa8;\x8
7\x95\x8a\x01e\x98\xd3\xc8>\xc0B\x00\x00\u07d4\xa1\x06F[\xbd\x19\u1dbc\xe5\r\x1b\x11W\xdcY
\tZ60\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xa1\x06\xe6\x92>\xddS\u028e\xd6P\x96\x8a\x91\b\xd
6\xcc\xfd\x96p\x8a\x02\x02\xfe\x15\x05\xaf\uc240\x00\u07d4\xa1\t\u12f0\xa3\x9c\x9e\xf8/\xa1\x95
\x97\xfc^\xd8\xe9\xebmX\x89X\xe7\x92n\xe8X\xa0\x00\x00\u07d4\xa1\x1a\x03\u013b&\xd2\x1e\xf
fg}]U\\\x80\xb2TS\xeez\x89\x03\xcb'Y\xbcA\x0f\x80\x00\u07d4\xa1\x1e\xff\xabl\xf0\xf5\x97,\xff\xe4
\xd5e\x96\xe9\x89h\x14J\x8f\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4\xa1
M\xad_V\a(\xa3\\\r\x8f\u01d4\x81\x05{\xf7s\x86\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xa1&#\x
e6)\u07d3\tg\x04\xb1`\x84\xbe,\u061dV-\xa4\x8a\x01\xcc\xc92E\x11\xe4P\x00\x00\xe0\x94\xa1*l-
\x98]\xaf\x0eO_ z\xe8Q\xaa\xf7)\xb32\u034a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94\xa13m\xfb\x96\xb6\xbc\xbeK>\xdf2\x05\xbeW#\xc9\x0f\xa
dR\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xa1;\x9d\x82\xa9\x9b<\x9b\xbaZ\xe7.\xf2\x19\x9e\xdc};\xb3l\x89lj\xccg\u05f1\xd4\
x00\x00\xe0\x94\xa1<\xfe\x82mm\x18A\u072eD;\xe8\u00c7Q\x816\xb5\xe8\x8a\x1d\xa5jK\b5\xbf\
x80\x00\x00\xe0\x94\xa1C.\xd2\u01b7wz\x88\xe8\xd4m8\x8epG\u007f
\x8c\xa5\x8a\x01\xb1\xa7\xe4\x13\xa1\x96\xc5\x00\x00\u07d4\xa1D\xf6\xb6\x0fr\xd6J!\xe30\xda\x
dbb\u0619\n\xde+\t\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa1P%\xf5\x95\xac\xdb\xf3\x11\x0fw\
u017f$G~eH\xf9\xe8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa1X\x14\x8a.\x0f>\x92\xdc,\xe3\x8f\xe
b\xc2\x01\a\xe3%<\x96\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa1a`\x85\x1d+\x9c4\x9b\x92\xe4o\x
82\x9a\xbf\xb2\x10\x945\x95\x89a\t=|,m8\x00\x00\u07d4\xa1f\xf9\x11\xc6D\xac2\x13\u049e\x0e\x
1a\xe0\x10\xf7\x94\u056d&\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa1m\x9e=c\x98aY\xa8\x00\xb4h
7\xf4^\x8b\xb9\x80\xee\v\x89n\x11u\xdaz\xd1
\x00\x00\u07d4\xa1pp\xc2\xe9\u0169@\xa4\xec\x0eIT\xc4\xd7\xd6C\xbe\x8fI\x89lk\x17\x03;6\x1c
\x80\x00\u07d4\xa1|\x9eC#\x06\x95\x18\x18\x9dR\a\xa0r\x8d\u02d20j?\x8965\u026d\xc5\u07a0\x
00\x00\u07d4\xa1\x83`\xe9\x85\xf2\x06.\x8f\x8e\xfe\x02\xad,\xbc\x91\xad\x9aZ\xad\x89\xa2\xa1]\t
Q\x9b\xe0\x00\x00\u07d4\xa1\x91\x14\x05\xcfn\x99\x9e\xd0\x11\xf0\xdd\xcd*O\xf7\u008f%&\x89\
x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xa1\x92i\x80\a\xcc\x11\xaa`=\"\x1d_\xee\xa0v\xbc\xf7\xc3\r\
x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa1\x92\xf0j\xb0R\xd5\xfd\u007f\x94\xee\xa81\x8e\x82x\x15\
xfegz\x89\a\x1f\x8a\x93\xd0\x1eT\x00\x00\u07d4\xa1\x99\x81D\x96\x8a\\p\xa6AUT\xce\xfe\u0082
F\x90\u0125\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa1\xa1\xf0\xfam
\xb5\nyO\x02\xefR\b\\\x9d\x03j\xa6\u028965\u026d\xc5\u07a0\x00\x00\u07d4\xa1\xae\x8dE@\xd
4\xdbo\xdd\xe7\x14oA[C\x1e\xb5\\y\x83\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xa1\xb4|M\x0e\

xd6\x01\x88B\xe6\xcf\xc8c\n\u00e3\x14.^k\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xa1\xc4\xf4Z\x82\xe1\xc4x\xd8E\b.\xb1\x88u\xc4\xeae9\xab\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4\xa1\xdc\xd0\xe5\xb0Z\x97|\x96#\xe5\xae/Y\xb9\xad\xa2\xf3>1\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xa1\xe48\n;\x1ft\x96s\xe2p\"\x99\x93\xeeU\xf3Vc\xb4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa1\xf1\x93\xa0Y/\x1f\xeb\x9f\xdf\xc9\n\xa8\x13xN\xb8\x04q\u0249K\xe4\xe7&{j\xe0\x00\x00\u07d4\xa1\xf2\x85@P\xf8re\x8e\xd8.R\xb0\xad{\xbc\x1c\xb9!\xf6\x89m\x03\x17\xe2\xb3&\xf7\x00\x00\u07d4\xa1\xf5\xb8@\x14\rZ\x9a\xce\xf4\x02\xac<\u00c8jh\xca\xd2H\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa1\xf7e\xc4O\xe4_y\x06w\x94HD\xbeO-
B\x16_\xbd\x89\xc7\xe9\xcf\xdev\x8e\xc7\x00\x00\u07d4\xa1\xf7\xdd\xe1\xd78\xd8\xcdg\x9e\xa1\xee\x96[\xee\"K\xe7\xd0M\x89=\x18DP\xe5\xe9<\x00\x00\u07d4\xa1\xf8\u063c\xf9\x0ew\u007f\x19\xb3\xa6Iu\x9a\xd9P'\xab\xdf\u00c9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa2\x02TrB\x80onp\xe7@X\xd6\xe5)-
\xef\xc8\xc8\u0509l\x87T\xc8\xf3\f\b\x00\x00\u07d4\xa2\r\a\x1b\x1b\x000cI}y\x90\xe1$\x9d\xab\xf3l5\xf7\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa2\r\x8f\xf6\f\xaa\xe3\x1d\x02\xe0\xb6e\xfaC]v\xf7|\x94B\x89\x1a\x8a\x90\x9d\xfc\xef@\x00\x00\u07d4\xa2\x11\xda\x03\xcc\x0e1\xec\xceS\t\x99\x87\x18QU(\xa0\x90\u07c9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa2\x14B\xab\x054\n\xdeh\xc9\x15\xf3\xc39\x9b\x99U\xf3\xf7\xeb\x89*\x03I\x19\u07ff\xbc\x00\x00\u07d4\xa2\"\"Y\u075c>=\xed\x12p\x84\xf8\b\xe9*\x18\x870,\x89\b\xc83\x9d\xaf\xedH\x00\x00\u07d4\xa2*\xde\r\xdb\\n\xf8\xd0\u034d\xe9M\x82\xb1\x10\x82\xcb.\x91\x897KW\xf3\xce\xf2p\x00\x00\u07d4\xa2L:\xb6!\x81\xe9\xa1[x\xc4b\x1eL|X\x81'\xbe&\x89\b\xcd\xe4:\x83\xd31\x00\x00\u07d4\xa2W\xadYK\u0603(\xa7\xd9\x0f\xc0\xa9\a\u07d5\xee\xca\xe3\x16\x89\x1c7\x86\xff8F\x93\x00\x00\u07d4\xa2[\bd7\xfd!\x92\u0420\xf6On\xd0D\xf3\x8e\xf3\xda2\x89\x12)\x0f\x15\x18\v\xdc\x00\x00\u07d4\xa2v\xb0X\u02d8\u060b\xee\xdbg\xe5CPl\x9a\r\x94p\u0609\x90\xaa\xfcv\xe0/\xbe\x00\x00\u07d4\xa2\x82\xe9i\xca\xc9\xf7\xa0\xe1\xc0\u0350\xf5\xd0\xc48\xacW\r\xa3\x89\"\a\xeb\x89\xfc'8\x00\x00\xe0\x94\xa2\x91\xe9\u01d9\rU-
\u046e\x16\u03bc?\xca4,\xba\xf1\u044a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xa2\x93\x19\xe8\x10i\xe5\xd6\r\xf0\x0f=\xe5\xad\xee5\x05\xec\xd5\xfb\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xa2\x96\x8f\xc1\xc6K\xac\vz\xe0\u058b\xa9l\x87Mm\xb2S\xf4\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xa2\x9d[\xdat\xe0\x03GHr\xbdX\x94\xb8\x853\xffd\u00b5\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa2\x9df\x1acv\xf6m\vt\xe2\xfe\x9d\x8f&\xc0$~\xc8L\x89\xdf3\x04\a\x9c\x13\xd2\x00\x00\u07d4\xa2\xa45\xdeD\xa0\x1b\xd0\ucc9eD\xe4vD\xe4j\f\xdf\xfb\x89\x1b\x1dDZz\xff\xe7\x80\x00\u07d4\xa2\xac\xe4\u0253\xbb\x1eS\x83\xf8\xact\xe1y\x06n\x81O\x05\x91\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xa2\xb7\x01\xf9\xf5\xcd\u041eK\xa6+\xae\xba\u3a02W\x10X\x85\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa2\u0145O\xf1Y\x9f\x98\x89,W%\xd2b\xbe\x1d\xa9\x8a\xad\xac\x89\x11\t\xff30\x10\xe7\x80\x00\u07d4\xa2\xc7\xea\xff\xdc,\x9d\x93sE
l\x90\x9aR\u07f1LG\x8f\x89\a\xc0\x86\x0eZ\x80\xdc\x00\x00\u07d4\xa2\u04aabk\t\xd6\xd4\xe4\xb1?\u007f\xfcZ\x88\xbdz\xd3gB\x89\xfb\x80xPuS\x83\x00\x00\u07d4\xa2\u04cd\xe1\xc79\x06\xf6\xa7\xcan\xfe\xb9|\xf6\xf6\x9c\xc4!\xbe\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xa2\xdce\xee%kY\xa5\xbdy)wO\x90K5\x8d\U000ed84a\x04\x83\xbc\xe2\x8b\xeb\t\xf8\x00\x00\u07d4\xa2\xe0h:\x80]\xe6\xa0^\xdb/\xfb\xb5\xe9o\x05p\xb67\u00c9\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa2\u1e2a\x90\x0e\x9c\x13\x9b?\xa1\"5OaV\xd9*\x18\xb1\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xa2\u2d54\x1e\f\x01\x94K\xfe\x1d_\xb4\xe8\xa3K\x92,\u03f1\x89\n\u05ce\xbcZ\xc6

\x00\x00\u07d4\xa2\xe4`\xa9\x89\xcb\x15V_\x9e\u0327\xd1!\xa1\x8eN\xb4\x05\xb6\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa2\xec\xce,I\xf7*\t\x95\xa0\xbd\xa5z\xac\xf1\xe9\xf0\x01\xe2*\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xa2\xf4r\xfeO\"\xb7}\xb4\x89!\x9e\xa4\x02=\x11X*\x93)\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xa2\xf7\x98\xe0w\xb0}\x86\x12N\x14\a\xdf2\x89\r\xbbKcy\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa2\xf8k\xc0a\x88N\x9e\xef\x05d\x0e\xddQ\xa2\xf7\xc0Yli\x89lID\xfeG\xec\x05\x00\x00\u07d4\xa2\xfa\x17\xc0\xfbPl\xe4\x94\x00\x8b\x95W\x84\x1c?d\x1b\x8c\xae\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa3\x04X\x8f\r\x85\f\xd8\u04cfv\xe9\xe8<\x1b\xf6>3>\u0789\x02(V\x01!l\x8c\x00\x00\u07d4\xa3\x05\x8cQszN\x96\xc5_.\xf6\xbd{\xb3X\x16~\u00a7\x89
\xdb:\xe4H\x1a\u0500\x00\u07d4\xa3\t\xdfT\u02bc\xe7\f\x95\xec03\x14\x9c\xd6g\x8ao\xd4\u03c9\f\x1f\x12\xc7Q\x01X\x00\x00\u07d4\xa3\nER\x0eR\x06\xd9\x00@p\xe6\xaf>{\xb2\xe8\xddS\x13\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xa3\x0e\n\xcbSL\x9b0\x84\xe8P\x1d\xa0\x90\xb4\xeb\x16\xa2\xc0\u0349lk\x93[\x8b\xbd@\x00\x00\u07d4\xa3
0\x95\xed\xb7\x02\x8ehq\xce\n\x84\xf5HE\x9f\x830\n\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xa3!\t\x1d0\x18\x06By\xdb9\x9d+*\x88\xa6\xf4@\xae$\x89\xadx\xeb\u016cb\x00\x00\x00\u07d4\xa3#-
\x06\x8dP\x06I\x03\xc9\xeb\xc5c\xb5\x15\xac\u0237\xb0\x97\x89l\x87T\xc8\xf3\f\b\x00\x00\xe0\x94\xa3$\x1d\x89\n\x92\xba\xf5)\b\xdcJ\xa0Irk\xe4&\xeb\u04ca\x04<-
\xa6a\xca/T\x00\x00\u07d4\xa3)F&\xec)\x84\xc4;C\xdaM]\x8eFi\xb1\x1dKY\x896\xa4\xcfcc\x19\xc0\x00\x00\u07d4\xa3,\xf7\xdd\xe2\f=\xd5g\x9f\xf5\xe3%\x84\\p\u0156&b\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa39\xa3\xd8\xca(\x0e'\xd2A[&\xd1\xfcy2(\xb6`C\x896\U00086577\x8f\xf0\x00\x00\u07d4\xa3<\xb4P\xf9[\xb4n%\xaf\xb5\x0f\xe0_\xee\xe6\xfb\x8c\xc8\xea\x89*\x11)\u0413g
\x00\x00\u07d4\xa3?p\xdaru\xef\x05q\x04\u07e7\xdbd\xf4r\xe9\xf5\xd5S\x89\x04YF\xb0\xf9\xe9\xd6\x00\x00\u07d4\xa3@v\xf8K\xd9\x17\xf2\x0f\x83B\u024b\xa7\x9eo\xb0\x8e\xcd1\x89\u3bb5sr@\xa0\x00\x00\u07d4\xa3C\x0e\x1fd\u007f2\x1e\xd3G9V##\xc7\xd6#A\vV\x8964\xfb\x9f\x14\x89\xa7\x00\x00\u07d4\xa3O\x9dV\x8b\xf7\xaf\xd9L*[\x8a_\xf5\\f\xc4\by\x99\x89\x84}P;\"\x0e\xb0\x00\x00\u07d4\xa3V\x06\xd5\x12
\xee\u007f!F\xd4\x11X.\xe4\xeeJEYn\x89\u062a\xbe\b\v\xc9@\x00\x00\u07d4\xa3VU\x1b\xb7}OE\xa6\xd7\xe0\x9f\n\b\x9ey\u0322I\u02c9\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\xa3\\\x19\x13,\xac\x195Wj\xbf\xedl\x04\x95\xfb\a\x88\x1b\xa0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa3e\x91\x8b\xfe?&'\xb9\xf3\xa8gu\xd8un\x0f\u0629K\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xa3n\r\x94\xb9Sd\xa8&q\xb6\b\xcb-
72Ea)\t\x89\b!\xd2!\xb5)\x1f\x80\x00\u07d4\xa3u\xb4\xbc$\xa2N\x1fyu\x93\xcc0+/3\x10c\xfa\\\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa3v\"\xac\x9b\xbd\xc4\xd8+u\x01]t[\x9f\x8d\xe6Z(\uc25d\xc0\\\xce(\u00b8\x00\x00\xe0\x94\xa3y\xa5\a\fP=/\xac\x89\xb8\xb3\xaf\xa0\x80\xfdE\xedK\xec\x8a\x04+\xf0kx\xed;P\x00\x00\u07d4\xa3\x80-
\x8ae\x9e\x89\xa2\xc4~\x90T0\xb2\xa8'\x97\x89P\xa7\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa3\x83\x06\xcbp\xba\xa8\u4446\xbdh\xaap\xa8=$/)\a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa3\x84vi\x1d4\x94.\xeak/v\x88\x92#\x04}\xb4az\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa3\x87\xceN\x96\x1axG\xf5`\a\\d\xe1YkVA\xd2\x1c\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\xa3\x87\xec\xde\x0e\xe4\xc8\a\x94\x99\xfd\x8e\x03G;\u060a\xd7R*\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xa3\x88:$\xf7\xf1f

_\x1aj\x99I\al&\xa7nqx\x89b\xa9\x92\xe5:\n\xf0\x00\x00\xe0\x94\xa3\x8b[\xd8\x1a\x9d\xb9\u04b2\x1d^\xc7\xc6\x05R\xcd\x02\xedV\x1b\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xa3\x90\xca\x12+\x85\x01\xee>^\a\xa8\xcaKA\x9f~M\xae\x15\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xa3\x93*1\xd6\xffu\xfb;\x12q\xac\xe7\u02a7\xd5\xe1\xff\x10Q\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xa3\x94\xadO\xd9\xe6S\x0eo\\S\xfa\xec\xbe\u0781\xcb\x17-
\xa1\x8a\x01/\x93\x9c\x99\xed\xab\x80\x00\x00\u07d4\xa3\x97\x9a\x92v\n\x13Z\xdfi\xd7/u\xe1gu_\x1c\xb8\u00c9\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xa3\x9b\xfe\xe4\xae\u027du\xbd\"\u01b6r\x89\x8c\xa9\xa1\xe9]2\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa3\xa2b\xaf\u0493h\x19#\b\x92\xfd\xe8O-
ZYJ\xb2\x83\x89e\xea=\xb7UF`\x00\x00\u07d4\xa3\xa2\xe3\x19\xe7\u04e1D\x8bZ\xa2F\x89S\x16\f-
\xbc\xbaq\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa3\xa5{\a\x16\x13(\x04\xd6\n\xac(\x11\x97\xff+=#{\x01\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\xa3\xa9>\xf9\xdb\xea&6&=\x06\xd8I/jA\u0790|\"\x89\x03@\xaa\xd2\x1b;p\x00\x00\xe0\x94\xa3\xae\x18y\x00}\x80\x1c\xb5\xf3RqjM\u063a'!\xde=\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4\xa3\xba\r:6\x17\xb1\xe3\x1bNB,\xe2i\xe8s\x82\x8d]i\x89.\x14\x1e\xa0\x81\xca\b\x00\x00\u07d4\xa3\xbc\x97\x9bp\x80\t/\xa1\xf9/n\x0f\xb3G\u2359PE\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\xa3\xbf\xf1\u07e9\x97\x16h6\f\r\x82\x82\x842\xe2{\xf5Ng\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xa3\xc1J\xce(\xb1\x92\u02f0b\x14_\u02fdXi\xc6rq\xf6\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xa3\xc3:\xfc\x8c\xb4pN#\x15=\xe2\x04\x9d5\xaeq3$r\x89+X\xad\u06c9\xa2X\x00\x00\u07d4\xa3\u0430<\xff\xbb&\x9fyj\u009d\x80\xbf\xb0}\xc7\u01ad\x06\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa3\u0543\xa7\xb6[#\xf6\vy\x05\xf3\xe4\xaab\xaa\xc8\u007fB'\x898\xbe\xfa\x12mZ\x9f\x80\x00\u07d4\xa3\xdb6J3-
\x88K\xa9;&\x17\xaeM\x85\xa1H\x9b\xeaG\x89\\(=A\x03\x94\x10\x00\x00\u07d4\xa3\xe0Q\xfbtJ\xa3A\f;\x88\xf8\x99\xf5\xd5\u007f\x16\x8d\xf1-
\x89\xa00\xdc\xeb\xbd/L\x00\x00\u07d4\xa3\xe3\xa6\xeaP\x95s\xe2\x1b\xd0#\x9e\xce\x05#\xa7\xb7\u061b/\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xa3\xf4\xad\x14\xe0\xbbD\xe2\xce,\x145\x9cu\xb8\xe72\xd3pT\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa3\xfa\xccP\x19\\\vI3\xc8X\x97\xfe\xcc[\xbd\x99\\4\xb8\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa4\x03Z\xb1\xe5\x18\b!\xf0\xf3\x80\xf1\x13\x1bs\x87\xc8\u0641\u0349\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa4\n\xa2\xbb\xce\fr\xb4\xd0\xdf\xff\xccBq[+T\xb0\x1b\xfa\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa4\x19\xa9\x84\x14#c&uuV`\x894\x0e\xea\x0e\xa2\b\x19\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94\xa4!\u06f8\x9b:\aA\x90\x84\xad\x10\xc3\xc1]\xfe\x9b2\xd0\u008a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xa4\"\xe4\xbf\v\xf7AG\u0309[\xed\x8f\x16\xd3\xce\xf3BaT\x89\x12\xef?b\xee\x116\x80\x00\u07d4\xa4%\x9f\x83E\xf7\u3a37+\x0f\xec,\xf7^2\x1f\xdaM\u0089g\x8a\x93
b\xe4\x18\x00\x00\u07d4\xa4)\b\xe7\xfeS\x98\n\x9a\xbf@D\xe9W\xa5Kp\u973e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa4)\xfa\x88s\x1f\xdd5\x0e\x8e\xcdn\xa5B\x96\xb6HO\u6549j\xc5\xc6-
\x94\x86\a\x00\x00\xe0\x94\xa40\x99]\xdb\x18[\x98e\xdb\xe6%9\xad\x90\xd2.Ks\u008a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa46\xc7TS\xcc\xcaJ\x1f\x1bb\xe5\u0123\r\x86\xdd\xe4\xbeh\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xa47\xfen\xc1\x03\u028d\x15\x8fc\xb34\"N\u032c[>\xa3\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xa4;m\xa6\xcbz\xacW\x1d\xff'\xf0\x9d9\xf8F\xf57i\xb1\x8

9\x14\x99\x8f2\xacxp\x00\x00\u07d4\xa4;\x81\xf9\x93V\xc0\xaf\x14\x1a\x03\x01\rw\xbd\x04,q\xc1\xee\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa4>\x19G\xa9$+5Ua\xc3\n\x82\x9d\xfe\uc881Z\xf8\x89\xd2=\x99\x96\x9f\u0591\x80\x00\u07d4\xa4H\x9aP\xea\xd5\xd5DZ{\xeeM-
U6\u00a7lA\xf8\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa4O\xe8\x00\xd9o\xca\xd7;qp\xd0\xf6\x10\u02cc\x06\x82\xd6\u0389\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xa4T2\xa6\xf2\xac\x9dVW{\x93\x8a7\xfa\xba\xc8\xcc|F\x1c\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa4f\xd7p\u0618\xd8\xc9\xd4\x05\xe4\xa0\xe5Q\xef\xaf\xcd\xe5<\xf9\x89\x1a\xb2\xcf|\x9f\x87\xe2\x00\x00\xe0\x94\xa4g\a1\x17X\x93\xbb\xcf\xf4\xfa\x85\u0397\xd9O\xc5\x1cK\xa8\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xa4kC\x87\xfbM\xcc\xe0\x11\xe7nMsT}D\x81\xe0\x9b\xe5\x89Hz\x9a0E9D\x00\x00\u07d4\xa4l\xd27\xb6>\xeaC\x8c\x8e;e\x85\xf6y\xe4\x86\b2\xac\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa4wy\u063c\x1c{\xce\x0f\x01\x1c\xcb9\xefh\xb8T\xf8\u078f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa4\x82kl8\x82\xfa\xd0\xed\\\x8f\xbb%\xcc@\xccO3u\x9f\x89p\x1bC\xe3D3\xd0\x00\x00\u07d4\xa4\x87Y(E\x8e\xc2\x00]\xbbW\x8c\\\xd35\x80\xf0\xcf\x14R\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa4\x9fR:\xa5\x13d\xcb\xc7\u0655\x16=4\xebY\r\xed/\b\x89\x90'B\x1b*\x9f\xbc\x00\x00\u07d4\xa4\xa4\x9f\v\xc8h\x8c\xc9\xe6\xdc\x04\xe1\xe0\x8dR\x10&\xe6Ut\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa4\xa7\xd3\x06\xf5\x10\xcdX5\x94(\xc0\xd2\xf7\xc3`\x9dVt\u05c9\xb5\x8c\xb6\x1c<\xcf4\x00\x00\u07d4\xa4\xa8:\a8y\x9b\x97\x1b\xf2\xdep\x8c.\xbf\x91\x1c\xa7\x9e\xb2\x89
\x86\xac5\x10R`\x00\x00\u07d4\xa4\xb0\x9d\xe6\xe7\x13\xdciTnv\xef\n\xcf@\xb9O\x02A\xe6\x89\x11}\xc0b~\xc8p\x00\x00\xe0\x94\xa4\u04b4)\xf1\xadSI\xe3\x17\x04\x96\x9e\xdc_%\xee\x8a\xca\x10\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\xa4\xd6\xc8.\u076eYG\xfb\xe9\xcd\xfb\xd5H\xae3\xd9\x1aq\x91\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xa4\xda4E\r\"\xec\x0f\xfc\xed\xe0\x00K\x02\xf7\x87.\xe0\xb7:\x89\x05\x0fafs\xf0\x83\x00\x00\xe0\x94\xa4\xddY\xab^Q}9\x8eI\xfaS\u007f\x89\x9f\xedL\x15\xe9]\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xa4\xe6#E\x1e~\x94\xe7\u86e5\xed\x95\u0228:b\xff\xc4\xea\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa4\xed\x11\xb0r\u061f\xb16u\x9f\u019bB\x8cH\xaa]L\xed\x89\x0e?\x15'\xa0<\xa8\x00\x00\u07d4\xa4\xfb\x14@\x9ag\xb4V\x88\xa8Y>\\\xc2\xcfYl\xedo\x11\x89a\t=|,m8\x00\x00\xe0\x94\xa5\x14\xd0\x0e\xddq\b\xa6\xbe\x83\x9ac\x8d\xb2AT\x18\x17A\x96\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\xe0\x94\xa5\"\xde~\xb6\xae\x12PR*Q13\xa9;\xd4(IG\\\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xa5$\xa8\xcc\xccIQ\x8d\x17\n2\x82p\xa2\xf8\x813\xfb\xaf]\x89\x0f\xf7\x02-
\xac\x10\x8a\x00\x00\u07d4\xa59\xb4\xa4\x01\xb5\x84\xdf\xe0\xf3D\xb1\xb4\"\xc6UC\x16~.\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xa5>\xadT\xf7\x85\n\xf2\x148\xcb\xe0z\xf6\x86'\x9a1[\x86\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa5C\xa0f\xfb2\xa8f\x8a\xa0sj\f\x9c\xd4\rx\t\x87'\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa5gw\vj\xe3
\xbd\xdeP\xf9\x04\xd6c\xe7F\xa6\x1d\xac\xe6\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa5h\xdbMW\xe4\xd6tb\xd73\u019a\x9e\x0f\xe2n!\x83'\x89;k\xff\x92f\xc0\xae\x00\x00\u07d4\xa5i\x8059\x1eg\xa4\x90\x13\xc0\x00
yY1\x14\xfe\xb3S\x89\r\x02\xabHl\xed\xc0\x00\x00\u07d4\xa5p\":\xe3\u02a8QA\x8a\x98C\xa1\xacU\xdbH$\xf4\xfd\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xa5s`\xf0\x02\xe0\xd6M-
tE}\x8c\xa4\x85~\xe0\v\xcd\u07c9\x123\xe22\xf6\x18\xaa\x00\x00\u07d4\xa5u\xf2\x89\x1d\xcf\u0368<\\\xf0\x14t\xaf\x11\xee\x01\xb7-

\u0089\x05l\xd5_\xc6M\xfe\x00\x00\u07d4\xa5x;\xf342\xff\x82\xacI\x89\x85\xd7\xd4`\xaeg\xec6s\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xa5\x87MuF5\xa7b\xb3\x81\xa5\xc4\u01d2H:\xf8\xf2=\x1d\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94\xa5\xa4\"\u007fl\xf9\x88%\xc0\u057a\xffS\x15u,\xcc\x1a\x13\x91\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa5\xabK\xd3X\x8fF\xcb'.V\xe9=\xee\u04c6\xba\x8bu=\x89HB\xf0A\x05\x87,\x80\x00\xe0\x94\xa5\xba\xd8e\t\xfb\xe0\xe0\xe3\xc0\xe9?m8\x1f\x1a\xf6\xe9\u0501\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xa5\xc36\b;\x04\xf9G\x1b\x8cn\xd76y\xb7Mf\xc3c\uc263e\nL\x9d
\xe2\x00\x00\u07d4\xa5\xcd\x129\x92\x19K4\xc4x\x13\x140;\x03\xc5IH\xf4\xb9\x89l\xfc\xc3\xd9\x1d\xa5c\x00\x00\u07d4\xa5\u0578\xb6-\x00-
\xef\x92A7\x10\xd1;o\xf8\xd4\xfc}\u04c9\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\xa5\xd9ni}F5\x8d\x11\x9a\xf7\x81\x9d\xc7\b\u007fj\xe4\u007f\xef\x8a\x03\x17\xbe\xe8\xaf3\x15\xa7\x80\x00\u07d4\xa5\xde^CO\xdc\xddh\x8f\x1c1\xb6\xfbQ,\xb1\x96rG\x01\x89+^:\xf1k\x18\x80\x00\x00\u07d4\xa5\xe0\xfc<:\xff\xed=\xb6q\tG\xd1\xd6\xfb\x01\u007f>'m\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa5\xe9;I\xea|P\x9d\xe7\xc4Ml\xfe\xdd\xefY\x10\u07aa\xf2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa5\xe9\xcdKt%]\"\xb7\u0672z\xe8\xddC\xedn\xd0%+\x89)\x8d\xb2\xf5D\x11\u0640\x00\xe0\x94\xa5\xf0\a{5\x1flP\\\xd5\x15\u07e6\xd2\xfa\u007f\\L\u0487\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xa5\xf0u\xfd@\x135W{f\x83\u0081\xe6\xd1\x01C-
\xc6\xe0\x89\x91Hx\xa8\xc0^\xe0\x00\x00\u07d4\xa5\xfe,\xe9\u007f\x0e\x8c8V\xbe\r\xe5\xf4\u0732\xce]8\x9a\x16\x89\x01=\xb0\xb8\xb6\x86>\x00\x00\u07d4\xa5\xffb\"-
\x80\xc0\x13\xce\xc1\xa0\xe8\x85\x0e\xd4\xd3T\xda\xc1m\x89\vA\a\\\x16\x8b\x18\x00\x00\u07d4\xa6\t\xc2m\xd3P\xc25\xe4K+\x9c\x1d\xdd\xcc\u0429\xd9\xf87\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa6\f\x12\tuO]\x87\xb1\x81\xdaO\b\x17\xa8\x18Y\xef\x9f\u0609\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\xe0\x94\xa6\x10\x1c\x96\x1e\x8e\x1c\x15y\x8f\xfc\xd0\xe3
\x1dw\x86\xec7:\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xa6\x13Ei\x96@\x8a\xf1\xc2\xe9>\x17w\x88\xabU\x89^+2\x8a\x01Y\x19\xffG|\x88\xb8\x00\x00\u07d4\xa6\x18\x87\x81\x8f\x91J
\xe3\x10w)\v\x83qZk-
n\xf9\x89e\xea=\xb7UF`\x00\x00\u07d4\xa6\x1aT\xdfxJD\xd7\x1bw\x1b\x871u\t!\x13\x81\xf2\x00\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa6\x1c\u06ed\xf0K\x1eT\u0203\xde`\x05\xfc\xdf\x16\xbe\xb8\xeb/\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xa69\xac\xd9k1\xbaS\xb0\u0407c\"\x9e\x1f\x06\xfd\x10^\x9d\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xa6BP\x10\x04\xc9\x0e\xa9\xc9\xed\x19\x98\xba\x14\nL\xd6,o_\x89\r\x94\xfb\x8b\x10\xf8\xb1\x80\x00\u07d4\xa6D\xed\x92,\xc27\xa3\xe5\u0117\x9a\x99Tw\xf3nP\xbcb\x89\x1f\xa7=\x84]~\x96\x00\x00\u07d4\xa6F\xa9\\moY\xf1\x04\xc6T\x1dw`uz\xb3\x92\xb0\x8c\x89\u3bb5sr@\xa0\x00\x00\xe0\x94\xa6HL\u0184\xc4\xc9\x1d\xb5>\xb6\x8aM\xa4Zjk\xda0g\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xa6N_\xfbpL,\x919\xd7~\xf6\x1d\x8c\u07e3\x1dz\x88\xe9\x89\a\xc0\x86\x0eZ\x80\xdc\x00\x00\xe0\x94\xa6T&\xcf\xf3x\xed#%5\x13\xb1\x9fIm\xe4_\xa7\u13ca\x01\x86P\x12|\xc3\u0700\x00\x00\u07d4\xa6jIc\xb2\u007f\x1e\xe1\x93+\x17+\xe5\x96N\r:\xe5KQ\x89\t`\xdbwh\x1e\x94\x00\x00\u07d4\xa6\u007f8\x81\x95eB:\xa8_>:\xb6\x1b\xc7c\u02eb\x89\u0749sw\xb0\"\u01be\b\x00\x00\u07d4\xa6\x8c14E\xc2-
\x91\x9e\xe4l\xc2\xd0\xcd\xff\x04:uX%\x89\x04\x13t\xfd!\xb0\u0600\x00\u07d4\xa6\x8e\f0\u02e3\xbcZ\x88>T\x03
\xf9\x99\xc7\xcdU\x8e\\\x89a\x9237b\xa5\x8c\x80\x00\u07d4\xa6\x90\xf1\xa4\xb2\n\xb7\xba4b\x86

\u079c\xa0@\xc4<\x19c\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xa6\x9d|\xd1}HB\xfe\x03\xf6*\x90\xb2\xfb\xf8\xf6\xaf{\xb3\x80\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xa6\xa0\x82R\xc8YQw\xcc.`\xfc'Y>#y\xc8\x1f\xb1\x89\x01\x16Q\xac>zu\x80\x00\u07d4\xa6\xa0\xdeB\x1a\xe5Om\x17(\x13\b\xf5dm/9\xf7w]\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa6\xb2\xd5s)s`\x10,\a\xa1\x8f\xc2\x1d\xf2\xe7I\x9f\xf4\xeb\x89\xd9o\u0390\u03eb\xcc\x00\x00\xe0\x94\xa6\xc9\x10\xceMIJ\x91\x9c\u036a\xa1\xfc;\x82\xaat\xba\x06\u03ca\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xa6\u3ea3\x8e\x10J\x1e'\xa4\xd8(i\xaf\xb1\xc0\xaen\xff\x8d\x89\x01\x11@\ueb4bq\x00\x00\u07d4\xa6\xee\xbb\xe4d\u04d1\x87\xbf\x80\u029c\x13\xd7
'\xec[\xa8\xbe\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xa6\xf6+\x8a=\u007f\x11\"\a\x01\xab\x9f\xff\xfc\xb3'\x95\x9a'\x85\x89\x1bn)\x1f\x18\u06e8\x00\x00\u07d4\xa6\xf93\a\xf8\xbc\xe01\x95\xfe\u0387
C\xe8\xa0?{\xd1\x1a\x89\x9csK\xadQ\x11X\x00\x00\u07d4\xa7\x01\xdfy\xf5\x94\x90\x1a\xfe\x14DH^k
\u00fd\xa2\xb9\xb3\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xa7\x02L\xfdt,\x1e\xc1<\x01\xfe\xa1\x8d0B\xe6_\x1d]\xee\x8a\x02c\x11\x9a(\xab\u0430\x80\x00\u07d4\xa7\x18\xaa\xadY\xbf9\\\xba+#\xe0\x9b\x02\xfe\f\x89\x81bG\x8960<\x97\xe4hx\x00\x00\u07d4\xa7$|S\xd0Y\xeb|\x93\x10\xf6(\xd7\xfclj\nw?\b\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xa7%7c\xcfJu\u07d2\xca\x1evm\xc4\xee\x8a'E\x14{\x8a\x02F7p\xe9\n\x8fP\x00\x00\u07d4\xa7.\xe6f\u0133^\x82\xa5\x06\x80\x8bD<\xeb\xd5\xc62\xc7\u0749+^:\xf1k\x18\x80\x00\x00\u07d4\xa7DD\xf9\x0f\xbbT\xe5o:\u0276\xcf\u032aH\x19\xe4aJ\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa7GC\x9a\xd0\u04d3\xb5\xa08a\xd7r\x962m\u8edd\xb9\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa7`{BW;\xb6\xf6\xb4\xd4\xf2<~*&\xb3\xa0\xf6\xb6\xf0\x89WG=\x05\u06ba\xe8\x00\x00\xe0\x94\xa7i)\x89\n{G\xfb\x85\x91\x96\x01lo\u0742\x89\u03b7U\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u0794\xa7kt?\x98\x1bi0r\xa11\xb2+\xa5\x10\x96\\/\xef\u05c8\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xa7m?\x15bQ\xb7,\f\xcfKG\xa39<\xbdoI\xa9\u0149Hz\x9a0E9D\x00\x00\u07d4\xa7t(\xbc\xb2\xa0\xdbv\xfc\x8e\xf1\xe2\x0eF\x1a\n2\u016c\x15\x89\x15\xbeat\xe1\x91.\x00\x00\u07d4\xa7u\x8c\xec\xb6\x0e\x8faL\u0396\x13~\xf7+O\xbd\awJ\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xa7w^J\xf6\xa2:\xfa
\x1f\xb7\x8b\x91^Q\xa5\x15\xb7\xa7(\x89\x06\x81U\xa46v\xe0\x00\x00\u07d4\xa7\u007f>\u1793\x88\xbb\xbb\"\x15\xc6#\x97\xb9e`\x13#`\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xa7\x85\x9f\xc0\u007fun\xa7\xdc\xeb\xbc\xcdB\xf0X\x17X-
\x97?\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa7\x96lH\x9fLt\x8az\u902a'\xa5t%\x17g\xca\xf9\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xa7\xa3\xbba9\xb0\xad\xa0\f\x1f\u007f\x1f\x9fV\u0654\xbaM\x1f\xa8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa7\xa3\xf1S\xcd\u00c8!\xc2\f]\x8c\x82A\xb2\x94\xa3\xf8+$\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xa7\xa5\x17\u05ed5\x82\v\t\u0517\xfa~U@\xcd\xe9IXS\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xa7\xc9\u04c8\xeb\xd8s\xe6k\x17\x13D\x83\x97\xd0\xf3\u007f\x8b\u04e8\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xa7\u073b\xa9\xb9\xbfgb\xc1EAlPjq\u3d17
\x9c\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xa7\xe7O\v\xdb'\x8f\xf0\xa8\x05\xa6Ha\x8e\xc5+\x16o\xf1\xbe\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xa7\xe87r\xbc
\x0f\x90\x06\xaa*&\r\xba\xa8H=\xc5+0\x89\vB\xd56f7\xe5\x00\x00\xe0\x94\xa7\xef5\u0387\xed\xa6\u008d\xf2HxX\x15\x05>\xc9zPE\x8a\x01\x0f\f\xe9I\xe0\x0f\x93\x00\x00\u07d4\xa7\xf9\"\f\x80G\

x82k\xd5\xd5\x18?Ngjmw\xbf\xed6\x89\bPh\x97k\xe8\x1c\x00\x00\u07d4\xa8\a\x10O'\x03\xd6y\xf8\u07af\xc4B\xbe\xfe\x84\x9eB\x95\v\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa8\f\xb1s\x8b\xac\b\xd4\xf9\xc0\x8bM\xef\xf5\x15T_\xa8XO\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xa8\x19\xd2\xec\xe1\"\xe0(\xc8\xe8\xa0J\x06M\x02\xb9\x02\x9b\b\xb9\x8965\u026d\xc5\u07a0\x00\x00\u0794\xa8%\xfdZ\xbby&\xa6|\xf3k\xa2F\xa2K\xd2{\xe6\xf6\xed\x88\xf4?\xc2\xc0N\xe0\x00\x00\u07d4\xa8(U9\x86\x9d\x88\xf8\xa9aS7Uq}~\xb6Uv\xae\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xa83\x82\xb6\xe1Rg\x97J\x85P\xb9\x8fqv\xc1\xa3S\xf9\xbe\x89\xbf\xfd\xaf/\xc1\xb1\xa4\x00\x00\xe0\x94\xa8DlG\x81\xa77\xacC(\xb1\xe1[\x8a\v?\xbb\x0f\xd6h\x8a\x04\x87\x94\xd1\xf2F\x19*\x00\x00\u07d4\xa8E[A\x17e\u0590\x1e1\x1erd\x03\t\x1eB\xc5f\x83\x89\xb7:\xec;\xfe\x14P\x00\x00\xe0\x94\xa8f\x13\xe6\u0124\xc9\xc5_\\\x10\xbc\xda2\x17]\u02f4\xaf`\x8a\x02C\xd6\xc2\xe3k\xe6\xae\x00\x00\u07d4\xa8m\xb0}\x9f\x81/G\x96b-
@\xe0=\x13Xt\xa8\x8at\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa8\u007fz\xbdo\xa3\x11\x94(\x96x\xef\xb6<\xf5\x84\xee^*a\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xa8\x80\u2a3f\x88\xa1\xa8&H\xb4\x01<I\xc4YLC<\u020a\x01\x00N.E\xfb~\xe0\x00\x00\u07d4\xa8\x85w\xa0s\xfb\xaf3\xc4\xcd
.\x00\xeap\xefq\x1b@\x06\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa8\x91L\x95\xb5`\xec\x13\xf1@
Ws8\xc3+\u02f7}:z\x89\t\xc2\x00vQ\xb2P\x00\x00\xe0\x94\xa8\x9a\xc9;#7\x04r\u06ac3~\x9a\xfd\xf6BT?>W\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa8\x9d\xf3HY\xed\xd7\xc8
\u06c8w@\xd8\xff\x9e\x15\x15|{\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa8\xa4<\x00\x91\x00al\xb4\xaeN\x03?\x1f\xc5\xd7\xe0\xb6\xf1R\x89\u0548\xd0x\xb4?M\x80\x00\u07d4\xa8\xa7\b\xe8O\x82\u06c6\xa3U\x02\x19;Ln\xe9\xa7n\xbe\x8f\x897\b\xba\xed=h\x90\x00\x00\xe0\x94\xa8\xa7\xb6\x8a\u06b4\xe3\xea\xdf\xf1\x9f\xfaX\xe3J?\xce\xc0\xd9j\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xa8\xa8\xdb\xdd\x1a\x85\u047e\xee%i\xe9\x1c\xccM\t\xae\u007fn\xa1\x8a\x01:k+VHq\xa0\x00\x00\u07d4\xa8\xac\xa7H\xf9\xd3\x12\xect\u007f\x8bex\x14&\x94\xc7\xe9\xf3\x99\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa8\xb6[\xa3\x17\x1a?w\xa65\v\x9d\xaf\x1f\x8dU\xb4\xd2\x01\xeb\x89(b\xf3\xb0\xd2\"\x04\x00\x00\u07d4\xa8\xbe\xb9\x1c+\x99\u0216J\xa9[kJ\x18K\x12i\xfc4\x83\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u0794\xa8\xc0\xb0/\xaf\x02\xcbU\x19\u0768\x84\xde{\xbc\x8c\x88\xa2\u0681\x88\xe7\xc2Q\x85\x05\x06\x00\x00\u07d4\xa8\xc1\u05aaA\xfe=e\xf6{\xd0\x1d\xe2\xa8f\xed\x1e\u066eR\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\u07d4\xa8\xca\xfa\xc3\"\x80\xd0!\x02\v\xf6\xf2\xa9x(\x83\u05ea\xbe\x12\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xa8\xdb\v\x9b
\x14S3<u\u007fj\u067c\xb5U\xc0-
\xa9;\x89wB\xb7\x83\x0f4\x1d\x00\x00\u07d4\xa8\xe4*N3\xd7Rl\xca\x19\u0663m\xcdn\x80@\xd0\xeas\x89:\x8c\x02\xc5\xea-\xe0\x00\x00\u07d4\xa8\xe7
\x1f\xf6\x19\xfa\xff\xc32\xe6\xad7\xedA\xe3\x01\xbf\x01J\x89
\x86\xac5\x10R`\x00\x00\u07d4\xa8\xee\x1d\xf5\xd4K\x12\x84i\xe9\x13V\x9e\xf6\xac\x81\xee\xdaO\u0209\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xa8\xef\x9a\xd2tC`B\x90>A<;\fb\xf5\xf5.\u0544\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa8\xf3\u007f\n\xb3\xa1\xd4H\xa9\xe3\xce@\x96_\x97\xa6F\b:4\x89\x11\xe0\xe4\xf8\xa5\v\xd4\x00\x00\u07d4\xa8\xf8\x9d\xd5\xccnd\u05f1\xee\xac\xe0\a\x02\x02,\xd7\xd2\xf0=\x89%\xf2s\x93=\xb5p\x00\x00\xe0\x94\xa9\x04v\xe2\xef\xdf\xeeO8{\x0f2\xa5\x06x\xb0\xef\xb5s\xb5\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa9\x14PF\xfa6(\xcf_\xd4\xc6\x13\x92{\xe51\xe6\xdb\x1f\u0749\x06\x12O\xee\x99;\xc0\x00\x00\u07d4\xa9\x14\u0375q\xbf\xd9=d\xdaf\xa4\xe1\b\xea\x13NP\xd0\x00\x89M\x878\x99G\x13y\x80\x00\xe0\x94

\xa9\x1aZ{4\x1f\x99\xc55\x14N
\xbe\x9ck;\xb4\u008eM\x8a\x01&u:\xa2$\xa7\v\x00\x00\u07d4\xa9%%Q\xa6$\xaeQ7\x19\u06beR\
a\xfb\xef\xb2\xfdwI\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xa9'\u050b\xb6\u02c1K\xc6\t\xcb\u0
2a9\x15\x1f]E\x9a'\xe1\x89\x0e\xb95\t\x00d\x18\x00\x00\u07d4\xa9)\u023dq\xdb\f0\x8d\xac\x06\b
\n\x17G\xf2\x1b\x14e\xaa\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xa9K\xbb\x82\x14\u03cd\
xa0\xc2\xf6h\xa2\xacs\xe8bHR\x8dK\x894\n\xad!\xb3\xb7\x00\x00\x00\u07d4\xa9Q\xb2D\xffP\u03
eeY\x1d^\x1a\x14\x8d\xf6\xa98\xef*\x1a\x89^\x00\x15\x84\xdf\xcfX\x00\x00\u07d4\xa9`\xb1\xca\x
dd;\\\x1a\x8el\xb3\xab\xca\xf5.\xe7\xc3\xd9\xfa\x88\x89R\x8b\xc3T^Rh\x00\x00\u07d4\xa9a\x17\x
1fSB\xb1s\xddp\xe7\xbf\xe5\xb5\xca#\x8b\x13\xbc\u0749\xb8'\x94\xa9$O\f\x80\x00\u07d4\xa9u\x
b0w\xfc\xb4\u030e\xfc\xbf\x83\x84Y\xb6\xfa$:AY\u0589\x02+\x1c\x8c\x12'\xa0\x00\x00\xe0\x94\x
a9{\xeb:H\xc4_\x15((L\xb6\xa9_}\xe4S5\x8e\u018a\x06\x90\x83l\n\xf5\xf5`\x00\x00\u07d4\xa9~\a!
DI\x9f\xe5\xeb\xbd5J\xcc~~\xfbX\x98]\b\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\xa9\x86v/zO)O.\v\x
172y\xad,\x81\xa2\"4X\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xa9\x8f\x10\x985\xf5\xea\xcd\x
05Cd|4\xa6\xb2i\xe3\x80/\xac\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xa9\x97\xdf\u01d8j'\x05\
bH\xfa\x1cd\u05e7\xd6\xe0z\u0322\x89\a\xc0\x86\x0eZ\x80\xdc\x00\x00\u07d4\xa9\x99\x91\u03b
d\x98\xd9\xc88\xc2_zt\x16\xd9\xe2D\xca%\r\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xa9\xa1\xc
d\xc3;\xfd7o\x1c\rv\xfbl\x84\xb6\xb4\xac'Mh\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xa9\xa8\xec\xa1\x1a#\xd6F\x89\xa2\xaa>A}\xbb=3k\xb5\x9a\x89\x0e4S\xcd;g\xba
\x80\x00\u07d4\xa9\xac\xf6\x00\b\x1b\xb5[\xb6\xbf\xba\xb1\x81_\xfcN\x17\xe8Z\x95\x89\n\u05ce\
xbcZ\xc6
\x00\x00\xe0\x94\xa9\xad\x19&\xbcf\xbd\xb31X\x8e\xa8\x197\x88SM\x98,\x98\x8a\x06ZM\xa2]0\x
16\xc0\x00\x00\u07d4\xa9\xaf!\xac\xbeH/\x811\x89j\"\x806\xbaQ\xb1\x94S\u00c9\x02\xb5\xe0!\x9
8\f\xc1\x80\x00\xe0\x94\xa9\xb2\xd2\xe0IN\xab\x18\xe0}7\xbb\xb8V\xd8\x0e\x80\xf8L\u04ca\x02\x
1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xa9\xbaoA;\x82\xfc\xdd\xf3\xaf\xfb\xbd\u0412\x87\xdc\xf5
\x04\x15\u0289\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xa9\xbe\x88\xad\x1eQ\x8b\v\xbb\x02J\xb1
\xd8\xf0\xe7?y\x0e\fv\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\xa9\xbf\xc4\x10\xdd\xdb
q\x1eE\xc0s\x87\xea\xb3\n\x05N\x19\xac\x89>\x99`\x1e\xdfNS\x00\x00\u07d4\xa9\u0522\xbc\xbe
[\x9e\bi\xd7\x0f\x0f\xe2\xe1\u05aa\xcdE\xed\u0149\n\xc6\xe7z\xb6c\xa8\x00\x00\xe0\x94\xa9\xd6
KO;\xb7\x85\a\"\xb5\x8bG\x8b\xa6\x917^\"NB\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xa9\xd
6\xf8q\xcax\x1au\x9a
\xac:\u06d7,\xf1()\xa2\b\x892$\xf4'#\xd4T\x00\x00\xe0\x94\xa9\xdc\x04$\u0196\x9dy\x83X\xb3\x9
3\xb1\x93:\x1fQ\xbe\xe0\n\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xa9\xe1\x94f\x1a\xacpN\xe9\
u07a0C\x97N\x96\x92\xde\xd8J]\x89\x1a&\xa5\x14\"\xa0p\x00\x00\u07d4\xa9\xe2\x837\xe65q\x9
3\xd9\xe2\xcb#k\x01\xbeD\xb8\x14'\u07c9wC\"\x17\xe6\x83`\x00\x00\u07d4\xa9\xe6\xe2^ekv%Xa
\x9f\x14z!\x98[\x88t\xed\xfe\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xa9\xe9\xdb\xcez,\xb06\x94y\x98
\x97\xbe\xd7\xc5M\x15_\u06a8\x89\n\xb5\xae\x8f\u025de\x80\x00\u07d4\xa9\xed7{}n\xc2Yq\xc1\
xa5\x97\xa3\xb0\xf3\xbe\xadW\u024f\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xaa\x02\x00\xf1\
xd1~\x9cT\xda\x06G\xbb\x969]W\xa7\x858\u06099>\xf1\xa5\x12|\x80\x00\x00\u07d4\xaa\f\xa3ss7
\x17\x8a\f\xaa\xc3\t\x9cXK\x05lV0\x1c\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4\xaa\x13kG\x96+\xb8\xb
4\xfbT\r\xb4\xcc\xf5\xfd\xd0B\xff\xb8\u03c9\x1b\x1bk\u05efd\xc7\x00\x00\xe0\x94\xaa\x14B-
o\n\xe5\xa7X\x19N\xd1W\x80\xc88\xd6\u007f\x1e\xe1\x8a\x06\t2\x05lD\x9d\xe8\x00\x00\u07d4\x
aa\x16&\x9a\xac\x9c\r\x800h\xd8/\u01d1Q\xda\xdd3Kf\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x

94\xaa\x16p&\u04da\xb7\xa8V5\x94N\xd9\xed\xb2\xbf\xeb\xa1\x18P\x8a\x01\xc1\xd5\xe2\x1bO\xcfh\x00\x00\u07d4\xaa\x1b7h\xc1m\x82\x1fX\x0ev\xc8\xe4\xc8\xe8m}\u01c8S\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xaa\x1d\xf9.Q\xdf\xf7\v\x19s\xe0\xe9$\xc6b\x87\xb4\x94\xa1x\x89\x1c\xf8J0\xa0\xa0\xc0\x00\x00\u07d4\xaa,g\x00\x96\xd3\xf990S%B~\xb9U\xa8\xa6\r\xb3\u0149l\x95Y\x06\x99#-
\x00\x00\u07d4\xaa15\xcbT\xf1\x02\xcb\xef\xe0\x9e\x96\x10:\x1ayg\x18\xffT\x89\x03\"\"\xd9\xc31\x94\x00\x00\u07d4\xaa2\x1f\xdb\xd4I\x18\r\xb8\xdd\xd3O\x0f\xe9\x06\xec\x18\xee\t\x14\x89%\"H\u07b6\xe6\x94\x00\x00\xe0\x94\xaa9%\xdc\"\v\xb4\xae!w\xb2\x880x\xb6\xdc4l\xa1\xb2\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xaa?)`\x1a\x131t^\x05\xc4(0\xa1^q\x93\x8ab7\x89\\(=A\x03\x94\x10\x00\x00\xe0\x94\xaaG\xa4\xff\xc9y622\u025b\x99\xfa\xda\x0f'4\xb0\xae\xee\x8a\x01\xb8H\x9d\xf4\xdb\xff\x94\x00\x00\u07d4\xaaI=?O\xb8fI\x1c\xf8\xf8\x00\xef\xb7\xe22N\xd7\xcf\xe5\x89\\(=A\x03\x94\x10\x00\x00\u07d4\xaaV\xa6]\u012b\xb7/\x11\xba\xe3+o\xbb\aDG\x91\xd5\u0249(\x94\xe9u\xbfII\x00\x00\xe0\x94\xaaZ\xfc\xfd\x83\t\xc2\u07dd\x15\xbe^jPN}pf$\u014a\x01<\xf4\"\xe3\x05\xa17\x80\x00\u07d4\xaa\x8e\xb0\x82;\a\xb0\xe6\xd2\n\xad\xda\x0e\x95\xcf85\xbe\x19.\x89\x01\xbc\x16\xd6t\xec\x80\x00\x00\u07d4\xaa\x91#~t\r%\xa9/\u007f\xa1F\xfa\xa1\x8c\xe5m\xc6\xe1\xf3\x892$\xf4'#\xd4T\x00\x00\u07d4\xaa\x96\x0e\x10\xc5#\x91\xc5N\x158|\xc6z\xf8'\xb51m\u0309lk\x93[\x8b\xbd@\x00\x00\u07d4\xaa\x9b\xd4X\x955\xdb'\xfa+\xc9\x03\xca\x17\xd6y\xddeH\x06\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xaa\xa8\xde\xfe\x11\xe3a?\x11\x06\u007f\xb9\x83bZ\b\x99Z\x8d\xfc\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xaa\xaa\xe6\x8b2\x14\x02\xc8\xeb\xc14h\xf3A\xc6<\f\xf0?\u0389Rf<\u02b1\xe1\xc0\x00\x00\u07d4\xaa\xad\x1b\xaa\xdeZ\xf0N+\x17C\x9e\x93Y\x87\xbf\x8c+\xb4\xb9\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xaa\xb0\n\xbfX(\xd7\xeb\xf2kG\u03ac\u0378\xba\x032Qf\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xaa\xbd\xb3\\\x15\x14\x98J\x03\x92\x13y?3E\xa1h\xe8\x1f\xf1\x89\x10\xca\u0216\xd29\x00\x00\x00\u07d4\xaa\xca`\xd9\xd7\x00\u7156\xbb\xbb\xb1\xf1\xe2\xf7\x0fF'\xf9\u060965\xbbw\xcbK\x86\x00\x00\u07d4\xaa\xce\u0629V;\x1b\xc3\x11\xdb\xdf\xfc\x1a\xe7\xf5u\x19\xc4D\f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xaa\u04b7\xf8\x10f\x95\a\x8el\x13\x8e\xc8\x1at\x86\xaa\xca\x1e\xb2\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xaa\xe6\x1eC\xcb\r\f\x96\xb3\x06\x99\xf7~\x00\xd7\x11\u0423\x97\x9b\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xaa\xe72\xed\xa6Y\x88\u00e0\f\u007fG/5\x1cF;\x1c\x96\x8e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xaa\xf0#\xfe\U0009091b\xb7\x8b\xb7\xab\xc9]f\x9cP\xd5(\xb0\x89\n\u05ce\xbcZ\xc6 \x00\x00\xe0\x94\xaa\xf5\xb2\a\xb8\x8b\r\xe4\xac@\xd7G\xce\xe0n\x17-
\xf6\xe7E\x8a\x06\xa7\xb7\x1d\u007fQ\u0410\x00\x00\u07d4\xaa\xf9\xeeK\x88lm\x1e\x95Io\xd2t#[\xf4\xec\xfc\xb0}\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\xaa\xfb{\x01:\xa1\xf8T\x1c~2{\xf6P\xad\xbd\x19L \x8f\x89I\x9e\t-
\x01\xf4x\x00\x00\xe0\x94\xab\t\x863\xee\xee\f\xce\xfd\xf62\xf9WTV\xf6\u0740\xfc\x86\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4\xab\f\xedv.\x16a\xfa\xe1\xa9*\xfb\x14\b\x88\x94\x13yH%\x89g\x8a\x93
b\xe4\x18\x00\x00\xe0\x94\xab\x14\xd2!\xe3=TF)\x19\x8c\u0416\xedc\u07e2\x8d\x9fG\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xab
\x9f\u0729y\u0426G\x01\n\xf9\xa8\xb5/\xc7\xd2\r\x8c\u044a\x01\xee\xe2S,|-
\x04\x00\x00\u07d4\xab'\xbax\xc8\xe5\xe3\xda\xef1\xad\x05\xae\xf0\xff\x03%r\x1e\b\x89\x19^\xce\x00n\x02\xd0\x00\x00\u07d4\xab(q\xe5\a\u01fe9eI\x8e\x8f\xb4b\x02Z\x1a\x1cBd\x89*\x03I\x19\u

07ff\xbc\x00\x00\u07d4\xab8a\"o\xfe\xc1(\x91\x87\xfb\x84\xa0\x8e\xc3\xed\x042d\xe8\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xab=\x86\xbc\x82\x92~\f\xd4!\xd1F\xe0\u007f\x91\x93'\xcd\xf6\xf9\x89g\x8a\x93
b\xe4\x18\x00\x00\xe0\x94\xab>b\xe7z\x8b\"^A\x15\x92\xb1\xaf0\aR\xfeA$c\x8a\x02\x15\xf85\xbcv\x9d\xa8\x00\x00\u07d4\xab>x)K\xa8\x86\xa0\xcf\xd5\xd3H\u007f\xb3\xa3\a\x8d3\x8dn\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xab@\x04\xc0@?~\xab\xb0\xeaXo!!V\xc4
=g\xf1\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xabAo\xe3\rX\xaf\xe5\xd9EL\u007f\xce\u007f\x83\v\xccu\x03V\x89\x0657\x01\xc6\x05\u06c0\x00\u07d4\xabEr\xfb\xb1\xd7+W]i\xecj\xd1s3\x87>\x85R\xfc\x89lj\xc5L\xdahG\x00\x00\u07d4\xabZy\x01av2\ts\xe8\xcd8\xf67U0\x02%1\xc0\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xab]\xfc\x1e\xa2\x1a\xdcB\u03cc?n6\x1e$?\xd0\xdaa\xe5\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xabke\xea\xb8\xdf\xc9\x17\xec\x02Q\xb9\xdb\x0e\u03e0\xfa\x03(I\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xabp\x91\x93.K\u00dd\xbbU#\x80\u0293O\xd7\x16m\x1en\x89\xb5\x0f\u03ef\xeb\xec\xb0\x00\x00\u07d4\xabt\x16\xff2%IQ\u02fcbN\xc7\xfbE\xfc~\u02a8r\x89\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\xab|B\xc5\xe5-
d\x1a\a\xadu\t\x9cb\x92\x8b\u007f\x86b/\x89\x126\x1a\xa2\x1d\x14\xba\x00\x00\u07d4\xab}T\xc7\xc6W\x0e\xfc\xa5\xb4\xb8\xcep\xf5*Ws\xe5\xd5;\x89\x0f(:\xbe\x9d\x9f8\x00\x00\u07d4\xab~\v\x83\xed\x9aBLm\x1ejo\x87\xa4\xdb\xf0d\t\xc7\u0589\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4\xab\x84\xa0\xf1G\xad&T\x00\x00+\x85\x02\x9aA\xfc\x9c\xe5\u007f\x85\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xab\x93\xb2n\xce\n\n\xa2\x13e\xaf\xed\x1f\xa9\xae\xa3\x1c\xd5Dh\x89W+{\x98sl
\x00\x00\u07d4\xab\x94\x8aJ\xe3y\\\xbc\xa11&\xe1\x92S\xbd\xc2\x1d:\x85\x14\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xab\x9a\xd3n\\t\xce.\x969\x9fW\x83\x941\xd0\u77d6\xab\x89\b\xe3\xf5\v\x17<\x10\x00\x00\u07d4\xab\xb2\xe6\xa7*@\xban\xd9\b\u037c\xec<V\x12X12\xfe\x89O%\x91\xf8\x96\xa6P\x00\x00\u07d4\xab\xc0h\xb4\x97\x9b\x0e\xa6Jb\u04f7\xaa\x89}s\x81\r\xc53\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xab\xc4_\x84\xdbs\x82\xdd\xe5L_}\x898\xc4/O:;\u0109\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xab\xc4\xca\xebGMF'\xcbn\xb4V\xec\xba\x0e\xcd\b\xed\x8a\xe1\x89\u0556{\xe4\xfc?\x10\x00\x00\xe0\x94\xab\xc7G\x06\x96I`\xdf\xe0\u0723\u0727\x9e\x92\x16\x05o\x1c\xf4\x8a\bxg\x83&\xea\xc9\x00\x00\x00\xe0\x94\xab\u0269\x9e\x8a!H\xa5Zm\x82\xbdQ\xb9\x8e\xb59\x1f\u06ef\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xab\u037c\x8f\x1d\xd1:\xf5x\u0524wJb\x18+\xed\xf9\xf9\xbe\x89\x01\xfc\xc2{\xc4Y\xd2\x00\x00\u07d4\xab\xd1T\x905\x13\xb8\xdaO\x01\x9fh(K\x06V\xa1\xd0\x16\x9b\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xab\xd2\x1e\xff\x95O\u01a7\xde&\x91*|\xbb0:f\a\x80N\x89R<\x9a\xa6\x96\xeb\x94\x00\x00\u07d4\xab\xd4\xd6\xc1fcX\xc0@o\xdf:\xf2H\xf7\x8e\u0383\x01\x04\x89r}\xe3J$\xf9\x00\x00\x00\u07d4\xab\xd9`[>\x91\xac\xfdwx0\xd1dcG\x8a\xe0\xfcw
\x89\a?u\u0460\x85\xba\x00\x00\xe0\x94\xab\u071f\x1b\xcfM\x19\xee\x96Y\x100\xe7r\xc340/}\x83\x8a\b~^\x11\xa8\x1c\xb5\xf8\x00\x00\u07d4\xab\xde\x14{*\xf7\x89\ua946T~f\xc4\xfa&d\xd3(\xa4\x89\rk`\x81\xf3L\x12\x80\x00\xe0\x94\xab\xe0|\xedj\xc5\xdd\xf9\x91\xef\xf6\xc3\xda\"jt\x1b\xd2C\xfe\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xab\xf1/\xa1\x9e\x82\xf7lq\x8f\x01\xbd\xca\x00\x03gE#\xef0\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xab\xf7(\u03d3\x12\xf2!(\x02NpF\xc2Q\xf5\xdcY\x01\xed\x8a\x06A\xe8\xa15c\xd8\xf8\x00\x00\u07d4\xab\xf8\xff\xe0p\x8a\x99\xb5(\xcc\x1e\xd4\xe9\xceK\r\x060\xbe\x8c\x89z\xb5\u00ae\xee\xe68\x00\x00\u07d4\xab\xfc\xf5\xf2P\x91\xceW\x87_\xc6t\xdc\xf1\x04\xe2\xa7=\xd2\xf2\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4\xab\xfe\x93d%\xdc\u

01f7K\x95P\x82\xbb\xaa\xf2\xa1\x1dx\xbc\x05\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\xac\x02OYO\x95X\xf0ICa\x8e\xb0\xe6\xb2\xeeP\x1d\xc2r\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xac\x12*\x03\xcd\x05\x8c\x12._\xe1{\x87/Hw\xf9\u07d5r\x89j\xc5\xc6-
\x94\x86\a\x00\x00\u07d4\xac\x14.\xda\x11W\xb9\xa9\xa6C\x90\xdf~j\xe6\x94\xfa\u0249\x05\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xac\x1d\xfc\x98Kq\xa1\x99)\xa8\x1d\x81\xf0J|\xbb\x14\a7\x03\x89
\x86\xac5\x10R`\x00\x00\xe0\x94\xac!\xc1\xe5\xa3\xd7\xe0\xb5\x06\x81g\x9d\xd6\u01d2\xdb\u0287\xde\u02ca\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\xe0\x94\xac(\x89\xb5\x96o\f\u007f\x9e\xdbB\x89\\\xb6\x9d\x1c\x04\xf9#\xa2\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xac(\xb5\xed\xea\x05\xb7o\x8c_\x97\bEA'|\x96ijL\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xac,\x8e\t\xd0d\x93\xa68XC{\xd2\v\xe0\x19bE\x03e\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\xac.vm\xac?d\x8fcz\xc6q?\u0770h\xe4\xa4\xf0M\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94\xac9\x00)\x8d\xd1M|\xc9mJ\xbbB\x8d\xa1\xba\xe2\x13\xff\xed\x8a\x05<\xa1)t\x85\x1c\x01\x00\x00\u07d4\xac=\xa5&\xcf\u0388)s\x02\xf3LI\xcaR\r\xc2q\xf9\xb2\x89+^:\xf1k\x18\x80\x00\x00\u07d4\xacD`\xa7nm\xb2\xb9\xfc\xd1R\xd9\xc7q\x8d\x9a\xc6\xed\x8co\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xacJ\xcf\xc3n\xd6\tJ'\xe1\x18\xec\xc9\x11\xcdG>\x8f\xb9\x1f\x89a\x91>\x14@<\f\x00\x00\u07d4\xacL\xc2V\xaet\xd6$\xac\xe8\r\xb0x\xb2
\u007fW\x19\x8fk\x89lyt\x12?d\xa4\x00\x00\u07d4\xacN\xe9\xd5\x02\xe7\xd2\xd2\xe9\x9eY\xd8\xca}_\x00\xc9KM\u058965\u026d\xc5\u07a0\x00\x00\u07d4\xacR\xb7~\x15fH\x14\xf3\x9eO'\x1b\xe6A0\x8d\x91\xd6\u0309\v\xed\x1d\x02c\xd9\xf0\x00\x00\u07d4\xacY\x99\xa8\x9d-
\u0486\u0568\fm\xee~\x86\xaa\xd4\x0f\x9e\x12\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\xac_br1H\r\r\x950.m\x89\xfc2\xcb\x1dO\xe7\xe3\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xac`\x8e+\xac\x9d\xd2\a(\u0494~\xff\xbb\xbf\x90\n\x9c\xe9K\x8a\x01EK\r\xb3uh\xfc\x00\x00\u07d4\xacm\x02\xe9\xa4k7\x9f\xacJ\u0271\u05f5\xd4{\xc8P\xce\x16\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\xacoh\xe87\xcf\x19a\xcb\x14\xabGDm\xa1h\xa1m\u0789\x89Hz\x9a0E9D\x00\x00\u07d4\xacw\xbd\xf0\x0f\u0558[]\xb1+\xbe\xf8\x008\n\xbc*\x06w\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xac~\x03p'#\xcb\x16\xee'\xe2-\u0438\x15\xdc-
\\\xae\x9f\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07d4\xac\x8bP\x9a\xef\xea\x1d\xbf\xaf+\xb35\x00\xd6W\vo\xd9mQ\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xac\x8e\x87\xdd\xda^x\xfc\xbc\xb9\xfa\u007f\xc3\xce\x03\x8f\x9f}.4\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xac\x9f\xffh\xc6\x1b\x01\x1e\xfb\xec\xf08\xedr\u06d7\xbb\x9er\x81\x8a\x02\x05\xb4\u07e1\xeetx\x00\x00\u07d4\xac\xa1\xe6\xbcd\xcc1\x80\xf6
\xe9M\u0171\xbc\xfd\x81X\xe4]\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xac\xa2\xa883\v\x170-
\xa71\xd3\r\xb4\x8a\x04\xf0\xf2\a\xc1\x89Hz\x9a0E9D\x00\x00\u07d4\xac\xaa\xdd\xcb\xf2\x86\xcb\x0e!]\xdaUY\x8f\u007f\xf0\xf4\xad\xa5\u018965\u026d\xc5\u07a0\x00\x00\u07d4\xac\xb9C8UK\u0108\u0308\xae-\x9d\x94\b\rk\u07c4\x10\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xac\xbc-
\x19\xe0l;\xab\xbb[o\x05+k\xf7\xfc7\xe0r)\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xac\xbd\x18U\x89\xf7\xa6\x8ag\xaaK\x1b\xd6Pw\xf8\xc6NN!\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xac\xc0bp,Ya]4D\xefb\x14\xb8\x86+\x00\x9a\x02\xed\x89QO\xcb$\xff\x9cP\x00\x0

0\u07d4\xac\xc0\x90\x9f\xda.\xa6\xb7\xb7\xa8\x8d\xb7\xa0\xaa\xc8h\t\x1d\xdb\xf6\x89\x013v_\x1e&\u01c0\x00\u07d4\xac\xc1\u01c7\x86\xabM+;'q5\xb5\xba\x12>\x04\x00Hk\x89\x04E\x91\xd6\u007f\xec\xc8\x00\x00\u07d4\xac\xc4j*U\\t\xde\u0522\xbd\tN\x82\x1b\x97\x84;@\xc0\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xac\u015f;0\xce\xff\xc5da\xcc[\x8d\xf4\x89\x02$\x0e\x0e{\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xac\xce\x01\xe0\xa7\x06\x10\xdcp\xbb\x91\xe9\x92o\xa9\x95\u007f7/\xba\x89\x1d\x1c_>\xda
\xc4\x00\x00\u07d4\xac\xd8\u0751\xf7\x14vLEg|c\xd8R\xe5n\xb9\xee\xce.\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xac\u2af6;\x06\x04@\x9f\xbd\xe3\xe7\x16\u0487mD\xe8\xe5\u0749\b=lz\xabc`\x00\x00\xe0\x94\xac\xec\x91\xefiA\xcfc\v\xa9\xa3\u71e0\x12\xf4\xa2\xd9\x1d\u050a\x10\xf0\xcf\x06M\u0552\x00\x00\x00\u07d4\xad\nJ\xe4x\xe9cn\x88\xc6\x04\xf2B\xcfT9\xc6\xd4V9\x89\xbe\xd1\xd0&=\x9f\x00\x00\x00\u07d4\xad\x17\x99\xaa\xd7`+E@\u0343/\x9d\xb5\xf1\x11P\xf1hz\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xad\x1dh\xa08\xfd%\x86\x06~\xf6\xd15\xd9b\x8ey\xc2\xc9$\x89\xfe\t\xa5'\x9e*\xbc\x00\x00\u07d4\xad*\\\x00\xf9#\xaa\xf2\x1a\xb9\xf3\xfb\x06n\xfa\n\x03\xde/\xb2\x8965\xbbw\xcbK\x86\x00\x00\u07d4\xad5e\xd5+h\x8a\xdd\xed\b\x16\x8b-8r\xd1}\n&\xae\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xad7|\xd2^\xb5>\x83\xae\t\x1a\n\x1d+E\x16\xf4\x84\xaf\u0789i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94\xadAM)\xcb~\xe9s\xfe\xc5N\"\xa3\x88I\x17\x86\xcfT\x02\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4\xadD5~\x01~$OGi1\u01f8\x18\x9e\xfe\xe8\n]\n\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xadW\xaa\x9d\x00\xd1\fC\x9b5\xef\xcc\v\xec\xac.9U\xc3\x13\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xadY\xa7\x8e\xb9\xa7J\u007f\xbd\xae\xfa\xfa\x82\xea\u0684u\xf0\u007f\x95\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xadZ\x8d<dx\xb6\x9fe}\xb3\x83z%u\xef\x8e\x1d\xf91\x89\x02\x01V\xe1\x04\xc1\xb3\x00\x00\u07d4\xadf\r\xec\x82U\"\xa9\xf6/\xce\xc3\u01771\x98\r\u0086\ua262\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xadf(5.\xd39\v\xaf\xa8m\x92>V\x01L\xfc\xb3`\xf4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xadr\x81!\x87?\x04V\xd0Q\x8b\x80\xabe\x80\xa2\x03pe\x95\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xads,\x97e\x93\xee\xc4x;N.\xcdy9yx\v\xfe\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xad}\xd0S\x85\x9e\xdf\xf1\xcbo\x9d*\xcb\xedm\xd5\xe32Bo\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xad\x80\xd8e\xb8\\4\xd2\xe6IK.z\xef\xeak\x9a\xf1\x84\u06c9\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xad\x8b\xfe\xf8\u018aH\x16\xb3\x91o5\xcb{\xfc\xd7\xd3\x04\tv\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xad\x8eH\xa3wi]\xe0\x146:R:(\xb1\xa4\fx\xf2\b\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xad\x91\n#\u0585\x06\x13eJ\xf7\x863z\u04a7\bh\xacm\x89lh\xcc\u041b\x02,\x00\x00\u07d4\xad\x92~\x03\xd1Y\x9ax\xca+\xf0\xca\u04a1\x83\xdc\xebq\xea\xc0\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94\xad\x92\xca\x06n\xdb|q\x1d\xfc[\x16a\x92\xd1\xed\xf8\xe7q\x85\x8a\a\x9f\x90\\o\xd3N\x80\x00\x00\u07d4\xad\x94#_\u00f3\xf4z$\x13\xaf1\u8111I\b\xef\fE\x89\x1b\x1b\x01B\xd8\x15\x84\x00\x00\u07d4\xad\x9e\x97\xa0H/5:\x05\xc0\xf7\x92\xb9w\xb6\xc7\xe8\x11\xfa_\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xad\x9fL\x89\n;Q\x1c\xeeQ\xdf\xe6\xcf\xd7\xf1\t;vA,\x89\x1bv|\xbf\xeb\f\xe4\x00\x00\u07d4\xad\xaa\x0eT\x8c\x03Z\xff\xedd\xcag\x8a\x96?\xab\xe9\xa2k\xfd\x89\x03\xcbq\xf5\x1f\xc5X\x00\x00\u07d4\xad\xb9H\xb1\xb6\xfe\xfe }\xe6^\x9b\xbc-\xe9\x8e`]\vW\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xad\xc1\x9e\xc85\xaf\xe3\u5347\u0713\xa8\xa9!<\x90E\x13&\x89j\xdb\xe54\"\x82\x00\x00\x00\u07d4\xad\xc8\"\x8e\xf9(\xe1\x8b*\x80}\x00\xfb<Iy\xcd\x1d\x9e\x96\x89\x01<i\xdf3N\xe8\x00\x00\u07d4\xad\xdb&1r'\xf4\\\x87\xa2\u02d0\xdcL\xfd\x02\xfb#\xca\xf8\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xad\xe6\xf8\x16;\xf7\u01fbJ\xbe\x8e\x98\x

93\xbd\f\xc1\x12\xfe\x88r\x89\x11\xc2]\x00M\x01\xf8\x00\x00\u07d4\xad\xeb
J\xa0\u00ce\x17\x9e\x81\xa9N\u0633\xe7\xd50G\xc2k\x89
\xf5\xb1\uab4d\x80\x00\x00\u07d4\xad\xebR\xb6\x04\xe5\xf7\u007f\xaa\xac\x88'[\x8dkI\xe9\xf9\xf
9\u007f\x89qBk\x00\x95n\xd2\x00\x00\xe0\x94\xad\xf1\xac\xfe\x99\xbc\x8c\x14\xb3\x04\xc8\xd9\x
05\xba'e{\x8a{\u010a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xad\xf8R\x03\xc87j_\u0798\x158J5\f8y
\xc4\u02d3\x89>1\xfcgX\x15\xaa\x00\x00\u07d4\xad\xff\r\x1d\v\x97G\x1ev\u05c9\xd2\u470at\xf9\x
bdT\xff\x89e\xea=\xb7UF`\x00\x00\u07d4\xae\x06,D\x86\x18d0u\xdez\x0004-
\xce\xd6=\xba\u05c9,\xc6\u034c\u0082\xb3\x00\x00\xe0\x94\xae\x10\xe2z\x01O\r0k\xaf&mH\x97\
u021a\xee\xe2\xe9t\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xae\x12k8,\xf2W\xfa\xd7\xf0\xbc}\x1
6)~T\xccrg\u0689\x10CV\x1a\x88)0\x00\x00\u07d4\xae\x13\xa0\x85\x11\x11\x0f2\xe5;\xe4\x12xE\
xc8C\xa1\xa5|{\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xae\x17\x9aF\r\xb6c&t=$\xe6u#\xa
5{$m\xaf\u007f\x8a\x01\x00\a\xae|\xe5\xbb\xe4\x00\x00\u07d4\xae\"(ey\x90y\xaa\xf4\xf0gJ\f\u06a
b\x02\xa6\xd5p\xff\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xae#\x9a\xcf\xfdN\xbe.\x1b\xa5\xb4\x17\
x05r\xdcy\xcce3\xec\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xae/\x9c\x19\xacv\x13e\x94C#\x93
\xb0G\x1d\b\x90!d\u04c9%\xdf\x05\u01a8\x97\xe4\x00\x00\u07d4\xae4\x86\x1d4\"S\x19O\xfcfR\x
df\xdeQ\xabD\xca\xd3\xfe\x89\x19F\bhc\x16\xbd\x80\x00\u07d4\xae6\xf7E!!\x91>\x80\x0e\x0f\xcd
\x1ae\xa5G\x1c#\x84o\x89\b\xe3\xf5\v\x17<\x10\x00\x00\u07d4\xae?\x98\xa4C\xef\xe0\x0f>q\x1d
R]\x98\x94\u071aa\x15{\x89\x10\x04\xe2\xe4_\xb7\xee\x00\x00\xe0\x94\xaeG\xe2`\x9c\xfa\xfe6\x
9df\xd4\x15\xd99\xde\x05\b\x1a\x98r\x8a\x05\xba\xec\xf0%\xf9\xb6P\x00\x00\u07d4\xaeO\x12.5\x
c0\xb1\xd1\xe4\x06\x92\x91E|\x83\xc0\u007f\x96_\xa3\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xa
ePU\x81L\xb8\xbe\f\x11{\xb8\xb1\xc8\u04b6;F\x98\xb7(\x89\x01\xbc\x93.\xc5s\xa3\x80\x00\u07d4
\xaeS\x8cs\u0173\x8d\x8dXM~\xbd\xad\xef\xb1\\\xab\xe4\x83W\x896'\xe8\xf7\x127<\x00\x00\u07
d4\xaeW\xcc\x12\x9a\x96\xa8\x99\x81\xda\xc6\r/\xfb\x87}]\xc5\xe42\x89<:#\x94\xb3\x96U\x00\x00
\u07d4\xaeZ\xa1\xe6\u00b6\x0fo\xd3\xef\xe7!\xbbJq\x9c\xbe=o]\x89+$\u01b5Z^b\x00\x00\u07d4\
xae\\\x9b\xda\xd3\xc5\u0221\"\x04D\xae\xa5\xc2)\xc1\x83\x9f\x1dd\x89\x19\xe2\xa4\xc8\x18\xb9\
x06\x00\x00\u07d4\xae\\\xe35Z{\xa9\xb32v\f\tP\u00bcE\xa8_\xa9\xa0\x89\x15\xaf\x1dx\xb5\x8c@
\x00\x00\xe0\x94\xae]\"\x1a\xfc\xd3\u0493U\xf5\b\xea\xdf\xca@\x8c\xe3<\xa9\x03\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xaec[\xf781\x11\x9d-
)\xc0\xd0O\xf8\xf8\xd8\u0425zF\x89Hz\x9a0E9D\x00\x00\xe0\x94\xaed\x81U\xa6X7\x0f\x92\x9b\x
e3\x84\xf7\xe0\x01\x04~I\xddF\x8a\x02\xdf$\xae2\xbe
D\x00\x00\xe0\x94\xaeo\fs\xfd\xd7|H\x97'Q!t\u0675\x02\x96a\x1cL\x8a\x01EB\xba\x12\xa37\xc0\x
00\x00\u07d4\xaep\xe6\x9d,J\n\xf8\x18\x80{\x1a'\x05\xf7\x9f\u0435\xdb\u01095e\x9e\xf9?\x0f\xc4
\x00\x00\u07d4\xaew9\x12N\xd1S\x05%\x03\xfc\x10\x14\x10\xd1\xff\xd8\xcd\x13\xb7\x8964\xfb\x
9f\x14\x89\xa7\x00\x00\u07d4\xaex\xbb\x84\x919\xa6\xba8\xae\x92\xa0\x9ai`\x1c\xc4\xcbb\u0449
\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xae\x84\"\x10\xf4M\x14\u0124\u06d1\xfc\x9d;;P\x01O{\
xf7\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xae\x84.\x81\x85\x8e\xcf\xed\xf6Plhm\xc2\x04\xac\
x15\xbf\x8b$\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xae\x89T\xf8\xd6\x16m\xe5\a\xcfa)}\x0f\xc
7\xcak\x9eq(\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xae\x9e\xcdk\u0755.\xf4\x97\xc0\x05\n\u0aca\
x82\xa9\x18\x98\u0389\x01\xa0Ui\r\x9d\xb8\x00\x00\u07d4\xae\x9f\\?\xbb\xe0\u027c\xbf\x1a\xf8\x
fft\xea(\v:]\x8b\b\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4\xae\xad\x88\u0589Ak\x1c\x91\xf26D!7[}
<p\xfb.\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xae\xad\xfc\u0417\x8e\xda\xd7J2\xbd\x01\xa0\xa5\x1
d7\xf2F\xe6a\x89\x0e\x189\x8ev\x01\x90\x00\x00\u07d4\xae\xb9\x16\xeb\xf4\x9d\x0f\x86\xc1?s1\

xce\xf1\x9e\x12\x997Q-\x89 \x85e[\x8d\x1b\n\x00\x00\u07d4\xae\xbdO
]\u7676K5d\xb2V\xd4*q\x1d7\uf649?\u03cbEt\xf8N\x00\x00\xe0\x94\xae\xc2|\xe2\x13>\x82\xd0R
R\n\xfb\\Wm\x9f~\xb9>\u048a\r\xd0A
\xba\t\xcf\xe6\x00\x00\u07d4\xae\xc2\u007f\xf5\xd7\xf9\xdd\u0691\x18?F\xf9\xd5%C\xb6\xcd+/\x8
9\x18e\x01'\xcc=\xc8\x00\x00\u07d4\xae\xe4\x9dh\xad\xed\xb0\x81\xfdCpZ_x\xc7x\xfb\x90\xdeH\
x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xae\xf5\xb1\"X\xa1\x8d\xec\a\xd5\xec.1et\x91\x9dy\xd
6\u0589lk\x93[\x8b\xbd@\x00\x00\u07d4\xae\xfc\xfe\x88\xc8&\xcc\xf11\xd5N\xb4\ua7b8\x0ea\xe1
\xee%\x89\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\xaf\x06\xf5\xfam\x12\x14\xecC\x96}\x1b\xd4\xdd\x
e7J\xb8\x14\xa98\x89\x04\xc5>\xcd\xc1\x8a`\x00\x00\u07d4\xaf\x11H\xefl\x8e\x10=u0\xef\xc9\x1
6y\u026c'\x00\t\x93\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xaf
>\"\x9d~mA\x9d\xf47\x8e\xa9\x87\x15Q_c\x14\x85\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94\xaf
X\xc7(,\xf6|\x8c<\xf90\x13<\x89a|\xe7])\x8a\x01w\"J\xa8D\xc7 \x00\x00\u07d4\xaf&\xf7\u01bfE>
x\xf0\x89S\u4c80\x04\xa2\xc1\xe2\t\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xaf0\x87\xe6.\x04\xbf\x90\rZT\xdc>\x94bt\u0692B;\x89\x01\x15\x8eF\t\x13\x
d0\x00\x00\u07d4\xaf6\x14\u0736\x8a6\xe4ZN\x91\x1ebybG\"-
Y[\x89z\x81\x06_\x11\x03\xbc\x00\x00\u07d4\xaf6\x15\u01c9\u0431\x15*\xd4\xdb%\xfe]\xcf\"(\x04
\xcfb\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xaf<\xb5\x96Y3\xe7\xda\u0603i;\x9c>\x15\xbe\xb6\x
8aHs\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xafD\x93\xe8R\x1c\xa8\x9d\x95\xf5&|\x1a\xb6?\x9fEA\
x1e\x1b\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xafL\xf4\x17\x85\x16\x1fW\x1d\f\xa6\x9c\x94\xf8\x02\x1fA)N\u028a\x02\x15\xf85
\xbcv\x9d\xa8\x00\x00\u07d4\xafR\x9b\xdbE\x9c\xc1\x85\xbe\xe5\xa1\u014b\xf7\xe8\xcc\xe2\\\x1
5\r\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xafg\xfd>\x12\u007f\xd9\xdc6\xeb?\xcdj\x80\u01feO
u2\xb2\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4\xafw\x1094Z40\x01\xbc\x0f\x8aY#\xb1&\xb6\rP\
x9c\x895e\x9e\xf9?\x0f\xc4\x00\x00\xe0\x94\xaf\u007fy\xcbAZ\x1f\xb8\u06fd\tF\a\xee\x8dA\xfb|Z;\
x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xaf\x87\xd27\x1e\xf3x\x95\u007f\xbd\x05\xba/\x
1df\x93\x1b\x01\u2e09%\xf2s\x93=\xb5p\x00\x00\u07d4\xaf\x88\x0f\xc7V}U\x95\xca\xcc\xe1\\?\xc
1L\x87B\xc2l\x9e\x89\a?u\u0460\x85\xba\x00\x00\u07d4\xaf\x8e\x1d\xcb1L\x95\r6\x87CM0\x98X\
xe1\xa8s\x9c\u0509\x0e~\xeb\xa3A\vt\x00\x00\u07d4\xaf\x99-
\xd6i\xc0\x88>U\x15\xd3\xf3\x11*\x13\xf6\x17\xa4\xc3g\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xaf\x
a1\u056d8\xfe\xd4GY\xc0[\x89\x93\xc1\xaa\r\xac\xe1\x9f@\x89\x04V9\x18$O@\x00\x00\xe0\x94\
xaf\xa59XnG\x19\x17J;F\xb9\xb3\xe6c\xa7\u0475\xb9\x87\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xaf\xa6\x94n\xff\xd5\xffS\x15O\x82\x01\x02S\xdfG\xae(\f\u0309j\xcb=\xf2~\x1f\x88
\x00\x00\u07d4\xaf\xc8\xeb\u860b\xd4\x10Z\xccL\x01\x8eTj\x1e\x8f\x9cx\x88\x89\x1b\x1a\xe4\xd
6\xe2\xefP\x00\x00\xe0\x94\xaf\xcc}\xbb\x83V\xd8B\xd4:\xe7\xe2<\x84\"\xb0\"\xa3\b\x03\x8a\x06o
\xfc\xbf\xd5\xe5\xa3\x00\x00\x00\u07d4\xaf\xd0\x19\xff6\xa0\x91U4ki\x97H\x15\xa1\xc9\x12\xc9\n
\xa4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xaf\xda\xc5\xc1\xcbV\xe2E\xbfp3\x00f\xa8\x17\uabecL\
u0449\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xaf\xdd\x1bxab\xb81~
\xf0\xe9y\xf4\xb2\xceHmv]\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xaf\xf1\x04Z\xdf'\xa1\xaa2\
x94a\xb2M\xe1\xba\u950ai\x8b\x89\x01\u03c4\xa3\n\n\f\x00\x00\u07d4\xaf\xf1\a\x96\v~\xc3N\u05
90\xb6e\x02M`\x83\x8c\x19\x0fp\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xaf\xf1\x1c\xcfi\x93
\x04\xd5\xf5\x86*\xf8`\x83E\x1c&\xe7\x9a\xe5\x89l]\xb2\xa4\xd8\x15\xdc\x00\x00\u07d4\xaf\xf1at\
nm\x90\x9f\xe9\x9cY\xa9\xb7yE\xc9\x1c\xc9\x14H\x89\x03@\xaa\xd2\x1b;p\x00\x00\xe0\x94\xaf\

xfc\x99\xd5\ubd28O\xe7x\x8d\x97\xdc\xe2t\xb08$\x048\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xaf\xfe\xa0G7\"\xcb\u007f\x0e\x0e\x86\xb9\xe1\x18\x83\xbfB\x8d\x8dT\x89i*\xe8\
x89p\x81\xd0\x00\x00\xe0\x94\xb0\t\x96\xb0Vn\xcb>rC\xb8\"y\x88\u0733R\xc2\x18\x99\x8a\x02\x
8a\x85t%Fo\x80\x00\x00\u07d4\xb0\x1e8\x9b(\xa3\x1d\x8eI\x95\xbd\xd7\xd7\xc8\x1b\xee\xab\x1e
A\x19\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb0-
\x06(s3EE\u03a2\x92\x18\xe4\x05w`Y\x0ft#\x89\xac\xb6\xa1\xc7\xd9:\x88\x00\x00\u07d4\xb0/\xa
2\x93\x87\xec\x12\xe3\u007fi\"\xacL\xe9\x8c[\t\xe0\xb0\x0f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\x
b06\x91k\xda\u03d4\xb6\x9eZ\x8ae`)u\xeb\x02a\x04\u0749\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x
94\xb0A1\x0f\xe9\xee\u0586L\xed\u053e\xe5\x8d\xf8\x8e\xb4\xed<\xac\x8a\x02\x1e\x19\xe0\u02
7a\xb2@\x00\x00\u07d4\xb0U\xafL\xad\xfc\xfd\xb4%\xcfe\xbad1\a\x8f\a\xec\u056b\x89\x05k\xc7^
-
c\x10\x00\x00\xe0\x94\xb0W\x11S\xdb\x1cN\u05ec\xae\xfe\x13\xec\xdf\xdbr\xe7\xe4\xf0j\x8a\x11\
f\xffyj\xc1\x95 \x00\x00\u07d4\xb0n\xab\t\xa6\x10\u01a5=V\xa9F\xb2\xc44\x87\xac\x1d[-
\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb0rI\xe0U\x04J\x91U5\x9a@)7\xbb\xd9T\xfeH\xb6\x89\x
05k\xc7^-
c\x10\x00\x00\xe0\x94\xb0v\x182\x8a\x90\x13\a\xa1\xb7\xa0\xd0X\xfc\xd5xn\x9er\xfe\x8a\x06gI]J
C0\xce\x00\x00\u07d4\xb0y\xbbM\x98f\x14:m\xa7*\xe7\xac\x00\"\x06)\x811\\\x89)3\x1eeX\xf0\xe0
\x00\x00\u07d4\xb0{\xcc\bZ\xb3\xf7)\xf2D\x00Ah7\xb6\x996\xba\x88s\x89lm\x84\xbc\xcd\xd9\xce\
x00\x00\u07d4\xb0{\xcf\x1c\xc5\xd4F.Q$\xc9e\xec\xf0\xd7\r\xc2z\xcau\x89V\xbcu\xe2\xd61\x00\x
00\x00\u07d4\xb0|\xb9\xc1$\x05\xb7\x11\x80uC\u0113De\xf8\u007f\x98\xbd-
\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xb0\u007f\u07af\xf9\x1dD`\xfel\xd0\u8870\xbd\x8d\"\xa6.\x
87\x8a\x01\x1d%)\xf3SZ\xb0\x00\x00\xe0\x94\xb0\x9f\xe6\xd44\x9b\x99\xbc7\x93\x80T\x02-
T\xfc\xa3f\xf7\xaf\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\xe0\x94\xb0\xaa\x00\x95\f\x0e\x81\xfa2\x
10\x17>r\x9a\xaf\x16:'\xcdq\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xb0\xacN\xfff\x80\xee\x14\x
16\x9c\xda\xdb\xff\xdb0\x80Om%\xf5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb0\xb3j\xf9\xae\xee\u
07d7\xb6\xb0\"\x80\xf1\x14\xf19\x84\xea2`\x895e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\xb0\xb7y\xb9K\
xfa<.\x1fX{\u031c~!x\x92\"7\x8f\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\xb0\xba\xeb0\xe3\x13wl
Lm$t\x02\xbaAg\xaf\u0361\u0309j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xb0\xbb)\xa8a\xea\x1dBME\x
ac\u053f\u0112\xfb\x8e\xd8\t\xb7\x89\x04V9\x18$O@\x00\x00\xe0\x94\xb0\xc1\xb1w\xa2
\xe4\x1f|t\xd0|\u0785i\xc2\x1cu\xc2\xf9\x8a\x01/\x93\x9c\x99\xed\xab\x80\x00\x00\u07d4\xb0\xc7\
xceL\r\xc3\u00bb\xb9\x9c\xc1\x85{\x8aE_a\x17\x11\u0389\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4
\xb0\xce\xf8\xe8\xfb\x89\x84\xa6\x01\x9f\x01\xc6y\xf2r\xbb\xe6\x8f\\w\x89\b=lz\xabc`\x00\x00\xe0\
x94\xb0\xd3+\xd7\xe4\u6577\xb0\x1a\xa3\xd0Ao\x80U}\xba\x99\x03\x8a\x03s\x9f\xf0\xf6\xe6\x13
0\x00\x00\xe0\x94\xb0\xd3\u0247+\x85\x05n\xa0\xc0\xe6\xd1\xec\xf7\xa7~<\u6ac5\x8a\x01\x0f\b\
xed\xa8\xe5U\t\x80\x00\u07d4\xb0\xe4i\u0206Y8\x15\xb3IV8Y]\xae\xf0f_\xaeb\x89i*\xe8\x89p\x81
\xd0\x00\x00\u07d4\xb0\xe7`\xbb\a\xc0\x81wsE\xe0W\x8e\x8b\u0218\"mN;\x89lk\x93[\x8b\xbd@\
x00\x00\u07d4\xb1\x040\x04\xec\x19A\xa8\xcfO+\x00\xb1W\x00\u076co\xf1~\x8965\u026d\xc5\u
07a0\x00\x00\u07d4\xb1\x05\xdd=\x98|\xff\xd8\x13\xe9\xc8P\n\x80\xa1\xad%}V\u0189lj\xccg\u05f
1\xd4\x00\x00\u07d4\xb1\x0f\u04a6G\x10/\x88\x1ft\xc9\xfb\xc3}\xa62\x94\x9f#u\x89\x02+\x1c\x8c\
x12'\xa0\x00\x00\xe0\x94\xb1\x15\xee:\xb7d\x1e\x1a\xa6\xd0\x00\xe4\x1b\xfc\x1e\xc7!\f/2\x8a\x0
2\xc0\xbb=\xd3\fN
\x00\x00\u07d4\xb1\x17\x8a\xd4s\x83\xc3\x1c\x814\xa1\x94\x1c\xbc\xd4t\xd0bD\xe2\x8965\u026

d\xc5\u07a0\x00\x00\xe0\x94\xb1\x17\x95\x89\u1779\xd4\x15W\xbb\xec\x1c\xb2L\xcc-
\xec\x1c\u007f\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xb1\x19\u76a9\xb9\x16Re\x81\xcb\xf5!\xefGJ\xe8M\xcf\xf4\x89O\xba\x10\x01\xe5\xbe\xfe\x00\x00\u07d4\xb1\x1f\xa7\xfb'\n\xbc\xdfZ.\xab\x95\xaa0\u013566\uffc9+^:\xf1k\x18\x80\x00\x00\u07d4\xb1$\xbc\xb6\xff\xa40\xfc\xae.\x86\xb4_'\xe3\xf2\x1e\x81\xee\b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb1)\xa5\xcbq\x05\xfe\x81\v\u0615\xdcr\x06\xa9\x91\xa4TT\x88\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\xe0\x94\xb1.\xd0{\x8a8\xadU\x066?\xc0z\vmy\x996\xbd\xaf\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xb14\xc0\x049\x1a\xb4\x99(x3zQ\xec$/B(WB\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb1?\x93\xaf0\xe8\xd7fs\x81\xb2\xb9[\xc1\xa6\x99\xd5\xe3\xe1)\x89\x16\u012b\xbe\xbe\xa0\x10\x00\x00\u07d4\xb1E\x92\x85\x86>\xa2\xdb7Y\xe5F\u03b3\xfb7a\xf5\x90\x9c\x89<\xd7*\x89@\x87\xe0\x80\x00\u07d4\xb1F\xa0\xb9%U<\xf0o\xca\xf5J\x1bM\xfe\xa6!)\aW\x89lnY\xe6|xT\x00\x00\xe0\x94\xb1Jz\xaa\x8fl\xf2\xfb\x9a\x81\x02\u05bb\xe4\u010a\xe7\xc0o\xb2\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xb1K\xbe\xffpr\tu\xdca\x91\xb2\xa4O\xf4\x9f&r\x87<\x89\a\xc0\x86\x0eZ\x80\xdc\x00\x00\xe0\x94\xb1L\xc8\xde3\xd63\x826S\x9aH\x90\xceFU\xa3+\u018a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xb1M\xdb\x03\x86\xfb`c\x98\xb8\xccGVZ\xfa\xe0\x0f\xf1\xd6j\x89\xa1*\xff\b>f\xf0\x00\x00\u07d4\xb1S\xf8(\xdd\amJ|\x1c%t\xbb-
\xee\x1aD\xa3\x18\xa8\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xb1T\x0e\x94\xcf\xf3F\\\xc3\u0447\xe7\xc8\u3f6f\x98FY\u2262\x15\xe4C\x90\xe33\x00\x00\u07d4\xb1X\xdbC\xfab\xd3\x0ee\xf3\u041b\xf7\x81\u01f6sr\uba89l]\xb2\xa4\xd8\x15\xdc\x00\x00\u07d4\xb1ar_\xdc\xed\xd1yR\xd5{#\xef([~K\x11i\xe8\x89\x02\xb6\xdf\xed6d\x95\x80\x00\u07d4\xb1dy\xba\x8e}\xf8\xf6>\x1b\x95\xd1I\u0345)\xd75\xc2\u0689-
\xe3:j\xac2T\x80\x00\u07d4\xb1f\xe3}.P\x1a\xe7<\x84\x14+_\xfbZ\xa6U\xddZ\x99\x89l]\xb2\xa4\xd8\x15\xdc\x00\x00\u07d4\xb1\x83\xeb\xeeO\xcbB\xc2
\xe4wt\xf5\x9dlT\xd5\xe3*\xb1\x89V\xf7\xa9\xc3<\x04\xd1\x00\x00\u07d4\xb1\x88\a\x84D\x02~8g\x98\xa8\xaehi\x89\x19\xd5\xcc#\r\x89\x0e~\xeb\xa3A\vt\x00\x00\u07d4\xb1\x89j7\xe5\u0602Z-
\x01vZ\xe5\xdeb\x99w\u0783R\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb1\x8eg\xa5\x05\n\x1d\xc9\xfb\x19\t\x19\xa3=\xa88\xefDP\x14\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb1\xa2\xb4:t3\xdd\x15\v\xb8\'"\xedQ\x9c\u05b1B\u04c2\x89\x94mb\rtK\x88\x00\x00\u07d4\xb1\xc0\u040b6\xe1\x84\xf9\x95*@7\xe3\xe5:f}\a\nN\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb1\xc3(\xfb\x98\xf2\xf1\x9a\xb6do\n|\x8cVo\xdaZ\x85@\x89\x87\x86x2n\xac\x90\x00\x00\xe0\x94\xb1\xc7Qxi9\xbb\xa0\xd6q\xa6w\xa1X\u01ab\xe7&^F\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\xb1\xcdK\xdf\xd1\x04H\x9a\x02n\u025dYs\a\xa0By\xf1s\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xb1\u03d4\xf8\t\x15\x05\x05_\x01\n\xb4\xba\u0196\xe0\xca\x0fg\xa1\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\xb1\u05b0\x1b\x94\xd8T\xfe\x8b7J\xa6^\x89\\\xf2*\xa2V\x0e\x892\xf5\x1e\u06ea\xa30\x00\x00\xe0\x94\xb1\u06e5%\v\xa9bWU$n\x06yg\xf2\xad/\a\x91\u078a\x10\xf0\xcf\x06M\u0552\x00\x00\x00\u07d4\xb1\xe2\u0755\xe3\x9a\xe9w\\U\xae\xb1?\x12\xc2\xfa#0S\xba\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb1\xe6\xe8\x10\xc2J\xb0H\x8d\xe9\xe0\x1eWH7\x82\x9f|w\u0409\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xb1\xe9\xc5\xf1\xd2\x1eauzk.\xe7Y\x13\xfcZ\x1aA\x01\u00c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xb2\x03\u049elV\xb9&\x99\u0139-
\x1fo\x84d\x8d\xc4\u03fc\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xb2\x16\xdcY\xe2|=ry\xf5\xcd[\xb2\xbe\u03f2`n\x14\u0649\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xb2\x1byy\xbf|\\\xa0\x1f\xa8
-

\xd6@\xb4\x1c9\xe6\u01bcu\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xb2#\xbf\x1f\xbf\x80H\\\xa2\xb5
V}\x98\xdb{\xc3SM\xd6i\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb2-
PU\xd9b15\x96\x1ej\xbd'<\x90\xde\xea\x16\xa3\xe7\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\xb2-
\xad\xd7\xe1\xe0R2\xa927\xba\xed\x98\xe0\u07d2\xb1\x86\x9e\x89lk\x93[\x8b\xbd@\x00\x00\u07
d4\xb24\x03_uDF<\xe1\xe2+\xc5S\x06F\x84\xc5\x13\xcdQ\x89\r\x89\xfa=\u010d\xcf\x00\x00\u07d
4\xb2G\u03dcr\xecH*\xf3\xea\xa7Ye\x8fy=g\nW\f\x891p\x8a\xe0\x04T@\x00\x00\u07d4\xb2ghA\x
ee\x9f-1\xc1r\xe8#\x03\xb0\xfe\x9b\xbf\x9f\x1e\t\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb2y\xc7\xd3U\u0088\x03\x92\xaa\u046a!\xee\x86|;5\a\u07c9D[\xe3\xf2\uf1d4\x0
0\x00\u07d4\xb2|\x1a$
L\x1e\x11\x8du\x14\x9d\xd1\t1\x1e\a\xc0s\xab\x89\xa8\r$g~\xfe\xf0\x00\x00\u07d4\xb2\x81\x81\x
a4X\xa4@\xf1\u01bb\x1d\xe8@\x02\x81\xa3\x14\x8fL5\x89\x14b\fW\xdd\xda\xe0\x00\x00\xe0\x94
\xb2\x82E\x03|\xb1\x92\xf7W\x85\u02c6\xcb\xfe|\x93\r\xa2X\xb0\x8a\x03c\\\x9a\xdc]\xea\x00\x00\
x00\u07d4\xb2\x87\xf7\xf8\xd8\u00c7,\x1bXk\xcd}\n\xed\xbf~s'2\x89\x01\x15\x8eF\t\x13\xd0\x00\x
00\u07d4\xb2\x8b\xb3\x9f4fQ|\xd4o\x97\x9c\xf5\x96S\xee}\x8f\x15.\x89\x18e\x01'\xcc=\xc8\x00\x0
0\u07d4\xb2\x8d\xbf\xc6I\x98\x94\xf7:q\xfa\xa0\n\xbe\x0fK\xc9\u045f*\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xb2\x96\x8f}5\xf2\b\x87\x161\xc6h{?=\xae\xab\xc6al\x89\bu\xc4\u007f(\x9fv\
x00\x00\u07d4\xb2\x9f[|\x190\xd9\xf9z\x11^\x06pf\xf0\xb5M\xb4K;\x8965\u026d\xc5\u07a0\x00\x0
0\u07d4\xb2\xa1D\xb1\xeag\xb9Q\x0f\"g\xf9\xda9\xd3\xf9=\xe2fB\x89lk\x93[\x8b\xbd@\x00\x00\u
07d4\xb2\xa2\xc2\x11\x16\x12\xfb\x8b\xbb\x8e}\xd97\x8dg\xf1\xa3\x84\xf0P\x89\x01\x15\x8eF\t\x
13\xd0\x00\x00\xe0\x94\xb2\xa4\x98\xf0;\xd7\x17\x8b\u0627\x89\xa0\x0fR7\xafy\xa3\xe3\xf8\x8a\
x04\x1b\xad\x15^e\x12
\x00\x00\u07d4\xb2\xaa/\x1f\x8e\x93\xe7\x97\x13\xd9,\xea\x9f\xfc\xe9\xa4\n\xf9\xc8-
\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb2\xb5\x16\xfd\u045e\u007f8d\xb6\xd2\xcf\x1b%*AV\xf1\xb
0;\x89\x02\xe9\x83\xc7a\x15\xfc\x00\x00\u07d4\xb2\xb7\u0374\xffKa\u0577\xce\v\"p\xbb\xb5&\x9
7C\xec\x04\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb2\xbd\xbe\u07d5\x90\x84v\xd7\x14\x8a7\f\u0193
t6(\x05\u007f\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb2\xbf\xaaX\xb5\x19l\\\xb7\xf8\x9d\x
e1_G\x9d\x188\xdeq=\x89\x01#n\xfc\xbc\xbb4\x00\x00\u07d4\xb2\xc5>\xfa3\xfeJ:\x1a\x80
\\s\xec;\x1d\xbc\xad\x06\x02\x89h\x01\u06b3Y\x18\x93\x80\x00\xe0\x94\xb2\xd06\x05\x15\xf1}\xa
b\xa9\x0f\u02ec\x82\x05\xd5i\xb9\x15\u05ac\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xb2\x
d1\xe9\x9a\xf9\x121\x85\x8epe\xdd\x19\x183\r\xc4\xc7G\u054a\x03\x89O\x0eo\x9b\x9fp\x00\x00\
u07d4\xb2\u066b\x96d\xbc\xf6\xdf <4o\u0192\xfd\x9c\xba\xb9
^\x89\x17\xbex\x97`e\x18\x00\x00\u07d4\xb2\u0777\x86\xd3yN'\x01\x87\xd0E\x1a\xd6\u0237\x9e
\x0e\x87E\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xb2\xe0\x85\xfd\xdd\x14h\xba\aA['NsN\x11
#\u007f\xb2\xa9\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xb2\xe9\xd7k\xf5\x0f\xc3k\xf7\u04d4Kc\xe9\u0288\x9bi\x99h\x89\x902\xeab\
xb7K\x10\x00\x00\xe0\x94\xb2\xf9\xc9r\xc1\xe9swU\xb3\xff\x1b0\x88s\x83\x969[&\x8a\x04<3\xc1\
x93ud\x80\x00\x00\xe0\x94\xb2\xfc\x84\xa3\xe5\nP\xaf\x02\xf9M\xa08>\u055fq\xff\x01\u05ca\x06
ZM\xa2]0\x16\xc0\x00\x00\u07d4\xb3\x05\v\xef\xf9\xde3\xc8\x0e\x1f\xa1R%\xe2\x8f,A:\xe3\x13\x
89%\xf2s\x93=\xb5p\x00\x00\u07d4\xb3\x11\x96qJH\xdf\xf7&\xea\x943\xcd)\x12\xf1\xa4\x14\xb3\
xb3\x89\x91Hx\xa8\xc0^\xe0\x00\x00\xe0\x94\xb3\x14[tPm\x1a\x8d\x04|\xdc\xdcU9*{SPy\x9a\x8a
\x1bb)t\x1c\r=]\x80\x00\u07d4\xb3 \x83H6\xd1\xdb\xfd\xa9\xe7\xa3\x18M\x1a\xd1\xfdC
\xcc\xc0\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb3#\u073f.\xdd\xc58.\u4efb

\x1c\xa3\x93\x1b\xe8\xb48\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xb3$\x00\xfd\x13\xc5P\t\x17\xcb\x03{)\xfe\"\xe7\xd5\"\x8f-
\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xb3%gL\x01\xe3\xf7)\rR&3\x9f\xbe\xacg\xd2!'\x9f\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\xb3(%\xd5\xf3\xdb$\x9e\xf4\xe8\\\xc4\xf31S\x95\x89v\u8f09\x1b-
\xf9\xd2\x19\xf5y\x80\x00\u07d4\xb3*\xf3\xd3\xe8\xd0u4I&To.2\x88{\xf9;\x16\xbd\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb3/\x1c&\x89\xa5\xcey\xf1\xbc\x97\v1XO\x1b\xcf\"\x83\xe7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb3<\x03#\xfb\xf9\xc2l\x1d\x8a\xc4N\xf7C\x91\u0400F\x96\u0689\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xb3O\x04\xb8\xdbe\xbb\xa9\xc2n\xfcL\xe6\xef\xc5\x04\x81\xf3\xd6]\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xb3U}9\xb5A\x1b\x84D__T\xf3\x8fb\xd2qM\x00\x87\x89
\x86\xac5\x10R`\x00\x00\xe0\x94\xb3X\xe9|p\xb6\x05\xb1\xd7\xd7)\u07f6@\xb4<^\xaf\xd1\xe7\x8a\x04<3\xc1\x93ud\x80\x00\x00\u0794\xb3^\x8a\x1c\r\xac~\x0ef\u06ecsjY*\xbdD\x01%a\x88\xcf\xceU\xaa\x12\xb3\x00\x00\xe0\x94\xb3fx\x94\xb7\x86<\x06\x8a\xd3D\x87?\xcf\xf4\xb5g\x1e\x06\x89\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xb3qw1\xda\xd6Q2\xday-
\x87`0\xe4j\xc2'\xbb\x8a\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb3s\x1b\x04l\x8a\u0195\xa1'\xfdy\u0425\xd5\xfaj\xe6\xd1.\x89lO\xd1\xee$nx\x00\x00\u07d4\xb3|+\x9fPc{\xec\xe0\u0295\x92\b\xae\xfe\xe6F;\xa7
\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xb3\x88\xb5\xdf\xec\xd2\xc5\u4d56W|d%V\xdb\xfe'xU\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb3\x8cNS{]\xf90\xd6Zt\xd0C\x83\x1dkH[\xbd\xe4\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\xb3\x919Wa\x94\xa0\x86a\x95\x15\x1f3\xf2\x14\n\xd1\u0306\u03ca\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xb3\x9fL\x00\xb2c\f\xab}\xb7)^\xf4=G\xd5\x01\xe1\u007f\u05c9\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb3\xa6K\x11vrOT\t\xe1AJ5#f\x1b\xae\xe7KJ\x89\x01ch\xffO\xf9\xc1\x00\x00\u07d4\xb3\xa6\xbdA\xf9\xd9\xc3
\x1e\x05\v\x87\x19\x8f\xbd\xa3\x994\"\x10\x89\xc4a\xe1\xdd\x10)\xb5\x80\x00\u07d4\xb3\xa8\xc2\xcb}5\x8eW9\x94\x1d\x94[\xa9\x04Z\x02:\x8b\xbb\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb3\xaeT\xfb\xa0\x9d>\xe1\u05bd\xd1\xe9W\x929\x19\x02L5\xfa\x89\x03\x8d,\xeee\xb2*\x80\x00\u07d4\xb3\xb7\xf4\x93\xb4J,\x8d\x80\xecx\xb1\xcd\xc7Ze+s\xb0l\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb3\xc2(s\x1d\x18m-
\xed[_\xbe\x00Lfl\x8eF\x9b\x86\x89\x01\x92t\xb2Y\xf6T\x00\x00\u07d4\xb3\xc2``\x9b\x9d\xf4\t^l]\xff9\x8e\xeb^-
\xf4\x99\x85\x89\r\xc5_\xdb\x17d{\x00\x00\u07d4\xb3\xc6[\x84Z\xbal\xd8\x16\xfb\xaa\xe9\x83\xe0\xe4l\x82\xaa\x86\"\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb3\xc9H\x11\xe7\x17[\x14\x8b(\x1c\x1a\x84[\xfc\x9b\xb6\xfb\xc1\x15\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb3\xe2\x0e\xb4\xde\x18\xbd\x06\x02!h\x98\x94\xbe\u5bb2SQ\xee\x89\x03\xfc\x80\xcc\xe5\x16Y\x80\x00\u07d4\xb3\xe3\xc49\x06\x98\x80\x15f\x00\u0089.D\x8dA6\xc9-
\x9b\x89.\x14\x1e\xa0\x81\xca\b\x00\x00\xe0\x94\xb3\xf8*\x87\xe5\x9a9\xd0\u0480\x8f\aQ\xebr\xc22\x9c\xdc\u014a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xb3\xfc\x1dh\x81\xab\xfc\xb8\xbe\xcc\v\xb0!\xb8\xb7;r3\u0751\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\xb4\x05\x94\xc4\xf3fN\xf8I\u0326\"{\x8a%\xaai\t%\xee\x89\xd8\xd7&

\xb7\x17z\x80\x00\x00\u07d4\xb4\x1e\xaf]Q\xa5\xba\x1b\xa3\x9b\xb4\x18\u06f5O\xabu\x0e\xfb\x1f\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb4$\u058d\x9d\r\x00\xce\xc1\x93\x8c\x85N\x15\xff\xb8\x80\xba\x01p\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb4%bs\x96+\xf61\xd0\x14U\\\xc1\xda\r\xcc1akI\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb40g\xfep\u0675Ys\xbaX\xdcd\xdd\u007f1\x1eUBY\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb46W\xa5\x0e\xec\xbc0w\xe0\x05\xd8\xf8\xd9O7xv\xba\u0509\x01\xec\x1b:\x1f\xf7Z\x00\x00\u07d4\xb4<'\xf7\xa0\xa1\"\bK\x98\xf4\x83\x92%A\u0203l\xee,\x89&\u009eG\u0104L\x00\x00\xe0\x94\xb4A5v\x86\x9c\b\xf9Q*\xd3\x11\xfe\x92Y\x88\xa5-
4\x14\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xb4F\x05U$q\xa6\xee\xe4\u06abq\xff;\xb4\x13&\xd4s\xe0\x89-
~=Q\xbaS\xd0\x00\x00\u07d4\xb4GW\x1d\xac\xbb>\u02f6\xd1\xcf\v\f\x8f88\xe5#$\xe2\x89\x01\xa3\x18f\u007f\xb4\x05\x80\x00\u07d4\xb4G\x83\xc8\xe5{H\a\x93\xcb\u059aE\xd9\f{O\fH\xac\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb4H\x15\xa0\xf2\x8eV\x9d\x0e\x92\x1aJ\u078f\xb2d%&Iz\x89\x03\x027\x9b\xf2\xca.\x00\x00\u07d4\xb4Im\xdb'y\x9a\"$W\xd79y\x11g(\u8844[\x89\x8d\x81\x9e\xa6_\xa6/\x80\x00\xe0\x94\xb4RL\x95\xa7\x86\x0e!\x84\x02\x96\xa6\x16$@\x19B\x1cJ\xba\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xb4\\\xca\r6\x82fbh<\xf7\u0432\xfd\xach\u007f\x02\xd0\u010965\u026d\xc5\u07a0\x00\x00\u0794\xb4d@\u01d7\xa5V\xe0L}\x91\x04f\x04\x91\xf9k\xb0v\xbf\x88\xce\xc7o\x0eqR\x00\x00\u07d4\xb4j\u0386^,P\xeaF\x98\xd2\x16\xabE]\xffZ\x11\xcdr\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb4m\x11\x82\xe5\xaa\xca\xff\r&\xb2\xfc\xf7/<\x9f\xfb\xcd\xd9}\x89\xaa*`<\xdd\u007f,\x00\x00\u07d4\xb4\x89!\xc9h}U\x10tE\x84\x93n\x88\x86\xbd\xbf-
\xf6\x9b\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb4\x98\xbb\x0fR\x00\x05\xb6!jD%\xb7Z\xa9\xad\xc5-
b+\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xb4\xb1\x1d\x10\x9f`\x8f\xa8\xed\xd3\xfe\xa9\xf8\xc3\x15d\x9a\xeb=\x11\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xb4\xb1K\xf4TU\u042b\b\x035\x8bu$\xa7+\xe1\xa2\x04[\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xb4\xb1\x85\xd9C\xee+Xc\x1e3\xdf\xf5\xafhT\xc1y\x93\xac\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb4\xbf$\u02c3hk\xc4i\x86\x9f\xef\xb0D\xb9\tqi\x93\xe2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb4\xc2\x00@\xcc\u0661\xa3(=\xa4\u0522\xf3e\x82\bC\xd7\xe2\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb4\xc8\x17\x0f{*\xb56\xd1\u0662[\xdd
:\xe1(\x8d\xc3\u0549\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb4\xd8/.i\x94?}\xe0\xf5\xf7t8y@o\xac.\x9c\xec\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\xe0\x94\xb4\xddF\f\xd0\x16rZd\xb2.\xa4\xf8\xe0n\x06gN\x03>\x8a\x01#\x1b\xb8t\x85G\xa8\x00\x00\u07d4\xb4\xddT\x99\xda\xeb%\a\xfb-
\xe1\"\x97s\x1dLr\xb1k\xb0\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xb5\x04l\xb3\xdc\x1d\xed\xbd6E\x14\xa2\x84\x8eD\xc1\xdeN\xd1G\x8a\x03{}\x9b\xb8
@^\x00\x00\xe0\x94\xb5\b\xf9\x87\xb2\xde4\xaeL\xf1\x93\u0785\xbf\xf6\x13\x89b\x1f\x88\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xb5\tU\xaan4\x15q\x98f\b\xbd\u0211\xc2\x13\x9fT\f\u07c9j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xb5\f\x14\x9a\x19\x06\xfa\xd2xo\xfb\x13Z\xabP\x177\xe9\xe5o\x89\x15\b\x94\xe8l\xb3\x90\x00\x00\u07d4\xb5\f\x9fW\x89\xaeD\xe2\xdc\xe0\x17\xc7\x14\xca\xf0\f\x83\x00\x84\u0089\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\xb5\x14\x88,\x97\x9b\xb6B\xa8\r\u04c7T\u0578\xc8)m\x9a\a\x893\xc5I\x901r\f\x00\x00\u07d4\xb5\x1d\u0734\xddN\x8a\xe6\xbe3m\xd9eIq\xd9\xfe\xc8kA\x89\x16\xd4d\xf8=\u2500\x00\u07d4\xb5\x1eU\x8e\xb5Q/\xbc\xfa\x81\xf8\u043d\

x93\x8cy\xeb\xb5$+\x89&\u009eG\u0104L\x00\x00\u07d4\xb5#\xff\xf9t\x98q\xb3S\x88C\x887\xf7\xe6\xe0\u07a9\xcbk\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xb5-\xfbE\xde]t\xe3\xdf
\x832\xbcW\x1c\x80\x9b\x8d\xcf2\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xb55\xf8\u06c7\x9f\xc6\u007f\xecX\x82J\\\xbenT\x98\xab\xa6\x92\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\xb57\xd3jp\xee\xb8\xd3\xe5\xc8\r\xe8\x15\"\\\x11X\u02d2\u0109QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4\xb5;\xcb\x17L%\x184\x8b\x81\x8a\xec\xe0
6E\x96Fk\xa3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb5I>\xf1srDE\xcf4\\\x03]'\x9b\xa7Y\xf2\x8dQ\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb5S\xd2]kT!\xe8\x1c*\xd0^\v\x8b\xa7Q\xf8\xf0\x10\xe3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb5Tt\xbaX\xf0\xf2\xf4\x0el\xba\xbe\xd4\xea\x17n\x01\x1f\xca\u0589j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xb5U\xd0\x0f\x91\x90\xcc6w\xae\xf3\x14\xac\xd7?\xdc99\x92Y\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb5W\xab\x949\xefP\xd27\xb5S\xf0%\b6JFj\\\x03\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xb5jx\x00(\x03\x9c\x81\xca\xf3{gu\xc6
\u7195Gd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb5j\u04ae\xc6\xc8\xc3\xf1\x9e\x15\x15\xbb\xb7\u0751(RV\xb69\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb5t\x13\x06\n\xf3\xf1N\xb4y\x06_\x1e\x9d\x19\xb3uz\xe8\u0309\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xb5uI\xbf\xbc\x9b\xdd\x18\xf76\xb2&P\xe4\x8as`\x1f\xa6\\\x89\x18-
~L\xfd\xa08\x00\x00\xe0\x94\xb5w\xb6\xbe\xfa\x05N\x9c\x04\x04a\x85P\x94\xb0\x02\xd7\xf5{\u05ca\x18#\xf3\xcfb\x1d#@\x00\x00\u07d4\xb5{\x04\xfa#\xd1
?\xae\x06\x1e\xacEB\xcb`\xf3\xa5v7\x89\nZ\xa8P\t\xe3\x9c\x00\x00\u07d4\xb5\x87\f\xe3B\xd43C36s\x03\x8bGd\xa4n\x92_>\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb5\x87\xb4J,\xa7\x9eK\xc1\u074b\xfd\xd4:
qP\xf2\xe7\xe0\x89\",\x8e\xb3\xfff@\x00\x00\u07d4\xb5\x89gm\x15\xa0DH4B0\xd4\xff'\xc9^\xdf\x12,I\x8965\u026d\xc5\u07a0\x00\x00\u0794\xb5\x8bR\x86^\xa5]\x806\xf2\xfa\xb2`\x98\xb3R\u0283~\x18\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xb5\x90k\n\u9881X\xe8\xacU\x0e9\xda\bn\xe3\x15v#\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb5\xa4g\x96\x85\xfa\x14\x19l.\x920\xc8\xc4\xe3;\xff\xbc\x10\xe2\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\xb5\xa5\x89\u075f@q\u06f6\xfb\xa8\x9b?]]\xae}\x96\xc1c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb5\xa6\x06\xf4\xdd\u02f9G\x1e\xc6\u007fe\x8c\xaf+\x00\xees\x02^\x89\xeaun\xa9*\xfct\x00\x00\u07d4\xb5\xadQW\u0769!\xe6\xba\xfa\u0350\x86\xaes\xae\x1fa\x1d?\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb5\xad\xd1\u701f}\x03\x06\x9b\xfe\x88;\n\x93\"\x10\xbe\x87\x12\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb5\xba)\x91|x\xa1\xd9\xe5\xc5\xc7\x13fl\x1eA\x1d\u007fi:\x89\xa8\r$g~\xfe\xf0\x00\x00\u07d4\xb5\xc8\x16\xa8(<\xa4\xdfh\xa1\xa7=c\xbd\x80&\x04\x88\xdf\b\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb5\xca\xc5\xed\x03G}9\v\xb2g\xd4\xeb\xd4a\x01\xfb\xc2\xc3\u0689\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xb5\u037cA\x15@oR\u5a85\xd0\xfe\xa1p\u0497\x9c\u01fa\x89Hz\x9a0E9D\x00\x00\u0794\xb5\u0653M{)+\xcf`;(\x80t\x1e\xb7`(\x83\x83\xa0\x88\xe7\xc2Q\x85\x05\x06\x00\x00\u07d4\xb5\xddP\xa1]\xa3Ih\x89\nS\xb4\xf1?\xe1\xaf\b\x1b\xaa\xaa\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb5\xfa\x81\x84\xe4>\xd3\u0e2b\x91!da\xb3R\x8d\x84\xfd\t\x89\x91Hx\xa8\xc0^\xe0\x00\x00\u07d4\xb5\xfb~\xa2\xdd\xc1Y\x8bfz\x9dW\xdd9\xe8Z8\xf3]V\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xb6\x00B\x97R\xf3\x99\xc8\r\a4tK\xae\n\x02.\xcag\u0189\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb6\x00\xfe\xabJ\xa9lSu\x04\xd9`W\"1Ai,\x19:\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xb6\x04|\u07d3-

\xb3\xe4\x04_Iv\x12#AS~\u0556\x1e\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb6\x15\xe9@\x14>\xb5\u007f\x87X\x93\xbc\x98\xa6\x1b=a\x8c\x1e\x8c\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xb6\x1c4\xfc\xac\xdap\x1aZ\xa8p$Y\u07b0\u4b83\x8d\xf8\x8a\aiZ\x92\xc2\ro\xe0\x00\x00\xe0\x94\xb60d\xbd3U\xe6\xe0~-
7p$\x12Z3wlJ\xfa\x8a\b7Z*\xbc\xca$@\x00\x00\u07d4\xb65\xa4\xbcq\xfb(\xfd\xd5\xd2\xc3\"\x98:V\u0084Bni\x89\t79SM(h\x00\x00\u07d4\xb6F\u07d8\xb4\x94BtkaR\\\x81\xa3\xb0K\xa3\x10bP\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xb6YA\xd4LP\xd2Ffg\r6Gf\xe9\x91\xc0.\x11\u0089
\x86\xac5\x10R`\x00\x00\xe0\x94\xb6[\u05c0\xc7CA\x15\x16
'VR#\xf4NT\x98\xff\x8c\x8a\x04<0\xfb\b\x84\xa9l\x00\x00\u07d4\xb6d\x11\xe3\xa0-\xed\xb7&\xfay\x10}\xc9\v\xc1\xca\xe6MH\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb6fu\x14.1\x11\xa1\xc2\xea\x1e\xb2A\x9c\xfaB\xaa\xf7\xa24\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xb6o\x92\x12K^c\x03XY\xe3\x90b\x88i\xdb\u07a9H^\x8a\x02\x15\xf85\xbcv\x9d\xa8\x00\x00\u07d4\xb6rsJ\xfc\xc2$\xe2\xe6\t\xfcQ\xd4\xf0Ys'D\xc9H\x89\x10\x04\xe2\xe4_\xb7\xee\x00\x00\xe0\x94\xb6w\x1b\v\xf3B\u007f\x9a\xe7\xa9>|.a\xeec\x94\x1f\xdb\b\x8a\x03\xfb&i)T\xbf\xc0\x00\x00\u07d4\xb6z\x80\xf1p\x19}\x96\xcd\xccJ\xb6\u02e6'\xb4\xaf\xa6\xe1,\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4\xb6\x88\x99\xe7a\rL\x93\xa255\xbc\xc4H\x94[\xa1fo\x1c\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb6\xa8)3\xc9\xea\u06bd\x98\x1e]m`\xa6\x81\x8f\xf8\x06\xe3k\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\xb6\xaa\u02cc\xb3\v\xab*\xe4\xa2BF&\xe6\xe1+\x02\xd0F\x05\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xb6\xb3J&?\x10\xc3\xd2\xec\xeb\n\xccU\x9a{*\xb8\\\xe5e\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb6\xbf\xe1\xc3\xef\x94\xe1\x84o\xb9\xe3\xac\xfe\x9bP\xc3\xe9\x06\x923\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xb6\xcdt2\xd5\x16\x1b\xe7\x97h\xadE\xde>Dz\a\x98
c\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb6\xceM\xc5`\xfcs\xdci\xfbzb\xe3\x88\xdb~r\xeavO\x894]\xf1i\xe9\xa3X\x00\x00\u07d4\xb6\xde\u03c2\x96\x98\x19\xba\x02\xde)\xb9\xb5\x93\xf2\x1bd\xee\xda\x0f\x89(\x1d\x90\x1fO\xdd\x10\x00\x00\xe0\x94\xb6\xe6\xc3\"+ko\x9b\xe2\x87]*\x89\xf1'\xfbd\x10\x0f\xe2\x8a\x01\xb2\x1dS#\xcc0
\x00\x00\u07d4\xb6\xe8\xaf\xd9=\xfa\x9a\xf2\u007f9\xb4\xdf\x06\ag\x10\xbe\xe3\u07eb\x89\x01Z\xf1\u05cbX\xc4\x00\x00\xe0\x94\xb6\xf7\x8d\xa4\xf4\xd0A\xb3\xbc\x14\xbc[\xa5\x19\xa5\xba\f2\xf1(\x8a$}\xd3,?\xe1\x95\x04\x80\x00\xe0\x94\xb6\xfb9xbP\b\x14&\xa3B\xc7\rG\xeeR\x1e[\xc5c\x8a\x03-
&\xd1.\x98\v`\x00\x00\u07d4\xb7\r\xba\x93\x91h+J6Nw\xfe\x99%c\x01\xa6\xc0\xbf\x1f\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb7\x16#\xf3Q\a\xcft1\xa8?\xb3\xd2\x04\xb2\x9e\u0c67\xf4\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4\xb7\x1a\x13\xba\x8e\x95\x16{\x803\x1bR\u059e7\x05O\xe7\xa8&\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb7\x1bb\xf4\xb4H\xc0+\x12\x01\xcb^9J\xe6'\xb0\xa5`\xee\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xb7\" \xad\xe3d\xd06\x9f--
\xa7\x83\xcaGM{\x9b4\u0389\x1b\x1a\xb3\x19\xf5\xecu\x00\x00\xe0\x94\xb7#\r\x1d\x1f\xf2\xac\xa3f\x969\x14\xa7\x9d\xf9\xf7\xc5\xea,\x98\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94\xb7$\n\U000af433<\b\xae\x97d\x10>5\xdc\xe3c\x84(\x8a\x01\xca\xdd/\xe9hnc\x80\x00\u07d4\xb7'\xa9\xfc\x82\xe1\xcf\xfc\\\x17_\xa1HZ\x9b\xef\xa2\u037d\u04496'\xe8\xf7\x127<\x00\x00\u07d4\xb7,*\x01\x1c\r\xf5\x0f\xbbn(\xb2\n\xe1\xaa\xd2\x17\x88g\x90\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb7

8-
7\xdb\x03\x98\xacrA\f\xf9\x81=\xe9\xf8\xe1\uc36d\x8966\xc2^f\xec\xe7\x00\x00\u07d4\xb7;O\xf9\x9e\xb8\x8f\u061b\vmW\xa9\xbc3\x8e\x88o\xa0j\x89\x01\xbc\x16\xd6t\xec\x80\x00\x00\u07d4\xb7=jwU\x9c\x86\xcfet$)\x039K\xac\xf9n5p\x89\x04\xf1\xa7|\xcd;\xa0\x00\x00\u07d4\xb7Cr\xdb\xfa\x18\x1d\xc9$/9\xbf\x1d71\xdf\xfe+\xda\u03c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xb7G\x9d\xabP\"\xc4\xd5\u06ea\xf8\xde\x17\x1bN\x95\x1d\u0464W\x89\x04V9\x18$O@\x00\x00\u07d4\xb7I\xb5N\x04\u0571\x9b\xdc\xed\xfb\x84\xdaw\x01\xabG\x8c'\xae\x89\x91Hx\xa8\xc0^\xe0\x00\x00\u07d4\xb7N\xd2f`\x01\xc1c3\xcfz\xf5\x9eJ=H`6;\x9c\x89\n~\xbd^Cc\xa0\x00\x00\u07d4\xb7QI\xe1\x85\xf6\xe3\x92pWs\x90s\xa1\x82*\xe1\xcf\r\xf2\x89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4\xb7S\xa7_\x9e\xd1\v!d:\n=\xc0Qz\xc9k\x1a@h\x89\x15\xc8\x18[,\x1f\xf4\x00\x00\xe0\x94\xb7V\xadR\xf3\xbft\xa7\xd2LgG\x1e\b\x87Ci6PL\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xb7Wn\x9d1M\xf4\x1e\xc5Pd\x94):\xfb\x1b\xd5\xd3\xf6]\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb7X\x89o\x1b\xaa\x86O\x17\xeb\xed\x16\xd9S\x88o\xeeh\xaa\xe6\x8965\u026d\xc5\u07a0\x00\x00\u0794\xb7h\xb5#N\xba:\x99h\xb3Mm\xdbH\x1c\x84\x19\xb3e]\x88\xcf\xceU\xaa\x12\xb3\x00\x00\u07d4\xb7\x82\xbf\xd1\xe2\xdep\xf4gdo\x9b\xc0\x9e\xa5\xb1\xfc\xf4P\xaf\x89\x0e~\xeb\xa3A\vt\x00\x00\xe0\x94\xb7\xa2\xc1\x03r\x8bs\x05\xb5\xaen\x96\x1c\x94\xee\x99\xc9\xfe\x8e+\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\xb7\xa3\x1a|8\xf3\xdb\t2.\xae\x11\xd2'!A\xea\"\x99\x02\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb7\xa6y\x1c\x16\xebN!b\xf1Ke7\xa0+=c\xbf\xc6\x02\x89*Rc\x91\xac\x93v\x00\x00\u07d4\xb7\xa7\xf7|4\x8f\x92\xa9\xf1\x10\fk\xd8)\xa8\xacm\u007f\u03d1\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xb7\xc0w\x94ft\xba\x93A\xfbLtz]P\xf5\xd2\xdad\x15\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xb7\xc0\xd0\xcc\vM4-
@b\xba\xc6$\xcc\xc3\xc7\f\xc6\xda?\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb7\xc9\xf1+\x03\x8esCm\x17\xe1\xc1/\xfe\x1a\xec\u0373\xf5\x8c\x89\x1dF\x01b\xf5\x16\xf0\x00\x00\u07d4\xb7\xcck\x1a\xcc2\u0632\x95\xdfh\xed\x9d^`\xb8\xf6L\xb6{\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xb7\xcehK\t\xab\xdaS8\x9a\x87Si\xf7\x19X\xae\xac;\u0749lk\x93[\x8b\xbd@\x00\x00\u07d4\xb7\xd1.\x84\xa2\xe4\u01264Z\xf1\xdd\x1d\xa9\xf2PJ*\x99n\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xb7\xd2R\xee\x94\x02\xb0\xee\xf1D)_\x0ei\xf0\xdbXl\bq\x89#\xc7W\a+\x8d\xd0\x00\x00\xe0\x94\xb7\u0541\xfe\n\xf1\xec8?;<Ag\x83\xf3\x85\x14jv\x12\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xb7\xf6s\x14\u02c3.2\xe6;\x15\xa4\f\xe0\xd7\xff\xbd\xb2i\x85\x899\x82y&J\x81\x8d\x00\x00\xe0\x94\xb8\x04\x056\x95\x8dY\x98\xceK\xec\f\xfc\x9c\"\x04\x98\x98H\xe9\x8a\x05.\xa7\rI\x8f\xd5\n\x00\x00\u07d4\xb81\n\x16\xccj\xbcFP\aiK\x93\x0f\x97\x8e\xce\x190\xbd\x89(\x1d\x90\x1fO\xdd\x10\x00\x00\xe0\x94\xb84\xac\xf3\x01S\"\u0143\x82\uecb7\x968\x90n\x88\xb6\u078a\x05\x15\n\xe8J\x8c\xdf\x00\x00\x00\xe0\x94\xb8KS\u043b\x12VV\xcd\xdcR\xeb\x85*\xb7\x1drY\xf3\u054a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07d4\xb8L\x8b\x9f\xd3>\xce\x00\xaf\x91\x99\xf3\xcf_\xe0\xcc\xe2\x8c\xd1J\x89\xcf\x15&@\xc5\xc80\x00\x00\u07d4\xb8R\x18\xf3B\xf8\x01.\u069f'Nc\xce!R\xb2\xdc\xfd\xab\x89\xa8\r$g~\xfe\xf0\x00\x00\u07d4\xb8UP\x10wn<\\\xb3\x11\xa5\xad\xee\xfe\x9e\x92\xbb\x9ad\xb9\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xb8_&\xdd\x0er\xd9\u009e\xba\xf6\x97\xa8\xafwG,+X\xb5\x8a\x02\x85\x19\xac\xc7\x19\fp\x00\x00\u07d4\xb8_\xf0>{_\xc4\"\x98\x1f\xae^\x99A\xda\xcb\u06bau\x84\x89Hz\x9a0E9D\x00\x00\xe0\x94\xb8f\a\x02\x1bb\xd3@\xcf&R\xf3\xf9_\xd2\xdcgi\x8b\u07ca\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xb8}\xe1\xbc\u0492i\xd5!\xb8v\x1c\u00dc\xfbC\x19\xd2\xea\u054965\u026d\xc5\u07a0\x00\x00\u07d4\xb8\u007fSv\xc2\xde\vl\xc3\xc1y\xc0`\x87\xaaG=kFt\x89Hz\x9a0E9D\x00\x0

0\u07d4\xb8\x84\xad\u060d\x83\xdcVJ\xb8\xe0\xe0,\xbd\xb69\x19\xae\xa8D\x89lk\x93[\x8b\xbd@
\x00\x00\u07d4\xb8\x8a7\xc2\u007fx\xa6\x17\xd5\xc0\x91\xb7\u0577:7a\xe6_*\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4\xb8\x94x\"\u056c\u79ad\x83&\xe9T\x96\"\x1e\v\xe6\xb7=\x89\x01\x15\x8eF\t\x1
3\xd0\x00\x00\u07d4\xb8\x9c\x03n\xd7\u0112\x87\x99!\xbeA\xe1\f\xa1i\x81\x98\xa7L\x89b\xa9\x9
2\xe5:\n\xf0\x00\x00\xe0\x94\xb8\x9fF2\xdfY\t\xe5\x8b*\x99d\xf7O\xeb\x9a;\x01\xe0\u014a\x04\x8
8u\xbc\xc6\xe7\xcb\xeb\x80\x00\u07d4\xb8\xa7\x9c\x84\x94^G\xa9\xc3C\x86\x83\u05b5\x84,\xffv\
x84\xb1\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb8\xa9y5'Y\xba\t\xe3Z\xa5\x93]\xf1u\xbf\xf6x\xa1\b\
x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xb8\xab9\x80[\xd8!\x18Ol\xbd=$s4{\x12\xbf\x17\\\x89\
x06hZ\xc1\xbf\xe3,\x00\x00\xe0\x94\xb8\xac\x11}\x9f\r\xba\x80\x90\x14E\x82<L\x9dO\xa3\xfe\xdc
n\x8a\x03VLD'\xa8\xfc}\x80\x00\u07d4\xb8\xbc\x9b\xca\u007fq\xb4\xed\x12\xe6
C\x8db\x0fS\xc1\x144/\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xb8\xbe\xddWjKL
'\xdasZ[\xc3\xf53%*\x18\b\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xb8\xc2p=\x8c?/D\u0144\xbc\x1
0\xe7\xc0\xa6\xb6L\x1c\t~\x8a\x01,\u0778\xea\xd6\xf9\xf8\x00\x00\xe0\x94\xb8\xcc\x0f\x06\n\xad\
x92\xd4\xeb\x8b6\xb3\xb9\\\xe9\xe9\x0e\xb3\x83\u05ca\x1f\u00c4+\xd1\xf0q\xc0\x00\x00\u07d4\x
b8\xd2\xdd\xc6o0\x8c\x01X\xae<\xcb{\x86\x9f}\x19\x9d{2\x89-
\xcb\xf4\x84\x0e\xca\x00\x00\x00\u07d4\xb8\u04c9\xe6$\xa3\xa7\xae\xbc\xe4\xd3\xe5\xdb\xdfl\xd
c)\x93*\xed\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xb8\xd51\xa9d\xbc\xea\x13\x82\x96
\xc0\xce\xd7$\"\xda\xdbL\u0289\t7\x15\xccZ\xb8\xa7\x00\x00\xe0\x94\xb8\xd5\xc3$\xa8
\x9d|\x80I\xd0\u052e\xde\x02\xba\x80\xabW\x8b\x8a\x03\x93\x92\x86)\xff\xf7^\x80\x00\xe0\x94\x
b8\xf2\x00\x05\xb6\x13R\xff\xa7i\x9a\x1bR\xf0\x1fZ\xb3\x91g\xf1\x8a\x02\x1e\x19\xe0\u027a\xb2
@\x00\x00\u07d4\xb8\xf3\aX\xfa\xa8\b\xdb\xc9\x19\xaa{B^\xc9\"\xb9;\x81)\x8966\u05ef^\u024e\x
00\x00\u07d4\xb9\x01<Q\xbd\a\x8a\t\x8f\xae\x05\xbf*\xce\bI\u01be\x17\xa5\x89\x04V9\x18$O@\x
00\x00\u07d4\xb9\x14Kg|-\xc6\x14\xce\xef\xdfP\x98_\x11\x83
\x8e\xa6L\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb9\x16\xb1\xa0\x1c\xdcNV\xe7ew\x15\xea7\xe2\x
a0\xf0\x87\xd1\x06\x89\x82n1\x81\xe0'\x06\x80\x00\u07d4\xb9\x1d\x9e\x91l\xd4\r\x19=\xb6\x0ey
'x\xa0\bw\x16\xfc\x89\x15\xf1\xba\u007fG\x16
\x00\x00\xe0\x94\xb9#\x1e\xb2n_\x9eKM(\x8f\x03\x90g\x04\xfa\xb9l\x87\u058a\x04+\xf0kx\xed;P\
x00\x00\u07d4\xb9$'\xadux\xb4\xbf\xe2\n\x9fc\xa7\xc5Pm\\\xa1-
\u0209lk\x93[\x8b\xbd@\x00\x00\u07d4\xb9'\xab\xd2\u048a\xaa\xa2M\xb3\x17x\xd2t\x19\u07ce\x
1b\x04\xbb\x89\x01~\x11\u00a2oG\x80\x00\u07d4\xb9MG\xb3\xc0R\xa5\xe5\x0eBa\xae\x06\xa2\
x0fE\xd8\xee\u25c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xb9S\x96\u06aaI\r\xf2V\x93$\xfc\xc6b;\xe0R\
xf12\u0289lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xb9Y\xdc\xe0.\x91\xd9\xdb\x02\xb1\xbd\x8b}\x17\xa
9\xc4\x1a\x97\xaf\t\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xb9\\\x9b\x10\xaa\x98\x1c\xf4\xa6
zq\xccR\xc5\x04\xde\xe8\xcfX\xbd\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xb9\\\xfd\xa8F[\xa9\
xc2f\x1b$\x9f\u00ebf\x1b\u07e3_\xf0\x89\x11JNy\xa2\xc2\x10\x80\x00\xe0\x94\xb9hA\u02bb\xc7\
xdb\u059e\xf0\u03cf\x81\xdf\xf3\u0225\xe2\x15p\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xb9zg
3\xcd_\xe9\x98d\xb3\xb34`\xd1g$4\xd5\xca\xfd\x89le\xbb\xaaF\xc2\u03c0\x00\xe0\x94\xb9\x81\xa
d^kw\x93\xa2?\xc6\xc1\xe8i.\xb2\x96]\x18\xd0\u068a\x02\x1e\x18\xd2\xc8!\xc7R\x00\x00\u07d4\x
b9\x8c\xa3\x17\x85\xef\x06\xbeI\xa1\xe4~\x86O`\xd0v\xcaG.\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\
u07d4\xb9\x92\x0f\xd0\xe2\xc75\xc2VF<\xaa$\x0f\xb7\xac\x86\xa9=\xfa\x89_h\xe8\x13\x1e\u03c0
\x00\x00\u07d4\xb9\x92\xa9g0\x8c\x02\xb9\x8a\xf9\x1e\xe7`\xfd;kH$\xab\x0e\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4\xb9\xa9\x85P\x1e\xe9P\x82\x9b\x17\xfa\xe1\xc9\xcf4\x8c1VT,\x89\x0f\xf1u\x17\

u029ab\x00\x00\xe0\x94\xb9\xb0\xa3!\x9a2\x88\u0673[\t\x1b\x14e\v\x8f\xe2=\xce+\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\xe0\x94\xb9\xcfq\xb2&X>:\x92\x11\x03\xa51o\x85Zew\x9d\x1b\x8a\x05\x15\n\xe8J\x8c\xdf\x00\x00\x00\u07d4\xb9\xe9\f\x11\x92\xb3\xd5\xd3\xe3\xab\a\x00\xf1\xbfe_]\xd44z\x89\x1b\x19\xe5\vD\x97|\x00\x00\u07d4\xb9\xfd83\xe8\x8e|\xf1\xfa\x98y\xbd\xf5Z\xf4\xb9\x9c\xd5\xce?\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xba\x02I\xe0\x1d\x94[\xef\x93\xee^\xc6\x19%\xe0<\\\xa5\t\xfd\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xba\x0f9\x02;\xdb)\xeb\x18b\xa9\xf9\x05\x9c\xab]0nf/\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xba\x10\xf2vB\x90\xf8uCCr\xf7\x9d\xbfq8\x01\u02ac\x01\x893\xc5I\x901r\f\x00\x00\u07d4\xba\x151\xfb\x9ey\x18\x96\xbc\xf3\xa8\x05X\xa3Y\xf6\xe7\xc1D\xbd\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\xba\x17m\xbe2I\xe3E\xcdO\xa9g\xc0\xed\x13\xb2LG\u5189\x15\xae\xf9\xf1\xc3\x1c\u007f\x00\x00\xe0\x94\xba\x1f\x0e\x03\u02da\xa0!\xf4\xdc\xeb\xfa\x94\xe5\u0209\xc9\u01fc\x9e\x8a\x06\u0450\xc4u\x16\x9a
\x00\x00\u07d4\xba\x1f\xca\xf2#\x93~\xf8\x9e\x85gU\x03\xbd\xb7\xcaj\x92\x8bx\x89\"\xb1\xc8\xc1\"z\x00\x00\x00\xe0\x94\xba$\xfcCgS\xa79\xdb,\x8d@\xe6\xd4\xd0LR\x8e\x86\xfa\x8a\x02\xc0\xbb=\xd3\fN
\x00\x00\u07d4\xbaB\xf9\xaa\xceL\x18E\x04\xab\xf5BWb\xac\xa2oq\xfb\u0709\x02\a\a}\u0627\x9c\x00\x00\u07d4\xbaF\x9a\xa5\u00c6\xb1\x92\x95\u0521\xb5G;T\x03S9\f\x85\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbad@\xae\xb3s{\x8e\xf0\xf1\xaf\x9b\f\x15\xf4\xc2\x14\xff\xc7\u03c965\u026d\xc5\u07a0\x00\x00\xe0\x94\xbam1\xb9\xa2a\xd6@\xb5\u07a5\x1e\xf2\x16,1\t\xf1\uba0a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xbap\xe8\xb4u\x9c\f<\x82\xcc\x00\xacN\x9a\x94\xdd[\xaf\xb2\xb8\x890C\xfa3\xc4\x12\xd7\x00\x00\u07d4\xba\x8ac\xf3\xf4\r\u4a03\x88\xbcP!/\xea\x8e\x06O\xbb\x86\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xba\x8eF\u059d.#C\xd8l`\xd8,\xf4, A\xa0\xc1\u0089\x05k\xc7^-c\x10\x00\x00\u07d4\xba\xa4\xb6L+\x15\xb7\x9f_
BF\xfdp\xbc\xbd\x86\xe4\xa9*\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xba\u0212,J\xcc},\xb6\xfdY\xa1N\xb4\\\xf3\xe7\x02!K\x89+^:\xf1k\x18\x80\x00\x00\u07d4\xba\xd25\xd5\b]\u01f0h\xa6|A&w\xb0>\x186\x88L\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xba\xd4B^\x17\x1c>r\x97^\xb4j\xc0\xa0\x15\xdb1Z]\x8f\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xba\xdc*\xef\x9fYQ\xa8\u05cak5\xc3\u0433\xa4\xe6\xe2\xe79\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xba\xdeCY\x9e\x02\xf8OL0\x14W\x1c\x97k\x13\xa3le\xab\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xba\xe9\xb8/r\x99c\x14\be\x9d\xd7N\x89\x1c\xb8\xf3\x86\x0f\xe5\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94\xbb\x03f\xa7\u03fd4E\xa7\r\xb7\xfeZ\xe3H\x85uO\xd4h\x8a\x01M\xef,B\xeb\xd6@\x00\x00\u07d4\xbb\aj\xac\x92
\x80i\xea1\x8a1\xff\x8e\xeb\x14\xb7\xe9\x96\xe3\x89\b\x13\xcaV\x90m4\x00\x00\u07d4\xbb\bW\xf1\xc9\x11\xb2K\x86\u0227\x06\x81G?\u6aa1\xcc\xe2\x89\x05k\xc7^-c\x10\x00\x00\u0794\xbb\x19\xbf\x91\u02edt\xcc\xeb_\x81\x1d\xb2~A\x1b\xc2\xea\x06V\x88\xf4?\xc2\xc0N\xe0\x00\x00\xe0\x94\xbb'\u01a7\xf9\x10uGZ\xb2)a\x90@\xf8\x04\xc8\xeczj\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xbb7\x1cr\xc9\xf01l\xea+\xd9\xc6\xfb\xb4\a\x9ewT)\xef\x89_h\xe8\x13\x1e\u03c0\x00\x00\xe0\x94\xbb;\x01\v\x18\xe6\xe2\xbe\x115\x87\x10&\xb7\xba\x15\xea\x0f\xde$\x8a\x02
|\x800\x9bwp\x00\x00\xe0\x94\xbb;\x90\x05\xf4o\xd2\xca;0\x16%\x99\x92\x8cw\xd9\xf6\xb6\x01\x8a\x01\xb1\xae\u007f+\x1b\xf7\xdb\x00\x00\u07d4\xbb?\xc0\xa2\x9c\x03Mq\b\x12\xdc\xc7u\xc8\u02b9\u048diu\x899\xd4\xe8D\xd1\xcf_\x00\x00\u07d4\xbbH\xea\xf5\x16\xce-\xec>A\xfe\xb4\xc6y\xe4\x95vA\x16O\x89\xcf\x15&@\xc5\xc80\x00\x00\u07d4\xbbKJKT\x80p\xffA

C,\x9e\b\xa0\xcao\xa7\xbc\x9fv\x89.\x14\x1e\xa0\x81\xca\b\x00\x00\u07d4\xbbV\xa4\x04r<\xff
\xd0hT\x88\xb0Z\x02\xcd\xc3Z\xac\xaa\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xbba\x8e%\"\x
1a\u0667@\xb2\x99\xed\x14\x06\xbc94\xb0\xb1m\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xbba\x
a0K\xff\xd5|\x10G\rE\u00d1\x03\xf6FP4v\x16\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xbbh#\xa1\x
bd\x81\x9f\x13QU8&J-
\xe0R\xb4D\"\b\x89\x01ch\xffO\xf9\xc1\x00\x00\u07d4\xbbl(J\xac\x8ai\xb7\\\u0770\x0f(\xe1EX;V\x
be\u0389lk\x93[\x8b\xbd@\x00\x00\u07d4\xbbu\xcbPQ\xa0\xb0\x94KFs\xcau*\x97\x03\u007f|\x8c\
x15\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbb\x99;\x96\xee\x92Z\xda}\x99\u05c6W=?\x89\x18\f\u3a89lk\x93[\x8b\xbd@\x00
\x00\u07d4\xbb\xa3\u0180\x04$\x8eH\x95s\xab\xb2t6w\x06k$\u0227\x89lk\x93[\x8b\xbd@\x00\x0
0\u07d4\xbb\xa4\xfa\xc3\xc4
9\xd8(\xe7B\xcd\xe0\xef\xff\xe7t\x94\x1b9\x89lj\u04c2\xd4\xfba\x00\x00\u07d4\xbb\xa8\xab\"\xd2\
xfe\xdb\xcf\xc6?hL\b\xaf\xdf\x1c\x17P\x90\xb5\x89\x05_)\xf3~N;\x80\x00\u07d4\xbb\xa9v\xf1\xa1!
_u\x12\x87\x18\x92\xd4_pH\xac\xd3V\u0209lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xbb\xab\x00\v\x04
\b\xed\x01Z7\xc0GG\xbcF\x1a\xb1N\x15\x1b\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xbb\xa
b\xf6d;\xebK\xd0\x1c\x12\v\xd0Y\x8a\t\x87\xd8)g\u0449\xb52\x81x\xad\x0f*\x00\x00\u07d4\xbb\xb
4\xee\x1d\x82\xf2\xe1VD,\xc938\xa2\xfc(o\xa2\x88d\x89JD\x91\xbdm\xcd(\x00\x00\u07d4\xbb\xb5
\xa0\xf4\x80,\x86H\x00\x9e\x8ai\x98\xaf5,\u0787TO\x89\x05-
T(\x04\xf1\xce\x00\x00\u07d4\xbb\xb6C\xd2\x18{6J\xfc\x10\xa6\xfd6\x8d}U\xf5\r\x1a<\x8965\u026
d\xc5\u07a0\x00\x00\u07d4\xbb\xb8\xff\xe4?\x98\u078e\xae\x18F#\xaeRd\xe4$\u0438\u05c9\x05\
xd5?\xfd\xe9(\b\x00\x00\u07d4\xbb\xbdn\u02f5u(\x91\xb4\u03b3\xcc\xe7:\x8fGpY7o\x89\x01\xf3\x
99\xb1C\x8a\x10\x00\x00\u07d4\xbb\xbf9\xb1\xb6y\x95\xa4\"APO\x97\x03\u04a1JQV\x96\x89U\x
a6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\xbb\xc8\xea\xffc~\x94\xfc\u014d\x91<wp\u020f\x9bG\x92w\x
89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbb\xc9\xd8\x11.[\xeb\x02\xdd)\xa2%{\x1f\xe6\x9b56\xa9E\x89lk\x93[\x8b\xbd@\x
00\x00\u07d4\xbb\xcae\xb3&n\xa2\xfbs\xa0?\x92\x165\xf9\x12\u01fe\xde\x00\x89j\xcb=\xf2~\x1f\x
88\x00\x00\u07d4\xbb\xf8B\x92\xd9T\xac\xd9\xe4\a/\xb8`\xb1PA\x06\xe0w\xae\x89QP\xae\x84\xa
8\xcd\xf0\x00\x00\u07d4\xbb\xf8Z\xaa\xa6\x83s\x8f\a;\xae\xf4J\xc9\xdc4\xc4\xc7y\xea\x89lk\x93[\
x8b\xbd@\x00\x00\u07d4\xbb\xf8am\x97rJ\xf3\xde\xf1e\xd0\xe2\x8c\u0689\xb8\x00\x00\x9a\x89\x
06.\xf1.+\x17a\x80\x00\xe0\x94\xbb\xfe\n\x83\f\xac\xe8{r\x93\x99:~\x94\x96\xced\xf8\u350a\x01E
B\xba\x12\xa37\xc0\x00\x00\u07d4\xbc\f\xa4\xf2\x17\xe0Ru6\x14\u05b0\x19\x94\x88$\xd0\xd8h\x
8b\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\xbc\x0e\x87E\u00e5ID\\+\xe9\x00\xf5#\x00\x80J\
xb5b\x89\x8a\a\x02\x9b\xf5\xddLS\xb2\x80\x00\u07d4\xbc\x0f\x98Y\x8f\x88\x05j&3\x96
\x92;\x8f\x1e\xb0t\xa9\xfd\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbc\x16\t\u0585\xb7kH\uc41a\xa0\x99!\x90\"\xf8\x9b,\u0349@\x13\x8b\x91~\u07f
8\x00\x00\xe0\x94\xbc\x17\x1eS\xd1z\u0276\x12A\xaeCm\xee\u01efE.t\x96\x8a\x01!\xeah\xc1\x1
4\xe5\x10\x00\x00\u07d4\xbc\x1b\x02\x1ax\xfd\xe4-
\x9bR&\xd6\xec&\xe0j\xa3g\x00\x90\x89\x04V9\x18$O@\x00\x00\u07d4\xbc\x1e\x80\xc1\x81acB\
xeb\xb3\xfb9\x92\a/\x1b(\xb8\x02\u0189\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xbc#qH\xd3\f\x13\
x83o\xfa,\xadR\x0e\xe4\xd2\xe5\xc4\xee\xff\x89j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94\xbcF\xd57\xcf
.\xdd@5e\xbd\xe73\xb2\xe3K!P\x01\xbd\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xbcNG\x15`\u
025c\x8a*K\x1b\x1a\xd0\xc3j\xa6P+|K\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xbcb\xb3\tj\x91\x

e7\xdc\x11\xa1Y*)=\xd2T!P\xd7Q\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xbci\xa0\u04a3\x1c=\xbfz\x91\"\x11i\x01\xb2\xbd\xfe\x98\x02\xa0\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xbckX6K\xf7\xf1\x95\x1c0\x9e\f\xba\x05\x95
\x1c\xd7?\x9a\x89b@\x1aE~E\xf8\x00\x00\u07d4\xbcs\xf7\xb1\xca;w;4$\x9a\xda.,\x8a\x92t\xcc\x17\u0089lk\x93[\x8b\xbd@\x00\x00\u07d4\xbcz\xfc\x84wA\"t\xfc&]\xf1<\x05DsB}C\u0189\a\f\x95\x92\f\xe3%\x00\x00\xe0\x94\xbc\x96\u007f\xe4A\x8c\x18\xb9\x98X\x96m\x87\x06x\u0722\xb8\x88y\x8a\x01\xd9\xcb\u074d~\xd2\x10\x00\x00\u07d4\xbc\x99\x9e8\\Z\xeb\xca\xc8\xd6\xf3\xf0\xd6\rZ\xa7%3m\r\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbc\x9c\x95\u07eb\x97\xa5t\u03a2\xaa\x80;\\\xaa\x19|\xef\f\xff\x89\x16\u012b\xbe\xbe\xa0\x10\x00\x00\u07d4\xbc\x9e\x0e\xc6x\x8f}\xf4\xc7\xfc!\n\xac\xd2
\xc2~E\xc9\x10\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xbc\xa3\xff\xd4h?\xba\n\u04fb\xc9\a4\xb6\x11\u069c\xfbE~\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbc\xae\u042c\xb6\xa7o\x11?|a5U\xa2\u00f0\xf5\xbf4\xa5\x89\n~\xbd^Cc\xa0\x00\x00\u07d4\xbc\xaf4y\x18\xef\xb2\xd6=\xde\x03\xe3\x92u\xbb\xe9}&\xdfP\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xbc\xb4\"\xdcM\u04aa\xe9J\xba\xe9^\xa4]\xd1s\x1b\xb6\xb0\xba\x89\x18BO_\v\x1bN\x00\x00\u07d4\xbc\xbd1%.\u0088\xf9\x1e)\x8c\xd8\x12\xc9!`\xe783\x1a\x89k\x1b\xc2\xca\xc0\x9aY\x00\x00\u07d4\xbc\xbfk\xa1f\xe24\r\xb0R\xea#\u0480)\xb0\xdej\xa3\x80\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\xbc\xc8E\x97\xb9\x1es\xd5\u0174\u059c\x80\xec\xf1F\x86\x0fw\x9a\x89\xedp\xb5\xe9\xc3\xf2\xf0\x00\x00\u07d4\xbc\xc9Y;-
\xa6\xdfj4\xd7\x1b\x1a\xa3\x8d\xac\xf8v\xf9[\x88\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xbc\xd9^\xf9bF+n\u07e1\x0f\u0687\xd7\"B\xfe>\xdb\\\x89\x12\x1d\x06\xe1/\xff\x98\x80\x00\u07d4\xbc\u065e\xdc!`\xf2\x10\xa0^:\x1f\xa0\xb0CL\xed\x00C\x9b\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbc\u07ec\xb9\xd9\x02<4\x17\x18.\x91\x00\xe8\xea\x1d73\x93\xa3\x89\x034-
`\xdf\xf1\x96\x00\x00\u07d4\xbc\xe1>\"2*\u03f3U\xcd!\xfd\r\xf6\f\xf9:\xdd&\u0189\n\u05ce\xbcZ\xc6 \x00\x00\xe0\x94\xbc\xe4\x04u\xd3E\xb0q-\xeep=\x87\xcdvW\xfc\u007f;b\x8a\x01\xa4
\xdb\x02\xbd}X\x00\x00\u07d4\xbc\xed\xc4&|\u02c9\xb3\x1b\xb7d\xd7!\x11q\x00\x8d\x94\xd4M\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbc\xfc\x98\xe5\xc8+j\xdb\x18\n?\xcb\x12\v\x9av\x90\xc8j?\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xbd\x04;g\xc6>`\xf8A\xcc\xca\x15\xb1)\xcd\xfee\x90\xc8\xe3\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbd\x04\u007f\xf1\xe6\x9c\u01b2\x9a\xd2d\x97\xa9\xa6\xf2z\x90?\xc4\u0749.\xe4IU\b\x98\xe4\x00\x00\u07d4\xbd\b\xe0\xcd\xde\xc0\x97\xdby\x01\ua05a=\x1f\xd9\u0789Q\xa2\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xbd\t\x12l\x89\x1cJ\x83\x06\x80Y\xfe\x0e\x15ylFa\xa9\xf4\x89+^:\xf1k\x18\x80\x00\x00\u07d4\xbd\f\\\u05d9\xeb\u0106B\xef\x97\xd7N\x8eB\x90d\xfe\u4489\x11\xac(\xa8\xc7)X\x00\x00\u07d4\xbd\x17\xee\xd8+\x9a%\x92\x01\x9a\x1b\x1b<\x0f\xba\xd4\\@\x8d\"\x89\r\x8drkqw\xa8\x00\x00\u07d4\xbd\x18\x037\v\u0771)\xd29\xfd\x16\xea\x85&\xa6\x18\x8a\u5389\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xbd+p\xfe\xcc7d\x0fiQO\xc7\xf3@IF\xaa\xd8k\x11\x89A\rXj
\xa4\xc0\x00\x00\u07d4\xbd0\x97\xa7\x9b<\r.\xbf\xf0\xe6\xe8j\xb0\xed\xad\xbe\xd4p\x96\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4\xbd2]@)\xe0\xd8r\x9fm9\x9cG\x82$\xae\x9ez\xe4\x1e\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\xbdC*9\x16$\x9bG$):\xf9\x14nI\xb8(\n\u007f*\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xbdG\xf5\xf7n;\x93\x0f\xd9HR\t\xef\xa0\xd4v=\xa0uh\x8965\u026d

\xc5\u07a0\x00\x00\u07d4\xbdK`\xfa\xect\n!\xe3\a\x13\x91\xf9j\xa54\xf7\xc1\xf4N\x89\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\u07d4\xbdK\u0571\"\xd8\xef{|\x8f\x06gE\x03 \xdb!\x16\x14.\x89
\x86\xac5\x10R`\x00\x00\u07d4\xbdQ\xee.\xa1C\u05f1\u05b7~~D\xbd\xd7\xda\x12\U00105b09G~\x06\u0332\xb9(\x00\x00\u07d4\xbdY\tN\aO\x8dy\x14*\xb1H\x9f\x14\x8e2\x15\x1f
\x89\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xbdZ\x8c\x94\xbd\x8b\xe6G\x06D\xf7\f\x8f\x8a3\xa8\xa5\\cA\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbd^G:\xbc\xe8\xf9zi2\xf7|/\xac\xaf\x9c\xc0\xa0\x05\x14\x89<\x92X\xa1\x06\xa6\xb7\x00\x00\u07d4\xbd_F\u02ab,=K(\x93\x96\xbb\xb0\u007f
<M\xa8%0\x89\x04V9\x18$O@\x00\x00\u07d4\xbdf\xff\xed\xb50\xea\v.\x85m\xd1*\xc2)l1\xfe)\xe0\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xbdg\xd2\xe2\xf8-
\xa8\x86\x13A\xbc\x96\xa2\xc0y\x1f\xdd\xf3\x9e@\x89\n\xd7\xc0yG\xc8\xfb\x00\x00\u07d4\xbdjGMf4[\xcd\xd7\aYJ\xdbc\xb3\fx\"\xafT\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xbdr;(\x9asg\xb6\xec\xe2E^\xd6\x1e\xdbIg\n\xb9\u010a\x01\x0f\f\u07a1d!?\x80\x00\u07d4\xbds\xc3\xcb\xc2j\x17Pb\xea\x03
\u0744\xb2S\xbc\xe6CX\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\xe0\x94\xbdt\x19\xdc*\t\nF\xe2\x87=}\xe6\xea\xaa\u055e\x19\xc4y\x8a\x01p\xbc\xb6qu\x9f\b\x00\x00\u07d4\xbd\x87e\xf4\x12\x99\xc7\xf4y\x92<O\u044f\x12mr)\x04}\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xbd\x93\xe5P@>*\x06\x11>\xd4\xc3\xfb\xa1\xa8\x91;\x19@~\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbd\x9eV\xe9\x02\xf4\xbe\x1f\xc8v\x8d\x808\xba\xc6>*\u02ff\x8e\x8965f3\xeb\xd8\xea\x00\x00\u07d4\xbd\xa4\xbe1~~K\xed\x84\xc0I^\xee2\xd6\a\xec8\xcaR\x89}2'yx\xefN\x80\x00\u07d4\xbd\xb6\v\x82:\x11s\xd4Z\a\x92$_\xb4\x96\xf1\xfd3\x01\u03c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xbd\xba\xf6CM@\xd65[\x1e\x80\xe4\f\u012b\x9ch\xd9a\x16\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xbd\xc0,\xd43\f\x93\xd6\xfb\xdaOm\xb2\xa8]\xf2/C\xc23\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbd\xc4aF+c\"\xb4b\xbd\xb3?\"y\x9e\x81\b\xe2A}\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\xbd\xc79\xa6\x99p\v.\x8e,JL{\x05\x8a\x0eQ=\u07be\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbd\xc7Hs\xaf\x92+\x9d\xf4t\x85;\x0f\xa7\xff\v\xf8\xc8&\x95\x89\xd8\xc9F\x00c\xd3\x1c\x00\x00\u07d4\xbd\xca*\x0f\xf3E\x88\xafb_\xa8\xe2\x8f\xc3\x01Z\xb5\xa3\xaa\x00\x89~\xd7?w5R\xfc\x00\x00\u07d4\xbd\xd3%N\x1b:m\xc6\xcc,i}Eq\x1a\xca!\xd5\x16\xb2\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xbd\u07e3M\x0e\xbf\x1b\x04\xafS\xb9\x9b\x82IJ\x9e=\x8a\xa1\x00\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xbd\xe4\xc7?\x96\x9b\x89\xe9\u03aef\xa2\xb5\x18DH\x0e\x03\x8e\x9a\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xbd\xe9xj\x84\xe7[H\xf1\x8erm\u05cdp\xe4\xaf>\xd8\x02\x8a\x016\x9f\xb9a(\xacH\x00\x00\u07d4\xbd\xed\x11a/\xb5\xc6\u0699\xd1\xe3\x0e2\v\xc0\x99Tf\x14\x1e\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xbd\xed~\a\xd0q\x1ehM\xe6Z\u0232\xabW\xc5\\\x1a\x86E\x89
\t\xc5\u023fo\xdc\x00\x00\u07d4\xbd\xf6\x93\xf83\xc3\xfeG\x17S\x18G\x88\xebK\xfeJ\xdc?\x96\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xbd\xf6\xe6\x8c\f\xd7X@\x80\xe8G\xd7,\xbb#\xaa\xd4j\xeb\x1d\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xbe\n/8_\t\xdb\xfc\xe9g2\xe1+\xb4\n\xc3l\x87\x1b\xa8\x89WL\x11^\x02\xb8\xbe\x00\x00\u07d4\xbe\f*\x80\xb9\xde\bK\x17(\x94\xa7l\xf4szOR\x9e\x1a\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xbe\x1c\xd7\xf4\xc4r\a\th\xf3\xbd\xe2h6k!\xee\xea\x83!\x89\xe9\x1a|\u045f\xa3\xb0\x00\x00\u07d4\xbe#F\xa2\u007f\xf9\xb7\x02\x04OP\r\xef\xf2\xe7\xff\xe6\x82EA\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xbe$q\xa6\u007f`G\x91\x87r\xd0\xe3h9%^\xd9\u0591\xae\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xbe+\"\x80R7h\xea\x8a\xc3\\\xd9\xe8\x88\xd6\nq\

x93\x00\u0509lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xbe+2nx\xed\x10\xe5P\xfe\xe8\xef\xa8\xf8\a\x03
\x96R/Z\x8a\bW\xe0\xd6\xf1\xdav\xa0\x00\x00\xe0\x94\xbe0Zyn3\xbb\xf7\xf9\xae\xaee\x12\x95\x
90f\xef\xda\x10\x10\x8a\x02M\xceT\xd3J\x1a\x00\x00\x00\u07d4\xbeG\x8e\x8e=\xdek\xd4\x03\xb
b-
\x1ce|C\x10\xee\x19'#\x89\x1a\xb2\xcf|\x9f\x87\xe2\x00\x00\u07d4\xbeN}\x98?.*ck\x11\x02\xecp9\
xef\xeb\xc8B\u9349\x03\x93\xef\x1aQ'\xc8\x00\x00\u07d4\xbeO\xd0sap\"\xb6\u007f\\\x13I\x9b\x8
2\u007fv69\xe4\xe3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbeRZ3\xea\x91aw\xf1r\x83\xfc\xa2\x9e\
x8b5\v\u007fS\v\x89\x8f\x01\x9a\xafF\xe8x\x00\x00\u07d4\xbeS2/C\xfb\xb5\x84\x94\xd7\xcc\xe1\
x9d\xda'+$P\xe8'\x89\n\xd7\u03afB\\\x15\x00\x00\u07d4\xbeS\x82F\xddNo\f
\xbfZ\xd17<;F:\x13\x1e\x86\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbeZ`h\x99\x98c\x9a\xd7[\xc1\x05\xa3qt>\xef\x0fy@\x89\x1b2|s\xe1%z\x00\x00\u
07d4\xbe\\\xba\x8d7By\x86\xe8\xca&\x00\xe8X\xbb\x03\xc3YR\x0f\x89\xa00\xdc\xeb\xbd/L\x00\x0
0\u07d4\xbe`\x03~\x90qJK\x91~a\xf1\x93\xd84\x90g\x03\xb1:\x89\\(=A\x03\x94\x10\x00\x00\u07d
4\xbec:77\xf6\x849\xba\xc7\xc9\nR\x14
X\ue38ao\x894\n\xad!\xb3\xb7\x00\x00\x00\xe0\x94\xbee\x9d\x85\xe7\xc3O\x883\xea\u007fH\x8d
\xe1\xfb\xb5\xd4\x14\x9b\xef\x8a\x01\xeb\xd2:\xd9\u057br\x00\x00\u07d4\xbes'M\x8cZ\xa4J<\xbe
\xfc\x82c\xc3{\xa1!\xb2\n\u04c9\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xbe\x86\u0430C\x84\x1
9\u03b1\xa081\x927\xbaR\x06\xd7.F\x8964\xfb\x9f\x14\x89\xa7\x00\x00\u07d4\xbe\x8d\u007f\x18
\xad\xfe]l\xc7u9I\x89\xe1\x93\f\x97\x9d\x00}\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xbe\x91\x86\
xc3JRQJ\xbb\x91\a\x86\x0fgO\x97\xb8!\xbd[\x89\x1b\xa0\x1e\xe4\x06\x03\x10\x00\x00\u07d4\xbe
\x93W\x93\xf4[p\xd8\x04]&T\xd8\xdd:\xd2K[a7\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4\xbe\x98\xa7\u0
07f\xd4\x10\x97\xb3OY\xd7X\x9b\xaa\xd0!e\x9f\xf7\x12\x890\xca\x02O\x98{\x90\x00\x00\u07d4\x
be\x9b\x8c4\xb7\x8e\xe9G\xff\x81G.\xdaz\xf9\xd2\x04\xbc\x84f\x89\b!\xab\rD\x14\x98\x00\x00\u0
7d4\xbe\xa0\r\xf1pg\xa4:\x82\xbc\x1d\xae\xca\xfbl\x140\x0e\x89\xe6\x89b\xa9\x92\xe5:\n\xf0\x00\
x00\u07d4\xbe\xa0\xaf\xc9:\xae!\b\xa3\xfa\xc0Yb;\xf8o\xa5\x82\xa7^\x89\\(=A\x03\x94\x10\x00\x0
0\u07d4\xbe\xb35\x8cP\u03dfu\xff\xc7mD<,\u007fU\aZ\x05\x89\x89\x90\xf54`\x8ar\x88\x00\x00\u
07d4\xbe\xb4\xfd1UYC`E\u0739\x9dI\xdc\xec\x03\xf4\fB\u0709lk\x93[\x8b\xbd@\x00\x00\u07d4\x
be\xc2\xe6\xde9\xc0|+\xaeUj\u03fe\xe2\xc4r\x8b\x99\x82\xe3\x89\x1f\x0f\xf8\xf0\x1d\xaa\xd4\x00\
x00\u07d4\xbe\xc6d\x0fI\t\xb5\x8c\xbf\x1e\x80cB\x96\x1d`u\x95\tl\x89lj\xccg\u05f1\xd4\x00\x00\u0
7d4\xbe\xc8\xca\xf7\xeeIF\x8f\xeeU.\xff:\xc5#N\xb9\xb1}B\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xb
e\xce\xf6\x1c\x1cD+\xef|\xe0Ks\xad\xb2I\xa8\xba\x04~\x00\x896;V\u00e7T\xc8\x00\x00\u0794\xb
e\xd4d\x9d\xf6F\u2052)\x03-
\x88hUo\xe1\xe0S\u04c8\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94\xbe\xd4\xc8\xf0\x06\xa2|\x1e_|\x
e2\x05\xdeu\xf5\x16\xbf\xb9\xf7d\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07d4\xbe\xe8\u0430\bB\x
19T\xf9-
\x00\r9\x0f\xb8\xf8\xe6X\xea\xee\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xbe\xec\u05af\x90\f\x8b\
x06J\xfc\xc6\a?-
\x85\u055a\xf1\x19V\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xbe\xef\x94!8y\xe0&\"\x14+\xeaa)\tx\x
93\x9a`\u05ca\x016\x85{2\xad\x86\x04\x80\x00\xe0\x94\xbe\xf0}\x97\xc3H\x1f\x9dj\xee\x1c\x98\xf
9\xd9\x1a\x18\n2D+\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xbe\xfbD\x8c\f_h?\xb6~\xe5p\xba\xf0\xdbV\x86Y\x97Q\x89j\
xcb=\xf2~\x1f\x88\x00\x00\u07d4\xbf\x05\a\f,4!\x93\x11\xc4T\x8b&\x14\xa48\x81\r\xedm\x89lk\x93

[\x8b\xbd@\x00\x00\u07d4\xbf\x05\xff^\xcf\r\xf2\u07c8wY\xfb\x82t\xd928\xac&}\x89+^:\xf1k\x18\x80\x00\x00\xe0\x94\xbf\t\xd7pH\xe2p\xb6b3\x0e\x94\x86\xb3\x8bC\xcdx\x14\x95\x8a\\S\x9b{\xf4\xff(\x80\x00\x00\u07d4\xbf\x17\xf3\x97\xf8\xf4o\x1b\xaeE\u0447\x14\x8c\x06\xee\xb9Y\xfaM\x896I\u0156$\xbb0\x00\x00\u07d4\xbf\x186A\xed\xb8\x86\xce`\xb8\x19\x02a\xe1OB\xd9<\xce\x01\x89\x01[5W\xf1\x93\u007f\x80\x00\u07d4\xbf*\xeaZ\x1d\xcfn\u04f5\xe829D\xe9\x83\xfe\xdf\u046c\xfb\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\xbf@\x96\xbcT}\xbf\xc4\xe7H\t\xa3\x1c\x03\x9e{8\x9d^\x17\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\xbfI\xc1H\x981eg\u0637\t\xc2\xe5\x05\x94\xb3f\xc6\u04cc\x89'\xbf8\xc6TM\xf5\x00\x00\u07d4\xbfLs\xa7\xed\xe7\xb1d\xfe\a!\x14\x846T\xe4\xd8x\x1d\u0789lk\x93[\x8b\xbd@\x00\x00\u07d4\xbfP\xce.&K\x9f\xe2\xb0h0az\xed\xf5\x02\xb25\x1bE\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xbfY\xae\xe2\x81\xfaC\xfe\x97\x19CQ\xa9\x85~\x01\xa3\xb8\x97\xb2\x89
\x86\xac5\x10R`\x00\x00\u07d4\xbfh\u048a\xaf\x1e\xee\xfe\xf6F\xb6^\x8c\xc8\u0450\xf6\xc6\u069c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xbfi%\xc0\aQ\x00\x84@\xa6s\x9a\x02\xbf+l\u06ab^:\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xbfw\x01\xfcb%\u0561x\x15C\x8a\x89A\xd2\x1e\xbc]\x05\x9d\x89e\xea=\xb7UF`\x00\x00\u07d4\xbf\x8b\x80\x05\xd66\xa4\x96d\xf7Bu\xefBC\x8a\xcde\xac\x91\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbf\x92A\x8a\fl1$M\"\x02`\xcb>\x86}\u05f4\xefI\x89\x05i\x00\xd3<\xa7\xfc\x00\x00\u07d4\xbf\x9a\xcdDE\xd9\xc9UF\x89\u02bb\xba\xb1\x88\x00\xff\x17A\u008965\u026d\xc5\u07a0\x00\x00\u07d4\xbf\x9f'\x1fz~\x12\xe3m\xd2\xfe\x9f\xac\xeb\xf3\x85\xfeaB\xbd\x89\x03f\xf8O{\xb7\x84\x00\x00\u07d4\xbf\xa8\xc8X\xdf\x10,\xb1$!\x00\x8b\n1\xc4\xc7\x19\n\xd5`\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xbf\xae\xb9\x10ga}\u03cbD\x17+\x02\xafaVt\x83]\xba\x89\b\xb5\x9e\x88H\x13\b\x80\x00\xe0\x94\xbf\xb0\xea\x02\xfe\xb6\x1d\xec\x9e\"\xa5\a\tY3\x02\x99\xc40r\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xbf\xbc\xa4\x18\xd3R\x9c\xb3\x93\b\x10b\x03*n\x11\x83\u01b2\u070a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xbf\xbe\x05\u831c\xbb\xcc\x0e\x92\xa4\x05\xfa\xc1\xd8]\xe2H\xee$\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xbf\xbf\xbc\xb6V\u0099+\xe8\xfc\u0782\x19\xfb\xc5J\xad\u055f)\x8a\x02\x1e\x18\xd2\xc8!\xc7R\x00\x00\u07d4\xbf\xc5z\xa6f\xfa\u239f\x10zI\xcbP\x89\xa4\xe2!Q\u074965\u026d\xc5\u07a0\x00\x00\u07d4\xbf\u02d70$c\x04p\r\xa9\vAS\xe7\x11Ab.\x1cA\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xbf\xd9<\x90\u009c\a\xbc_\xb5\xfcI\xae\xeaU\xa4\x0e\x13O5\x8a\x05\xed\xe2\x0f\x01\xa4Y\x80\x00\x00\xe0\x94\xbf\xe3\xa1\xfcn$\xc8\xf7\xb3%\x05`\x99\x1f\x93\u02e2\u03c0G\x8a\x10\xf0\xcf\x06M\u0552\x00\x00\x00\u07d4\xbf\u6f30\xf0\xc0xRd3$\xaa]\xf5\xfdb%\xab\xc3\u0289\x04\t\xe5+H6\x9a\x00\x00\u07d4\xbf\xf5\xdfv\x994\xb8\x94<\xa9\x13}\x0e\xfe\xf2\xfen\xbb\xb3N\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xbf\xfbi)$\x1fx\x86\x93'>p\"\xe6\x0e>\xab\x1f\xe8O\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc0\x06O\x1d\x94t\xab\x91]V\x90l\x9f\xb3
\xa2\xc7\t\x8c\x9b\x89\x13h?\u007f<\x15\xd8\x00\x00\u07d4\xc0\a\xf0\xbd\xb6\xe7\x00\x92\x02\xb7\xaf>\xa9\t\x02i|r\x14\x13\x89\xa2\xa0\xe4>\u007f\xb9\x83\x00\x00\u07d4\xc0\n\xb0\x80\xb6C\xe1\u00ba\xe3c\xe0\u0455\xde.\xff\xfc\x1cD\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u0794\xc0
wD\x9a\x13Jz\xd1\xef~M\x92z\xff\xec\ueb75\xae\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xc0$q\xe3\xfc.\xa0S&\x15\xa7W\x1dI2\x89\xc1<6\xef\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc0-
n\xad\xea\xcf\x1bx\xb3\u0285\x03\\c{\xb1\xce\x01\xf4\x90\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u0

7d4\xc03\xb12Z\n\xf4Tr\xc2U'\x85;\x1f\x1c!\xfa5\u0789lk\x93[\x8b\xbd@\x00\x00\u07d4\xc03\xbe\x10\xcbHa;\xd5\xeb\xcb3\xedI\x02\xf3\x8bX0\x03\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xc04[3\xf4\x9c\xe2\u007f\xe8,\xf7\xc8M\x14\x1ch\xf5\x90\xcev\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc0=\xe4*\x10\x9bezd\xe9\"$\xc0\x8d\xc1'^\x80\u0672\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc0@i\u07f1\x8b\tlxg\xf8\xbe\xe7zm\xc7Gz\xd0b\x89\x90\xf54`\x8ar\x88\x00\x00\xe0\x94\xc0A?Z|-\x9aK\x81\b(\x9e\xf6\xec\xd2qx\x15$\xf4\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\xc0C\xf2E-\u02d6\x02\xefb\xbd6\x0e\x03=\xd29q\xfe\x84\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xc0OK\xd4\x04\x9f\x04F\x85\xb8\x83\xb6)Y\xaec\x1df~5\x8a\x01;\x80\xb9\x9cQ\x85p\x00\x00\u07d4\xc0V\u053dk\xf3\u02ec\xace\xf8\xf5\xa0\xe3\x98\v\x85'@\xae\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xc0[t\x06
\xf1s\xf1nRG\x1d\u00cb\x9cQJ\v\x15&\x89\a\x96\xe3\xea?\x8a\xb0\x00\x00\u07d4\xc0i\xef\x0e\xb3B\x99\xab\xd2\xe3-
\xab\xc4yD\xb2r3H$\x89\x06\x81U\xa46v\xe0\x00\x00\u07d4\xc0l\xeb\xbb\xf7\xf5\x14\x9af\xf7\xeb\x97k>G\xd5e\x16\xda/\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xc0r^\u00bd\xc3:\x1d\x82`q\u07a2\x9db\xd48Z\x8c%\x8a\b\xa0\x85\x13F:\xa6\x10\x00\x00\u07d4\xc0~8g\xad\xa0\x96\x80z\x05\x1al\x9c4\xcc;?J\xd3J\x89`\xf0f
\xa8IE\x00\x00\u07d4\xc0\x89^\xfd\x05m\x9a:\x81\xc3\xdaW\x8a\xda1\x1b\xfb\x93V\u03c9\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xc0\x90\xfe#\xdc\xd8k5\x8c2\xe4\x8d*\xf9\x10$%\x9fef\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xc0\x9af\x17*\xea7\r\x9ac\xda\x04\xffq\xff\xbb\xfc\xff\u007f\x94\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc0\x9e<\xfc\x19\xf6\x05\xff>\xc9\xc9\xc7\x0e%@\xd7\xee\x97Cf\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xc0\xa0*\xb9N\xbeV\xd0E\xb4\x1bb\x9b\x98F.:\x02J\x93\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xc0\xa3\x93\b\xa8\x0e\x9e\x84\xaa\xaf\x16\xac\x01\xe3\xb0\x1dt\xbdk-\x89\afM\xddL\x1c\v\x80\x00\u07d4\xc0\xa6\u02edwi*=\x88\xd1A\xefv\x9a\x99\xbb\x9e<\x99Q\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xc0\xa7\xe8C]\xff\x14\xc2Uws\x9d\xb5\\$\u057fW\xa3\u064a\nm\xd9\f\xaeQ\x14H\x00\x00\u07d4\xc0\xae\x14\xd7$\x83./\xce'x\xde\u007f{\x8d\xaf{\x12\xa9>\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc0\xaf\xb7\u0637\x93p\xcf\xd6c\u018c\u01b9p*7\u035e\xff\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc0\xb0\xb7\xa8\xa6\xe1\xac\xdd\x05\xe4\u007f\x94\xc0\x96\x88\xaa\x16\u01ed\x8d\x89\x03{m\x02\xacvq\x00\x00\xe0\x94\xc0\xb3\xf2D\xbc\xa7\xb7\xde[H\xa5>\u06dc\xbe\xab\vm\x88\xc0\x8a\x01;\x80\xb9\x9cQ\x85p\x00\x00\u07d4\xc0\xc0M\x01\x06\x81\x0e>\xc0\xe5J\x19\U000ab157\xe6\x9aW=\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\xc0\xca2w\x94.tE\x87K\xe3\x1c\xeb\x90)rqO\x18#\x89\r\x8drkqw\xa8\x00\x00\u07d4\xc0\u02ed<\xcd\xf6T\xda\"\xcb\xcf\\xe\x97\xca\x19U\xc1\x15\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc0\xcb\xf6\x03/\xa3\x9e|F\xffw\x8a\x94\xf7\xd4E\xfe\"\xcf0\x89\x10\xce\x1d=\x8c\xb3\x18\x00\x00\u07d4\xc0\xe0\xb9\x03\b\x8e\fc\xf5=\xd0iWTR\xaf\xf5$\x10\u00c9\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xc0\xe4W\xbdV\xec6\xa1$k\xfa20\xff\xf3\x8eY&\xef\"\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xc0\xed\rJ\xd1\r\xe045\xb1S\xa0\xfc%\xde;\x93\xf4R\x04\x89\xabM\xcf9\x9a:`\x00\x00\u07d4\xc0\xf2\x9e\xd0\af\x11\xb5\xe5^\x13\x05G\xe6\x8aH\xe2m\xf5\u4262\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xc1\x13(x#\\]\u06e5\xd9\xf3\"\x8bR6\xe4p
\xdco\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc1\x17\r\xba\xad\xb3\xde\xe6\x19\x8e\xa5D\xba\xe

c\x93%\x18`\xfd\xa5\x89A\rXj
\xa4\xc0\x00\x00\xe0\x94\xc1&W=\x87\xb0\x17ZR\x95\xf1\xdd\a\xc5u\u03cc\xfa\x15\xf2\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xc1'\xaa\xb5\x90e\xa2\x86D\xa5k\xa3\xf1^.\xac\x13\xda)\x95\x89
\x86\xac5\x10R`\x00\x00\xe0\x94\xc1+\u007f@\u07da/{\xf9\x83f\x14\"\xab\x84\xc9\xc1\xf5\bX\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xc1,\xfb{=\xf7\x0f\xce\xca\x0e\xde&5\x00\xe2xs\xf8\xed\x16\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc1/\x88\x1f\xa1\x12\xb8\x19\x9e\xcb\xc7>\xc4\x18W\x90\xe6\x14\xa2\x0f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc18Lnq~\xbeK#\x01NQ\xf3\x1c\x9d\xf7\xe4\xe2[1\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xc1C\x8c\x99\xddQ\xef\x1c\xa88j\xf0\xa3\x17\xe9\xb0AEx\x88\x89\f\x1d\xaf\x81\u0623\xce\x00\x00\u07d4\xc1c\x12(\xef\xbf*.:@\x92\xee\x89\x00\xc69\xed4\xfb\u02093\xc5I\x901r\f\x00\x00\u07d4\xc1u\xbe1\x94\xe6iB-
\x15\xfe\xe8\x1e\xb9\xf2\xc5lg\xd9\u0249\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xc1\x82v\x86\xc0\x16\x94\x85\xec\x15\xb3\xa7\xc8\xc0\x15\x17\xa2\x87M\xe1\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xc1\x8a\xb4g\xfe\xb5\xa0\xaa\xdf\xff\x91#\x0f\xf0VFMx\xd8\x00\x89lk\x93[\x8b\xbd@\x00\x00\u0794\xc1\x95\x05CUM\x8aq0\x03\xf6b\xbba,\x10\xadL\xdf!\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xc1\xa4\x1aZ'\x19\x92&\xe4\xc7\xeb\x19\x8b\x03\x1bY\x19o\x98B\x89\nZ\xa8P\t\xe3\x9c\x00\x00\u07d4\xc1\xb2\xa0\xfb\x9c\xadE\xcdi\x91\x92\xcd'T\v\x88\xd38By\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xc1\xb2\xaa\x8c\xb2\xbfb\xcd\xc1:G\xec\xc4e\u007f\xac\xaa\x99_\x98\x8967\x93\xfa\x96\u6980\x00\u07d4\xc1\xb5\x00\x01\x1c\xfb\xa9]|\xd66\xe9^l\xbfagFK%\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xc1\xb9\xa5pM5\x1c\xfe\x98?y\xab\xee\xc3\u06fb\xae;\xb6)\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc1\xcb\xd2\xe23*RL\xf2\x19\xb1\r\x87\x1c\xcc
\xaf\x1f\xb0\xfa\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xc1\xcd\xc6\x01\xf8\x9c\x04(\xb3\x13\x02\u0447\xe0\xdc\b\xad}\x1cW\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xc1\u052f8\xe9\xbay\x90@\x89HI\xb8\xa8!\x93u\xf1\xacx\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xc1\xe1@\x9c\xa5,%CQ4\xd0\x06\u00a6\xa8T-
\xfbrs\x89\x01\xdd\x1eK\xd8\xd1\xee\x00\x00\u07d4\xc1\xeb\xa5hJ\xa1\xb2L\xbac\x15\x02c\xb7\xa9\x13\x1a\xee\u008d\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc1\xec\x81\xdd\x12=K|-
\u0674\xd48\xa7\a,\x11\u0707L\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc1\xf3\x9b\xd3]\xd9\xce\xc37\xb9oG\xc6w\x81\x81`\xdf7\xb7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u0794\xc1\xff\xad\a\u06d6\x13\x8cK*S\x0e\xc1\xc7\xde)\xb8\xa0Y,\x88\xf4?\xc2\xc0N\xe0\x00\x00\xe0\x94\xc2\x1f\xa6d:\x1f\x14\xc0)\x96\xadqD\xb7Y&\xe8~\xcbK\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xc24\nL\xa9L\x96x\xb7IL<\x85%(\xed\xe5\xeeR\x9f\x89\x02\xa3k\x05\xa3\xfd|\x80\x00\u07d4\xc29\xab\u07ee>\x9a\xf5E\u007fR\xed+\x91\xfd\n\xb4\xd9\xc7\x00\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc2;/\x92\x1c\xe4\xa3z%\x9e\u4b4b!X\xd1]fOY\x89\x01`\x89\x95\xe8\xbd?\x80\x00\u07d4\xc2C\x99\xb4\xbf\x86\xf73\x8f\xbfd^;\"\xb0\u0dd79\x12\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc2L\u03bc#D\xcc\xe5d\x17\xfbhL\xf8\x16\x13\xf0\xf4\xb9\xbd\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\xc2Rf\xc7gf2\xf1>\xf2\x9b\xe4U\ud50a\xddVw\x92\x89Hz\x9a0E9D\x00\x00\u07d4\xc2\\\xf8&U\f\x8e\xaf\x10\xaf\"4\xfe\xf9\x04\u0779R\x13\xbe\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc2f?\x81E\xdb\xfe\xc6\xc6F\xfc\\I\x96\x13E\xde\x1c\x9f\x11\x89%g\xacp9+\x88\x00\x00\u07d4\xc2pEh\x854+d\vL\xfc\x1bR\x0e\x1aTN\xe0\xd5q\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xc2sv\xf4]!\xe1^\xde;&\xf2e_\xce\xe0,\xcc\x0f*\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xc2w\x97q\xf0Smy\xa8p\x8fi1\xab\xc4K05\u964

a\x04<O\x83\x00\u0733H\x00\x00\u07d4\xc2\u007fN\b\t\x9d\x8c\xf3\x9e\xe1\x16\x01\x83\x8e\xf9\xfc\x06\xd7\xfcA\x89a\t=|,m8\x00\x00\u07d4\u0082\xe6\x99?\xbez\x91.\xa0G\x15?\xfd\x92t'\x0e([\x89\a\x96\v3\x12Gc\x80\x00\u07d4\u0083a\x88\u0662\x92S\xe0\xcb\xdaeq\xb0X\u0089\xa0\xbb2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u00aat\x84~\x86\xed\xfd\xd3\xf3\xdb\"\xf8\xa2\x15/\xee\xe5\xb7\xf7\x89o\x11\x88\x86\xb7\x84\xa2\x00\x00\xe0\x94\u00b2\xcb\xe6[\xc6\xc2\xeez<u\xb2\xe4|\x18\x9c\x06.\x8d\x8b\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\u00ba\xe4\xa23\xc2\xd8W$\xf0\u06be\xbd\xa0$\x9d\x83>7\u04ca\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xc2\xc1>r\xd2h\xe7\x15\r\u01d9\xe7\xc6\xcf\x03\u0209T\xce\u05c9%\xf2s\x93=\xb5p\x00\x00\u07d4\xc2\xcb\x1a\xda]\xa9\xa0B8s\x81G\x93\xf1aD\xef6\xb2\xf3\x89HU~;p\x17\xdf\x00\x00\u07d4\xc2\xd1w\x8e\xf6\xee_\xe4\x88\xc1E\xf3Xkn\xbb\xe3\xfb\xb4E\x89>\x1f\xf1\xe0;U\xa8\x00\x00\xe0\x94\xc2\xd9\xee\xdb\xc9\x01\x92c\xd9\xd1l\u016e\a-
\x1d=\xd9\xdb\x03\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xc2\xe0XJq4\x8c\xc3\x14\xb7;
)\xb6#\v\x92\u06f1\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc2\xe2\u0518\xf7\r\xcd\bY\xe5\v\x02:q\nmK!3\xbd\x8989\x11\xf0\f\xbc\xe1\x00\x00\u07d4\xc2\xed_\xfd\u046d\xd8U\xa2i/\xe0b\xb5\xd6\x18t#`\u0509A\rXj
\xa4\xc0\x00\x00\u07d4\xc2\xee\x91\xd3\xefX\xc9\u0465\x89\x84N\xa1\xae1%\xd6\u017ai\x894\x95tD\xb8@\xe8\x00\x00\u07d4\xc2\xfa\xfd\xd3\n\xcbmg\x06\xe9)<\xb0&A\xf9\xed\xbe\a\xb5\x89Q\x00\x86\vC\x0fH\x00\x00\u07d4\xc2\xfd\v\xf7\xc7%\xef>\x04~Z\xe1\u009f\xe1\x8f\x12\xa7)\x9c\x89Hz\x9a0E9D\x00\x00\u07d4\xc2\xfe}us\x1fcm\xcd\t\xdb\xda\x06q9;\xa0\xc8*}\x89wC\"\x17\xe6\x83`\x00\x00\u07d4\xc3\x10z\x9a\xf32-
R8\xdf\x012A\x911b\x959W}\x89\x1a\xb4\xe4d\xd4\x141\x00\x00\xe0\x94\xc3\x11\v\xe0\x1d\xc9sL\xfcn\x1c\xe0\u007f\x87\xd7}\x13E\xb7\xe1\x8a\x01\x0f\f\xe9I\xe0\x0f\x93\x00\x00\u07d4\xc38\xcaR\xae\xe1\x97E\xbe\\1\xfc\xdcT\x14\x8b\xb2\xc4\u0409\x02\xb5\xaa\xd7,e
\x00\x00\u07d4\xc3%\xc3R\x80\x1b\xa8\x83\xb3\"l_\xeb\r\xf9\xea\xe2\xd6\xe6S\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\xc3.\xc7\xe4*\xd1l\xe3\xe2UZ\xd4\xc5C\x06\xed\xa0\xb2gX\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc32\xdfP\xb1<\x014\x90\xa5\xd7\xc7]\xbf\xa3f\u0687\xb6\u0589\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xc3:\u0373\xba\x1a\xab'P{\x86\xb1]g\xfa\xf9\x1e\xcfb\x93\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xc3>\u0393Z\x8fN\xf98\xea~\x1b\xac\x87\u02d2]\x84\x90\u028a\a\x03\x8c\x16x\x1fxH\x00\x00\u07d4\xc3@\xf9\xb9\x1c&r\x8c1\xd1!\xd5\xd6\xfc;\xb5m=\x86$\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc3F\xcb\x1f\xbc\xe2\xab(]\x8eT\x01\xf4-
\xd7#M7\xe8m\x89\x04\x86\u02d7\x99\x19\x1e\x00\x00\xe0\x94\xc3H=n\x88\xac\x1fJ\xe7<\xc4@\x8dl\x03\xab\xe0\xe4\x9d\u028a\x03\x99\x92d\x8a#\u0220\x00\x00\xe0\x94\xc3H\xfcZF\x13#\xb5{\xe3\x03\u02c96\x1b\x99\x19\x13\xdf(\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xc3N;\xa12.\xd0W\x11\x83\xa2O\x94
N\xe4\x9c\x18fA\x89\x03'\xaf\uf927\xbc\x00\x00\xe0\x94\xc3[\x95\xa2\xa3s|\xb8\xf0\xf5\x96\xb3E$\x87+\xd3\r\xa24\x8a\x01\x98\xbe\x85#^-
P\x00\x00\xe0\x94\xc3c\x1cv\x98\xb6\xc5\x11\x19\x89\xbfE''\xb3\xf99Zm\xea\x8a\x02C'X\x96d\x1d\xbe\x00\x00\u07d4\xc3l\vc\xbf\xd7\\/\x8e\xfb\x06\b\x83\xd8h\xcc\xcdl\xbd\xb4\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94\xc3uk\xcd\xcc~\xect\xed\x89j\xdf\xc35'Y0&n\b\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\u00c4\xacn\xe2|9\xe2\xf2x\xc2 \xbd\xfa[\xae\xd6&\xd9\u04c9
\x86\xac5\x10R`\x00\x00\u07d4\u00e0F\xe3\u04b2\xbfh\x14\x88\x82n2\xd9\xc0aQ\x8c\xfe\x8c\x89\x8c\xf2?\x90\x9c\x0f\xa0\x00\x00\u07d4\u00e9\"j\xe2u\xdf,\xab1+\x91\x10@cJ\x9c\x9c\x9e\xf6\

x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u00f9(\xa7o\xadex\xf0O\x05U\xe69R\xcd!\xd1R\n\x89l
k\x93[\x8b\xbd@\x00\x00\xe0\x94\xc3\xc2)s)\xa6\xfd\x99\x11~T\xfcj\xf3y\xb4\xd5VT~\x8a\x01EB\
xba\x12\xa37\xc0\x00\x00\u07d4\xc3\xc3\xc2Q\rg\x80
HZcs]\x13\a\xecL\xa60+\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc3\xcbk6\xafD?,n%\x8bJ9U:\x8
1\x87G\x81\x1f\x89WG=\x05\u06ba\xe8\x00\x00\xe0\x94\xc3\xdbVW\xbbr\xf1\rX\xf21\xfd\xdf\x11\
x98\n\xffg\x86\x93\x8a\x01@a\xb9\xd7z^\x98\x00\x00\xe0\x94\xc3\u06df\xb6\xf4lH\n\xf3De\u05d7
S\xb4\xe2\xb7Jg\u038a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xc3\xddX\x908\x860;\x92\x86%%z\x
e1\xa0\x13\xd7\x1a\xe2\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc3\xe0G\x1cd\xff5\xfaR2\xcc1!\
xd1\u04cd\x1a\x0f\xb7\u0789lk\x93[\x8b\xbd@\x00\x00\u07d4\xc3\xe2\f\x96\u07cdN8\xf5\v&Z\x9
8\xa9\x06\xd6\x1b\xc5\x1aq\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc3\u31f0<\xe9\\\xcf\xd7\xfaQ\u
0744\x01\x83\xbcCS(\t\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xc3\xf8\xf6r\x95\xa5\xcd\x04\x93d\x
d0]#P&#\xa3\xe5.\x84\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xc4\x01\xc4'\xcc\xcf\xf1\r\xec\
xb8d
/6\xf5\x80\x83\"\xa0\xa8\x89\xb4{Q\xa6\x9c\xd4\x02\x00\x00\u07d4\xc4\b\x8c\x02_>\x85\x01?T9\
xfb4@\xa1s\x01\xe5D\xfe\x89~\t\xdbM\x9f?4\x00\x00\u07d4\xc4\x14a\xa3\u03fd2\u0246UU\xa4\x
8117\xc0v1#`\x8965\xc6 G9\u0640\x00\u07d4\xc4
8\x8f\xbe\xe8J\xd6V\xddh\xcd\xc1\xfb\xaa\x93\x92x\v4\x89\n-
\xcac\xaa\xf4\u0140\x00\u07d4\xc4\"P\xb0\xfeB\xe6\xb7\xdc\xd5\u0210\xa6\xf0\u020f__\xb5t\x89\
b\x1e\xe4\x82SY\x84\x00\x00\u07d4\xc4-
j\xebq\x0e:P\xbf\xb4Ml1\t)i\xa1\x1a\xa7\xf3\x89\b\"c\xca\xfd\x8c\xea\x00\x00\xe0\x94\xc4@\xc7\xc
a/\x96Kir\xeffJ\"a\xdd\xe8\x92a\x9d\x9c\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xc4K\xde\xc8\
xc3l\\h\xba\xa2\xdd\xf1\xd41i2)rlC\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xc4OJ\xb5\xbc`9|s~\xb0h3\x91\xb63\xf8<H\xfa\x8965\u026d
\xc5\u07a0\x00\x00\u07d4\xc4R\xe0\xe4\xb3\u05ae\x06\xb86\xf02\xca\t\xdb@\x9d\xdf\xe0\xfb\x8
9+^:\xf1k\x18\x80\x00\x00\u07d4\xc4Z\x1c\xa1\x03k\x95\x00A\x87\u036cD\xa3n3\xa9J\xb5\u00c9
\r\xd0\x0fr\x03\x01\x88\x00\x00\u07d4\xc4]G\xab\f\x9a\xa9\x8a[\xd6-
\x16\">\xa2G\x1b\x12\x1c\xa4\x89 .h\xf2\u00ae\xe4\x00\x00\u07d4\xc4h\x1es\xbb\x0e2\xf6\xb7&
H1\xffi\xba\xa4\x87~2\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xc4k\xbd\xefv\xd4\xca`\xd3\x16\xc0
\u007f]\x1ax\x0e;\x16_~\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc4}a\v9\x92P\xf7\x0e\xcf\x13\x89\x
ba\xb6),\x91&O#\x89\x0f\xa7\xe7\xb5\xdf<\xd0\x00\x00\u07d4\u0100;\xb4\a\xc7b\xf9\vu\x96\xe6\x
fd\u1513\x1ev\x95\x90\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u0106Q\xc1\xd9\xc1k\xffL\x95
T\x88l??&C\x1foh\x89#\xab\x95\x99\xc4?\b\x00\x00\u07d4\u0109\xc8?\xfb\xb0%*\xc0\xdb\xe3R\x
12\x17c\x0e\x0fl\x1f\x14\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u010bi<\xac\xef\xdb\xd6\xcb]
x\x95\xa4.1\x962~&\x1c\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u0113H\x9eV\u00fd\xd8)\x00}\xc
2\xf9VA)\x06\xf7k\xfa\x89\x02\xa7\x91H\x8eqT\x00\x00\u07d4\u0116\u02f0E\x9aj\x01`\x0f\u0149\
xa5Z2\xb4T!\u007f\x9d\x89\x0e\u0683\x8cI)\b\x00\x00\u07d4\u011c\xfa\xa9g\xf3\xaf\xbfU\x03\x10
a\xfcL\xef\x88\xf8]\xa5\x84\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\u0136\xe5\xf0\x9c\xc1\xb9\r\xf0
x\x03\xce=M\x13vj\x9cF\xf4\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\u013e\xc9c\b\xa2\x0f\x
90\u02b1\x83\x99\u0113\xfd=\x06Z\xbfE\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\xe0\x94\xc4\xc0\
x1a\xfc>\x0f\x04R!\xda\x12\x84\u05c7\x85tD/\xb9\xac\x8a\x01\x92<h\x8bs\xab\x04\x00\x00\u07d
4\xc4\xc1S\x18\xd3p\xc73\x18\xcc\x18\xbd\xd4f\u06eaLf\x03\xbf\x89\x01\x11du\x9f\xfb2\x00\x00\
u07d4\xc4\xc6\xcbr=\u05ef\xa7\xebSV\x15\xe5?<\xef\x14\xf1\x81\x18\x89lk\x8f\xce\r\x18y\x80\x0

0\u07d4\xc4\xccE\xa2\xb6<'\xc0\xb4B\x9eX\xcdB\xdaY\xbes\x9b\u058965\u026d\xc5\u07a0\x00\x00\u07d4\xc4\u03d3\x0e]\x11j\xb8\xd1;\x9f\x9a~\u012bP\x03\xa6\xab\u0789\x11X\xe4`\x91=\x00\x00\x00\u07d4\xc4\xd9\x16WNh\u011f~\xf9\xd3\xd8-
\x168\xb2\xb7\xee\t\x85\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\xe0\x94\xc4\xda\u0168\xa0&O\xbc\x10U9\x1cP\x9c\xc3\xee!\xa6\xe0L\x8a\x01`k\u007f\xa09\xcet\x00\x00\u07d4\xc4\xdd\x04\x8b\xfb\x84\x0e+\xc8\\\xb5?\xcbu\xab\xc4C\xc7\xe9\x0f\x89\xc9q\xdc\a\xc9\u01d0\x00\x00\u07d4\xc4\xf2\x91;&\\C\x0f\xa1\xab\x8a\xdf&\xc33\xfc\x1d\x9bf\xf2\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc4\xf7\xb1:\xc6\xd4\xebM\xb3\xd4\xe6\xa2R\xaf\x8a\a\xbdYW\u0689\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xc4\xf7\xd2\xe2\xe2
\x84\xc4Op\xfe\xaa\xb6\xc3!\x05\xf3\xda7o\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xc4\xffo\xbb\x1f\t\xbd\x9e\x10+\xa03\xd66\xac\x1cL\x0fS\x04\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc4\xff\xad\xaa\xf2\x82?\xbe\xa7\xbf\xf7\x02\x02\x1b\xff\u0105>\xb5\u0249\x02J\x19\xc1\xbdo\x12\x80\x00\xe0\x94\xc5\x00\xb7
sN\xd2)8\u05cc^H\xb2\xba\x93g\xa5u\xba\x8a\a\x12\x9e\x1c\xdf7>\xe0\x00\x00\u07d4\xc5\x0f\xe4\x15\xa6A\xb0\x85lNu\xbf\x96\x05\x15D\x1a\xfa5\x8d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc5\x13L\xfb\xb1\xdfz
\xb0\xedpWb.\xee\u0480\x94}\xad\x89\xcd\xff\x97\xfa\xbc\xb4`\x00\x00\xe0\x94\xc5\x17\xd01\\\x87\x88\x13\xc7\x17\u132f\xa1\xea\xb2eN\x01\u068a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xc5\x18y\x9aY%Wb\x13\xe2\x18\x96\xe0S\x9a\xbb\x85\xb0Z\xe3\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc5\"\xe2\x0f\xbf\x04\xed\u007fk\x05\xa3{G\x18\xd6\xfc\xe0\x14.\x1a\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xc5$\bmF\xc8\x11+\x12\x8b/\xafo|}\x81`\xa88l\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xc5-
\x1a\fs\u00a1\xbe\x84\x91Q\x85\xf8\xb3O\xaa\n\xdf\x1d\xe3\x89K\xe4\xea\xb3\xfa\x0f\xa6\x80\x00\xe0\x94\xc55\x94\xc7\u03f2\xa0\x8f(L\xc9\u05e6;\xbd\xfc\v1\x972\x8a\nk#(\xff:b\xc0\x00\x00\u07d4\xc57I(\xcd\xf1\x93pTC\xb1L\xc2\r\xa4#G<\xd9\u03c9\a}\x10P\x9b\xb3\xaf\x80\x00\u07d4\xc58\xa0\xff(*\xaa_Ku\u03f6,p\x03~\xe6}O\xb5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc5;P\xfd;+r\xbclC\v\xaf\x19JQU\x85\u04d8m\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xc5=y\xf7\u02dbp\x95/\xd3\x0f\xceX\xd5K\x9f\vY\xf6G\x8a\x01\x13\xe2\xd6tCE\xf8\x00\x00\u07d4\xc5I\u07c3\xc6\xf6^\xec\x0f\x1d\u0260\x93J_:P\xfd\x88\x89\x9d\xc0\\\xce(\u00b8\x00\x00\u07d4\xc5P\x05\xa6\xc3~\x8c\xa7\xe5C\xce%\x99s\xa3\xca\u0396\x1aJ\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc5U\xb91V\xf0\x91\x01#<o|\xf6\xeb<O\x19m3F\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xc5ZkGa\xfd\x11\xe8\xc8_\x15\x17Mtv|\u063d\x9ah\x89\a?u\u0460\x85\xba\x00\x00\u07d4\xc5nkb\xban@\xe5*\xab\x16}!\xdf\x02]\x00UuK\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc5s\xe8A\xfa\b\x17J
\x8b\x06\f\xcb{L\rv\x97\x12\u007f\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\xc5v\x12\u0791\x11\fH.oP[\xcd#\xf3\xc5\x04}\x1da\x89\xc2\x12z\xf8X\xdap\x00\x00\u07d4\u01443\x99\xd1P\x06k\xf7\x97\x9c4\xba)F 6\x8a\xd7\xc0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\u014b\x9c\xc6\x1d\xed\xbb\x98\xc3?\"M\"\x1f\x0e\"\x8bX43\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\u014fb\xfe\xe9q\x1ej\x05\xdc\t\x10\xb6\x18B\n\xa1'\xf2\x88\x89\xd7\xc1\x98q\x0ef\xb0\x00\x00\u07d4\u0153\xb5F\xb7i\x87\x10\xa2\x05\xadF\x8b,\x13\x15\"\x19\xa3B\x89T\x06\x923\xbf\u007fx\x00\x00\xe0\x94\u0153\xd6\xe3}\x14\xb5fd:\xc4\x13_$<\xaa\a\x87\xc1\x82\x8a\x02\x8a\x85t%Fo\x80\x00\x00\xe0\x94\u0163\xb9\x8eE\x93\xfe\xa0\xb3\x8cOEZPe\xf0Q\xa2\xf8\x15\x8a\x04L\xf4h\xaf%\xbfw\x00\x00\u07d4\u0164\x8a\x85\x00\xf9\xb4\xe2/\x0e\xb1loFIhvt&}

\x89,\x0e\xc5\x03\x85\x04>\x80\x00\xe0\x94\u0166)\xa3\x96%R\u02ce\xde\u0609cj\xaf\xbd\f\x18\
xcee\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u016e\x86\xb0\xc6\xc7\xe3\x90\x0f\x13h\x
10\\VS\u007f\xaf\x8dt>\x89\n1\x06+\xee\xedp\x00\x00\u07d4\u0170\t\xba\xea\xf7\x88\xa2v\xbd5\
x81:\xd6[@\v\x84\x9f;\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u0175l\xd24&|(\xe8\x9cok\"f\xb0\x8
6\xa1/\x97\f\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xc5\u01a4\x99\x8a3\xfe\xb7dCz\x8b\xe9
)\xa7;\xa3J\ad\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\xe0\x94\xc5\xc7=a\xcc\xe7\xc8\xfeL\x8f\xce)\xf3
\x90\x92\xcd\x19>\x0f\xff\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xc5\xc7Y\vV!\xec\xf85\x85\x
88\u079bh\x90\xf2baC\U000498a1]\tQ\x9b\xe0\x00\x00\u07d4\xc5\xcd\xce\xe0\xe8]\x11}\xab\xbf
Sj?@i\xbfD?T\xe7\x89j\xc5\xc6-
\x94\x86\a\x00\x00\u07d4\xc5\u050c\xa2\xdb/\x85\xd8\xc5U\xcb\x0e\x9c\xfe\x82i6x?\x9e\x89\n\u0
5ce\xbcZ\xc6
\x00\x00\xe0\x94\xc5\xde\x12\x03\xd3\xcc,\xea1\xc8.\xe2\xdeY\x16\x88\a\x99\xea\xfd\x8a\x01\x0f
\f\xf0d\xddY \x00\x00\u07d4\xc5\xe4\x88\xcf+Vw\x939q\xf6L\xb8 -
\xd0WR\xa2\xc0\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc5\xe8\x12\xf7o\x15\xf2\xe1\xf2\xf9\xbc
H#H<\x88\x04cog\x89\x03\xf5\x14\x19:\xbb\x84\x00\x00\u07d4\xc5\u94d34\xf1%.\u04ba&\x81D\x
87\xdf\u0498+1(\x89\x03\xcbq\xf5\x1f\xc5X\x00\x00\u07d4\xc5\xebB)^\x9c\xad\xea\xf2\xaf\x12\xd
e\u078a\x8dS\xc5y\xc4i\x89\xcf\x15&@\xc5\xc80\x00\x00\xe0\x94\xc5\xed\xbb\xd2\xca\x03WeJ\x
d0\xeaG\x93\xf8\xc5\xce\xcd0\xe2T\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xc5\xf6K\xab\xb
7\x031B\xf2\x0eF\u05eab\x01\xed\x86\xf6q\x03\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc5\xf6\x87q
rF\u068a \r
\xe5\u9f2c`\xb6\u007f8a\x89\x01\x8d\x99?4\xae\xf1\x00\x00\u07d4\xc6\x04[<5\vL\xe9\xca\fkuO\x
b4\x1ai\xb9~\x99\x00\x892$\xf4'#\xd4T\x00\x00\u07d4\xc6\v\x04eN\x00;F\x83\x04\x1f\x1c\xbdk\u
00cf\xda|\xdb\u0589lk\x93[\x8b\xbd@\x00\x00\u07d4\xc6\x14F\xb7T\xc2N;\x16B\xd9\xe5\x17e\xb
4\xd3\xe4k4\xb6\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc6\x18R\x13!\xab\xaf[&Q:J\x95(\bo\"\n\xdc
o\x89\x01v\xb3D\xf2\xa7\x8c\x00\x00\u07d4\xc6#FW\xa8\a8A&\xf8\x96\x8c\xa1p\x8b\xb0{\xaaI<\
x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc6%\xf8\u024d'\xa0\x9a\x1b\u02bdQ(\xb1\u00a9HV\
xaf0\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xc65^\xc4v\x8cp\xa4\x9a\xf6\x95\x13\u0343\xa5\xbc\xa7\xe3\xb9\u034a\x01EB\
xba\x12\xa37\xc0\x00\x00\u07d4\xc6:\xc4\x17\x99.\x9f\x9b`8n\xd9S\xe6\xd7\xdf\xf2\xb0\x90\xe8\x
89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4\xc6<\u05c8!\x18\xb8\xa9\x1e\aML\x8fK\xa9\x18Q0;\x9a\x8
9\x0e\x189\x8ev\x01\x90\x00\x00\u07d4\xc6R\x87\x1d\x19$\"\u01bc#_\xa0c\xb4J~\x1dC\u3149\b
g\x0e\x9e\xc6Y\x8c\x00\x00\xe0\x94\xc6gD\x1e\u007f)y\x9a\xbaadQ\xd5;?H\x9f\x9e\x0fH\x8a\x02
\xf2\x9a\xceh\xad\u0740\x00\x00\u07d4\xc6j\xe4\xce\xe8\u007f\xb352\x19\xf7\u007f\x1dd\x86\u0
140(\x032\x89\x01\x9a\x16\xb0o\xf8\xcb\x00\x00\u07d4\xc6t\xf2\x8c\x8a\xfd\a?\x8by\x96\x91\xb2\
xf0XM\xf9B\xe8D\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\u0197\xb7\x04w\u02b4.+\x8b&f\x81\xf4\x
aesu\xbb%A\x8a\x01.W2\xba\xba\\\x98\x00\x00\u07d4\u019b\x85U9\xce\x1b\x04qG(\xee\xc2Z7\x
f3g\x95\x1d\xe7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u019b\xe4@\x13Mb\x80\x98\x01D\xa9\xf6M
\x84t\x8a7\xf3I\x89&\u009eG\u0104L\x00\x00\u07d4\u019df<\x8d`\x90\x83\x91\xc8\xd26\x19\x15
3\xfd\xf7wV\x13\x89\x1aJ\xba\"\\
t\x00\x00\u0794\u01a2\x86\xe0e\xc8_:\xf7H\x12\xed\x8b\u04e8\xce]%\xe2\x1d\x88\xfc\x93c\x92\x
80\x1c\x00\x00\u07d4\u01a3\x0e\xf5\xbb3
\xf4\r\xc5\xe9\x81#\rR\xae:\xc1\x93\"\x89\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\u07d4\u01ae(}\xdb\x

e1\x14\x9b\xa1m\xdc\xcaO\xe0j\xa2\uaa48\xa9\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xc6\xc7\xc1\x917\x98\x97\u075c\x9d\x9a3\x83\x9cJ_b\xc0\x89\r\x89\xd8\xd8T\xb2$0h\x80\x00\xe0\x94\xc6\xcdh\xec56,Z\xd8L\x82\xadN\xdc#!%\x91-
\x99\x8a\x05\xe0T\x9c\x962\xe1\xd8\x00\x00\u07d4\xc6\u0615N\x8f?\xc53\xd2\xd20\xff\x02\\\xb4\xdc\xe1O4&\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xc6\xdb\u06de\xfd^\xc1\xb3xn\x06q\xeb\"y\xb2S\xf2\x15\xed\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc6\xdf
u\xeb\xd2@\xd4Hi\u00bek\u07c2\xe6=N\xf1\xf5\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc6\xe2\xf5\xaf\x97\x9a\x03\xfdr:\x1bn\xfar\x83\x18\u03dc\x18\x00\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\xc6\xe3$\xbe\xeb[6v^\xcdFB`\xf7\xf2`\x06\xc5\xc6.\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc6\xe4\xcc\fr\x83\xfc\x1c\x85\xbcH\x13\xef\xfa\xafr\xb4\x98#\xc0\x89\x0f\x03\x1e\xc9\xc8}\xd3\x00\x00\xe0\x94\xc6\xee5\x93B)i5)\xdcA\u067bq\xa2IfX\xb8\x8e\x8a\x04+\xf0kx\xed;P\x00\x00\u07d4\xc6\xfb\x1e\xe3t\x17\u0400\xa0\xd0H\x92;\u06ba\xb0\x95\xd0w\u0189\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xc7\x05'\xd4D\u0110\xe9\xfc?\\\xc4Nf\xebO0k8\x0f\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xc7\r\x85mb\x1e\xc1E0<\nd\x00\xcd\x17\xbb\xd6\xf5\xea\xf7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xc7\x0f\xa4Uv\xbf\x9c\x86_\x988\x93\x00,AI&\xf6\x10)\x89\x15\xb4\xaa\x8e\x97\x02h\x00\x00\u07d4\xc7\x11E\xe5)\u01e7\x14\xe6y\x03\xeeb\x06\xe4\xc3\x04+g'\x89M\x85<\x8f\x89\b\x98\x00\x00\u07d4\xc7\x1b*=q5\u04a8_\xb5\xa5q\u073ei^\x13\xfcC\u034965\u026d\xc5\u07a0\x00\x00\u07d4\xc7\x1f\x1du\x87?3\u0732\xddK9\x87\xa1-
\a\x91\xa5\xce'\x897\b\xba\xed=h\x90\x00\x00\u07d4\xc7\x1f\x92\xa3\xa5J{\x8c/^\xa4C\x05\xfc\u02c4\xee\xe21H\x89\x02\xb5\x9c\xa11\xd2\x06\x00\x00\u07d4\xc7!\xb2\xa7\xaaD\xc2\x12\x98\xe8P9\xd0\x0e.F\x0eg\v\x9c\x89\a\xa1\xfe\x16\x02w\x00\x00\x00\u07d4\xc7,\xb3\x01%\x8e\x91\xbc\b\x99\x8a\x80]\u0452\xf2\\/\x9a5\x89
\t\xc5\u023fo\xdc\x00\x00\xe0\x94\xc76\x8b\x97\t\xa5\xc1\xb5\x1c\n\xdf\x18ze\xdf\x14\xe1+}\xba\x8a\x02\x02o\xc7\u007f\x03\u5b80\x00\u07d4\xc79%\x9e\u007f\x85\xf2e\x9b\xef_`\x9e\xd8k=Yl\x1e\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xc7>!\x12(\"\x15\xdc\ab\xf3+~\x80}\xcd\x1az\xae>\x8a\x01v\f\xbcb;\xb3P\x00\x00\xe0\x94\xc7If\x80B\xe7\x11#\xa6H\x97^\b\xedc\x82\xf8>\x05\xe2\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4\xc7J9\x95\xf8\a\xde\x1d\xb0\x1a.\xb9\xc6.\x97\xd0T\x8fio\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc7Pl\x10\x19\x12\x1f\xf0\x8a,\x8c\x15\x91\xa6^\xb4\xbd\xfbJ?\x89
\x86\xac5\x10R`\x00\x00\u07d4\xc7\\7\xce-
\xa0k\xbc@\b\x11Y\u01ba\x0f\x97n9\x93\xb1\x89:y#\x15\x1e\xcfX\x00\x00\u07d4\xc7]\"Y0j\xec}\xf0\"v\x8ci\x89\x9ae!\x85\xdb\u0109\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xc7`\x97\x1b\xbc\x18\x1cj|\xf7tA\xf2BG\u045c\xe9\xb4\u03c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xc7a0\xc7<\xb9!\x028\x02\\\x9d\xf9]\v\xe5J\xc6\u007f\xbe\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4\xc7e\xe0\x04v\x81\tG\x81j\xf1B\xd4m.\u7f28\xccO\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xc7g^VG\xb9\xd8\xda\xf4\xd3\xdf\xf1\xe5R\xf6\xb0qT\xac8\x89\t\xc2\x00vQ\xb2P\x00\x00\u07d4\xc7{\x01\xa6\xe9\x11\xfa\x98\x8d\x01\xa3\xab3dk\xee\xf9\xc18\xf3\x89'\x1bo\xa5\xdb\xe6\xcc\x00\x00\u07d4\u01c3z\u0420\xbf\x14\x18i7\xac\xe0lUF\xa3j\xa5OF\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u01d8\x06\x03+\xc7\xd8(\xf1\x9a\u01a6@\u018e=\x82\x0f\xa4B\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u01d9\xe3N\x88\xff\x88\xbe}\xe2\x8e\x15\xe4\xf2\xa6=\v3\xc4\u02c9\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\u01ddPb\u01d6\xddwa\xf1\xf1>U\x8ds\xa5\x9f\x82\xf3\x8b\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\u01e0\x18\xf0\x96\x8aQ\xd1\xf6`<\\I\xdcT[\xcb\x0f\xf2\x93\x89\xd8\xd7&\x

b7\x17z\x80\x00\x00\u07d4\u01ef\xf9\x19)yt\x89UZ/\xf1\xd1M\\iZ\x10\x83U\x8965\u026d\xc5\u07a0\x00\x00\u0794\u01f1\xc8>c
?\x95G&>\xf6(.}\xa3;n\xd6Y\x88\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94\u01f3\x9b\x06\x04Q\x00\f\xa1\x04\x9b\xa1T\xbc\xfa\x00\xff\x8a\xf2b\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\u01ff\x17\xc4\xc1\x1f\x98\x94\x1fP~w\bO\xff\xbd-
\xbd=\xb5\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\u01ff.\xd1\xed1)@\xeej\xde\xd1Qn&\x8eJ`HV
\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xc7\xd4O\xe3,\u007f\x8c\xd5\xf1\xa9t'\xb6\xcd:\xfc\x9eE\x02>\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\xc7\xd5\xc7\x05@\x81\xe9\x18\xech{Z\xb3n
\x97=\x18\x13)5\x89\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\u07d4\xc7\xde^\x8e\xaf\xb5\xf6+\x1a\n\xf2\x19\\\xf7\x93\u01c9L\x92h\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xc7\xe30\xcd\f\x89\n\u025f\xe7q\xfc\xc7\xe7\xb0\t\xb7A=\x8a\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xc7\xea\xc3\x1a\xbc\xe6\xd5\xf1\u07a4\"\x02\xb6\xa6t\x15=\xb4z)\x89
\t\xc5\u023fo\xdc\x00\x00\xe0\x94\xc7\xecb\xb8\x04\xb1\xf6\x9b\x1e0p\xb5\xd3b\xc6/\xb3\t\xb0p\x8a\x02\xc4k\xf5A`f\x11\x00\x00\u07d4\xc7\xf7+\xb7X\x01k7G\x14\u0509\x9b\xce\"\xb4\xae\xc7\n1\x89:&\xc9G\x8f^-
\x00\x00\u0794\xc8\v6\u047e\xaf\xba_\xccdM`\xacnF\xed)'\xe7\u0708\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\xc8\x11\xc2\xe9\xaa\x1a\xc3F.\xba^\x88\xfc\xb5\x12\x0e\x9fn,\xa2\x89K\xe6\u0607\xbd\x87n\x00\x00\u07d4\xc8\x17\xdf\x1b\x91\xfa\xf3\x0f\xe3%\x15qr|\x97\x11\xb4]\x8f\x06\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xc8\x1f\xb7\xd2\x0f\u0480\x01\x92\xf0\xaa\xc1\x98\xd6\u05a3}?\xcb}\x89\x0e\x11I3\x1c-\xde\x00\x00\u07d4\xc8
\xc7\x11\xf0w\x05'8\a\xaa\xaam\xe4M\x0eKH\xbe.\x89\bg\x0e\x9e\xc6Y\x8c\x00\x00\u07d4\xc8#\x1b\xa5\xa4\x11\xa1>\"+)\xbf\xc1\b?v1X\xf2&\x8967\tlK\xcci\x00\x00\u07d4\xc86\xe2Jo\xcf)\x94;6\b\xe6b)\n!_e)\xea\x89\x0f\xd4Pd\xea\xee\x10\x00\x00\xe0\x94\xc8;\xa6\u0755I\xbe\x1d2\x87\xa5\xa6T\xd1\x06\xc3Lk]\xa2\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4\xc8>\x9djX%;\uefb7\x93\xe6\xf2\x8b\x05JXI\x1bt\x89\x0fF\u00b6\xf5\xa9\x14\x00\x00\u07d4\xc8A\x88O\xa4x_\xb7s\xb2\x8e\x97\x15\xfa\xe9\x9aQ40]\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc8M\x9b\xea\n{\x9f\x14\x02
\xfd\x8b\x90\x97\u03ff\xd5\xed\xf5d\x89\x06\xab\x9e\u0091\xad}\x80\x00\u07d4\xc8RB\x8d+Xd\x97\xac\xd3\fV\xaa\x13\xfbU\x82\xf8D\x02\x893B\xd6\r\xff\x19`\x00\x00\u07d4\xc8S![\x9b\x9f-
,\xd0t\x1eX^\x98{_\xb8\f!.\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\xc8S%\uaca5\x9b>\xd8c\xc8j_)\x06\xa0B)\xff\xa9\x89\x19=\u007f}%=\xe0\x00\x00\u07d4\xc8^\xf2}\x82\x04\x03\x80_\xc9\xed%\x9f\xffd\xac\xb8\xd64j\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc8akN\xc0\x91(\xcd\xff9\xd6\u4e6c\x86\xee\xc4q\xd5\xf2\x89\x01\r:\xa56\xe2\x94\x00\x00\xe0\x94\xc8a\x90\x90K\x8d\a\x9e\xc0\x10\xe4b\xcb\xff\xc9\b4\xff\xaa\\\x8a\x02#\x85\xa8'\xe8\x15P\x00\x00\u07d4\xc8q\r~\x8bZ;\u059aB\xfe\x0f\xa8\xb8|5\u007f\xdd\xcd\u0209\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xc8sR\u06e5\x82\xeef\xb9\xc0\x02\xa9b\xe0\x03\x13Ox\xb1\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xc8|w\xe3\xc2J\xde\xcd\xcd\x108\xa3\x8bV\xe1\x8d\xea\u04f7\x02\x8a\x01\xdd\f\x88_\x9a\r\x80\x00\x00\u07d4\xc8}:\xe3\u0607\x04\u066b\x00\t\xdc\xc1\xa0\x06q1\xf8\xba<\x89j\xc5\xc6-
\x94\x86\a\x00\x00\xe0\x94\u0201N4R>8\xe1\xf9'\xa7\xdc\xe8FjDz\t6\x03\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u0202U\xed\xdc\xf5!\xc6\xf8\x1d\x97\xf5\xa4!\x81\xc9\a=N\xf1\x89\x0f\u00d0D\xd0\n*\x80\x00\u07d4\u0205\xa1\x8a\xab\xf4T\x1b{{~\xcd0\xf6\xfa\u619d\x95i\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u020c\xa1\xe6\xe5\xf4\xd5X\xd17\x80\xf4\x88\xf1\rJ\xd3\x13\r4\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\u020e\xecT\xd3\x05\xc9(\xcc(H\xc2\xfe\xe251\xac\xb9mI\x89lj\u

04c2\xd4\xfba\x00\x00\xe0\x94\u021c\xf5\x04\xb9\xf3\xf85\x18\x1f\xd8BO\\\xcb\xc8\xe1\xbd\xdf}\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u0222\xc4\xe5\x9e\x1c\u007f\xc5H\x05X\x048\xae\xd3\xe4J\xfd\xf0\x0e\x89\x02b\x9ff\xe0\xc50\x00\x00\u07d4\u022aI\u301f\b\x99\xf2\x8a\xb5~gCp\x9dXA\x903\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4\u022b\x1a<\xf4l\xb8\xb0d\xdf.\"-9`s\x942w\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u0231\x85\x05%\xd9F\xf2\xae\x84\xf3\x17\xb1Q\x88\xc56\xa5\u0706\x89\x91\x8d\xdc:B\xa3\xd4\x00\x00\u07d4\xc8\xd4\xe1Y\x9d\x03\xb7\x98\t\xe0\x13\n\x8d\u00c4\b\xf0^\x8c\u04c9\x9f\xad\x06$\x12y\x16\x00\x00\u07d4\xc8\xdd'\xf1k\xf2$P\xf5w\x1b\x9f\xe4\xedO\xfc\xb3\t6\xf4\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xc8\xdezVL\u007f@\x12\xa6\xf6\xd1\x0f\u040fG\x89\x0f\xbf\a\u0509\x10CV\x1a\x88)0\x00\x00\u07d4\xc8\xe2\xad\xebT^I\x9d\x98,\f\x11sc\u03b4\x89\u0171\x1f\x895e\x9e\xf9?\x0f\xc4\x00\x00\xe0\x94\xc8\xe5X\xa3\xc5i~o\xb2:%\x94\u0200\xb7\xa1\xb6\x8f\x98`\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xc8\xf2\xb3
\xe6\xdf\xd7\t\x06\u0157\xba\xd2\xf9P\x13\x12\u01c2Y\x89Q\x93K\x8b:W\xd0\x00\x00\u07d4\xc9\x03\x00\xcb\x1d@w\xe6\xa6\xd7\xe1i\xa4`F\x8c\xf4\xa4\x92\u05c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xc9\f7e\x15k\u028eH\x97\xab\x80$\x19\x15<\xbeR%\xa9\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xc9\x10\xa9pUl\x97\x16\xeaS\xaff\xdd\xef\x93\x141$\x91=\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\xe0\x94\xc9\x12{\u007ff)\xee\x13\xfc?`\xbc/Dg\xa2\aE\xa7b\x8a\x03|\x9a\xa4\xe7\xceB\x1d\x80\x00\u07d4\xc9\x1b\xb5b\xe4+\xd4a0\xe2\u04eeFR\xb6\xa4\ub1bc\x0f\x89\x1dF\x01b\xf5\x16\xf0\x00\x00\xe0\x94\xc90\x88y\x05m\xfe\x13\x8e\xf8
\x8fy\xa9\x15\u01bc~p\xa8\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\xc94\xbe\xca\xf7\x1f\"_\x8bJK\xf7\xb1\x97\xf4\xac\x9604\\\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xc9?\xbd\xe8\xd4m+\xcc\x0f\xa9\xb3;\u063a\u007f\x80B\x12Ue\x89K\xe4\xe7&{j\xe0\x00\x00\u07d4\xc9@\x89U:\xe4\xc2,\xa0\x9f\xbc\x98\xf5pu\xcf.\u0155\x04\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xc9A\x10\xe7\x1a\xfeW\x8a\xa2\x18\xe4\xfc(d\x03\xb03\n\u038d\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xc9F\u056c\xc14n\xba\nry\xa0\xac\x1dF\\\x99m\x82~\x8a\x03x=T_\xdf\n\xa4\x00\x00\u07d4\xc9J(\xfb20\xa9\xdd\xfa\x96Nw\x0f,\xe3\xc2S\xa7\xbeO\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xc9JXR\x03\xda{\xba\xfd\x93\xe1X\x84\xe6`\u0531\xea\xd8T\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4\xc9O|5\xc0'\xd4}\xf8\xefO\x9d\xf8Z\x92H\xa1}\xd2;\x89\x01\x9f\x8euY\x92L\x00\x00\u07d4\xc9Q\x90\f4\x1a\xbb\xb3\xba\xfb\xf7\xee
)7pq\xdb\xc3j\x89\x11\xc2]\x00M\x01\xf8\x00\x00\u07d4\xc9S\xf94\xc0\xeb-
\x0f\x14K\u06b0\x04\x83\xfd\x81\x94\x86\\\xe7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc9f&r\x8a\xaaLO\xb3\xd3\x1c&\xdf:\xf3\x10\b\x17\x10\u0449\xb5\x0f\u03ef\xeb\xec\xb0\x00\x00\u07d4\xc9gQel\n\x8e\xf45{sD2!4\xb9\x83PJ\u0289lk\x93[\x8b\xbd@\x00\x00\u07d4\u0240Hh\u007f+\xfc\u027d\x90\xed\x18slW\xed\xd3R\xb6]\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u0241\xd3\x12\u0487\xd5X\x87\x1e\u0757:\xbbv\xb9y\xe5\xc3^\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\u0242Xmc\xb0\xd7L\x1b\x1a\xf8A\x83r\xe3\fv\x16\xbe\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\u0249CO\x82Z\xaf\x9cU/h^\xba|\x11\xdbJ_\xc7:\x89\x1b(\u014d\x96\x96\xb4\x00\x00\u07d4\u0249\xee\xc3\a\u80db\x9dr7\xcf\xda\b\x82)b\xab\u41c9\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\u0252\xbeY\xc6r\x1c\xafN\x02\x8f\x9e\x8f\x05\xc2\\UQ[\u0509\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u0255{\xa9L\x1b)\xe5'~\xc3f\"pI\x04\xc6=\xc0#\x89h>\xfcg\x82d,\x00\x00\xe0\x94\u025a\x9c\xd6\xc9\xc1\xbe54\xee\u0352\xec\xc2/\\8\xe9Q[\x8a\x01\x05Y;:\x16\x9dw\x00\x00\xe0\x94\u026c\x01\xc3\xfb\t)\x03?\f\xcc~\x1a\xcf\uaae7\x94]G\x8a\x02\xa3j\x9e\x9c\xa4\xd2\x0

3\x80\x00\u07d4\u0276\x98\xe8\x98\xd2\rMO@\x8eNM\x06\x19\"\xaa\x85c\a\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\u0276\xb6\x86\x11\x16\x91\xeej\xa1\x97\xc7#\x1a\x88\xdc`\xbd)]\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xc9\u01ec\v\u0753B\xb5\xea\xd46\t#\xf6\x8cr\xa6\xbac:\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xc9\xc8\r\xc1.{\xab\x86\xe9I\xd0\x1eL>\xd3_+\x9b\xba_\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xc9\xd7dF\u056a\xdf\xf8\vh\xb9\x1b\b\u035b\xc8\xf5U\x1a\xc1\x89&\xb4\xbd\x91\x10\xdc\xe8\x00\x00\xe0\x94\xc9\u073b\x05oM\xb7\xd9\xda9\x93b\x02\u017d\x820\xb3\xb4w\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xc9\xe0&\b\x06h(\x84\x8a\xeb(\xc76r\xa1)%\x18\x1fM\x89\x1b\x1bk\u05efd\xc7\x00\x00\u07d4\xca\x042\xcb\x15{Qy\xf0.\xbb\xa5\xc9\u0475O\xecM\x88\u028965\u026d\xc5\u07a0\x00\x00\u07d4\xca\x12,\xf0\U00094216\xb7HC\xf4\x9a\xfe\u043a\x16\x18\xee\u05c9\x1e[\x8f\xa8\xfe*\xc0\x00\x00\xe0\x94\xca\"\u0363`m\xa5\xca\xd0\x13\xb8\aG\x06\xd7\xe9\xe7!\xa5\f\x8a\x01q\x81\xc6\xfa9\x81\x94\x00\x00\u07d4\xca#\xf6-\xff\rd`\x03lb\xe8@\xae\xc5W~\v\xef\u0489\a\xa1\xfe\x16\x02w\x00\x00\x00\u07d4\xca%\xff4\x93L\x19B\xe2*N{\xd5o\x14\x02\x1a\x1a\xf0\x88\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xca7?\xe3\xc9\x06\xb8\xc6U\x9e\xe4\x9c\xcd\a\xf3|\xd4\xfbRf\x89a\t=|,m8\x00\x00\u07d4\xcaA\u032c0\x17
R\xd5\"\xcd//\x95}$\x81S@\x9f\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xcaB\x88\x01N\xdd\xc5c/_\xac\xb5\xe3\x85\x17\xa8\xf8\xbc]\x98\x89\x12nr\xa6\x9aP\xd0\x00\x00\u07d4\xcaB\x88c\xa5\xca06\x98\x92\xd6\x12\x18>\xf9\xfb\x1a\x04\xbc\xea\x89Rf<\u02b1\xe1\xc0\x00\x00\u07d4\xcaI\xa5\xf5\x8a\xdb\xef\xae#\xeeY\xee\xa2A\xcf\x04\x82b.\xaa\x89M\x85<\x8f\x89\b\x98\x00\x00\u07d4\xcaL\xa9\xe4w\x9dS\x0e\u02ec\xd4~j\x80X\xcf\xdee\u064f\x89+^:\xf1k\x18\x80\x00\x00\u07d4\xcae~\xc0o\xe5\xbc\t\xcf#\xe5*\xf7\xf8\f\xc3h\x9en\u07890\xca\x02O\x98{\x90\x00\x00\u07d4\xcaf\xb2(\x0f\xa2\x82\u0176v1\xceU+b\xeeU\xad\x84t\x89j\xc4\"\xf54\x92\x88\x00\x00\xe0\x94\xcal\x81\x8b\xef\xd2Q6\x1e\x02t@h\xbe\x99\u062a`\xb8J\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xcap\xf4\u077f\x06\x9d!C\xbdk\xbc\u007fikRx\x9b2\u7262\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94\xcatuvDjL\x8f0\xb0\x83@\xfe\xe1\x98\xdec\xec\x92\u03ca\x01|\x8e\x12\x06r*0\x00\x00\u07d4\xca{\xa3\xffSl~_\x0e\x158\x00\xbd8=\xb81)\x98\xe0\x89\t1\xac=k\xb2@\x00\x00\xe0\x94\u0282v\xc4w\xb4\xa0{\x80\x10{\x845\x94\x18\x96\a\xb5;\xec\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\u0284\t\b>\x01\xb3\x97\xcf\x12\x92\x8a\x05\xb6\x84U\xceb\x01\u07c9V\xbcu\xe2\xd61\x00\x00\x00\u07d4\u0298\u01d8\x8e\xfa\b\xe9%\uf719ER\x03&\xe9\xf4;\x99\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u029a\x04*j\x80o\xfc\x92\x17\x95\x00\xd2D)\xe8\xabR\x81\x17\x89;\xa1\x91\v\xf3A\xb0\x00\x00\u07d4\u029d\xec\x02\x84\x1a\xdf\\\xc9
WjQ\x87\xed\u04bdCJ\x18\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\u029f\xaa\x17T/\xaf\xbb8\x8e\xab!\xbcL\x94\u89f3G\x88\x89lk\x8f\xce\r\x18y\x80\x00\xe0\x94\u02aah\xeel\xdf\r4EJv\x9b\r\xa1H\xa1\xfa\xaa\x18e\x8a\x01\x87.\x1d\xe7\xfeR\xc0\x00\x00\u07d4\u02ad\x9d\xc2\rX\x9c\xe4(\xd8\xfd\xa3\xa9\xd5:`{y\x88\xb5\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\u02b0\xd3,\xf3v\u007f\xa6\xb3S|\x842\x8b\xaa\x9fPE\x816\x8a\x01\xe5\xb8\xfa\x8f\xe2\xac\x00\x00\x00\u07d4\u02b9\xa3\x01\xe6\xbdF\xe9@5P(\xec\xcd@\xceMZ\x1a\u00c9\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\u02b9\xa9z\xda\x06\\\x87\x81nh`\xa8\xf1Bo\xe6\xb3\xd7u\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\u02ba\xb6'N\xd1P\x89s~({\xe8x\xb7W\x93Hd\xe2\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\u02bd\xaf5OG
\xa4f\xa7d\xa5(\xd6\x0e:H*9<\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xca\xcbg^\t\x96#T\x04\ufbfb.\u02c1R'\x1bU\xe0\x89%\xf2s\x93=\xb5p\x00\x00\u07d4\xca\xd1O\x9e\xbb\xa7f\x80\xeb\x83k\a

\x9c\u007f{\xaa\xf4\x81\xedm\x89\f\xef={\xd7\xd04\x00\x00\xe0\x94\xca\xe3\xa2S\xbc\xb2\xcfN\x13\xba\x80\u0098\xab\x04\x02\xda|*\xa0\x8a\x01$\xbc\r\u0752\xe5`\x00\x00\u07d4\xca\xef\x02{\x1a\xb5\x04\xc7?A\xf2\xa1\ty\xb4t\xf9~0\x9f\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xca\xf4H\x1d\x9d\xb7\x8d\xc4\xf2_{J\u023d;\x1c\xa0\x10k1\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x94\xca\xfd\xe8U\x86L%\x98\xda<\xaf\xc0Z\u064d\U00089380H\x8a\x03\x00\xa8\xed\x96\xffJ\x94\x00\x00\xe0\x94\xcb\r\xd7\xcfN]\x86a\xf6\x02\x89C\xa4\xb9\xb7\\\x91D6\xa7\x8a\x19i6\x89t\xc0[\x00\x00\x00\u07d4\xcb\x1b\xb6\xf1\xda^\xb1\rH\x99\xf7\xe6\x1d\x06\xc1\xb0\x0f\u07f5-
\x898E$\xccp\xb7x\x00\x00\u07d4\xcb=vl\x98?\x19+\xce\xca\xc7\x0fN\xe0=\xd9\xffqMQ\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xcbB\xb4N\xb5\xfd`\xb5\x83~O\x9e\xb4rgR=\x1a\"\x9c\x89.\xe4IU\b\x98\xe4
\x00\x00\u07d4\xcbG\xbd0\u03e8\xecTh\xaa\xa6\xa9FB\xce\xd9\xc8\x19\xc8\u0509\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xcbH\xfe\x82e\u066fU\xebp\x06\xbc3VE\xb0\xa3\xa1\x83\xbe\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xcbJ\x91M+\xb0)\xf3._\xef\\#LO\xec--\xd5w\x89a\x94\x04\x9f0\xf7
\x00\x00\xe0\x94\xcbJ\xbf\u0082\xae\xd7n]W\xaf\xfd\xa5B\xc1\xf3\x82\xfc\xac\xf4\x8a\x01\xb9\x0f\x11\xc3\x18?\xaa\x00\x00\u07d4\xcbJ\xd0\xc7#\xdaF\xabV\xd5&\xda\f\x1d%\xc7=\xaf\xf1\n\x89\x1b\xa5\xab\xf9\xe7y8\x00\x00\u07d4\xcbK\xb1\xc6#\xba(\xdcB\xbd\xaa\xa6\xe7N\x1d*\xa1%l*\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xcbPXt\x12\x82#\x04\xeb\u02e0}\xab:\x0f\t\xff\xfe\u4189JD\x91\xbdm\xcd(\x00\x00\u07d4\xcbX\x99\v\u0350\u03ffm\x8f\t\x86\xf6\xfa`\x02v\xb9N-
\x8964\xbf9\xab\x98x\x80\x00\u07d4\xcbh\xaeZ\xbe\x02\xdc\xf8\xcb\u016aq\x9c%\x81FQ\xaf\x8b\x85\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xcbty\x10\x9bC\xb2fW\xf4F_M\x18\xc6\xf9t\xbe_B\x89b\xa9\x92\xe5:\n\xf0\x00\x00\xe0\x94\xcb}+\x80\x89\xe91,\u026e\xaa's\xf3S\b\xecl*{\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u02c6\xed\xbc\x8b\xbb\x1f\x911\x02+\xe6IV^\xbd\xb0\x9e2\xa1\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u02d3\x19\x9b\x9c\x90\xbcI\x15\xbd\x85\x9e=B\x86m\xc8\xc1\x87I\x89\f\x90\xdf\a\xde\xf7\x8c\x00\x00\u07d4\u02d4\xe7o\xeb\xe2\b\x11g3\xe7n\x80]H\xd1\x12\xec\x9f\u028965\u026d\xc5\u07a0\x00\x00\u07d4\u02dbQ\x03\xe4\u0389\xafOd\x91aP\xbf\xf9\xee\u02df\xaa\\\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\u02e2\\zP<\xc8\xe0\xd0Iq\xca\x05\xc7b\xf9\xb7b\xb4\x8b\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\u02e2\x88\xcd<\x1e\xb4\u055d\xdb\x06\xa6B\x1c\x14\xc3E\xa4{$\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u02f3\x18\x9eK\xd7\xf4_\x17\x8b\x1c0\xc7n&1MJK\n\x89\x0f\xfe\vg|e\xa9\x80\x00\xe0\x94\u02f7\xbe\x17\x95?,\u0313\u1f19\x80[\xf4U\x11CNL\x8a\n\xae[\x9d\xf5m/
\x00\x00\xe0\x94\xcb\xc0KM\x8b\x82\xca\xf6p\x99o\x16\f6)@\xd6o\xcf\x1a\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xcb\u07974\xb8\xe6\xaaS\x8c)\x1dm\u007f\xac\xed\xb0\xf38\xf8W\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xcb\xe1\xb9H\x86M\x84t\xe7e\x14XX\xfc\xa4U\x0fxK\x92\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xcb\xe5/\xc53\xd7\xdd`\x8c\x92\xa2`\xb3|?E\u07b4\xeb3\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xcb\xe8\x10\xfe\x0f\xec\xc9dGJ\x1d\xb9w(\xbc\x87\xe9s\xfc\xbd\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xcb\xf1j\x0f\xe2tRX\xcdR\xdb+\xf2\x19T\xc9u\xfcj\x15\x89\x10CV\x1a\x88)0\x00\x00\xe0\x94\xcb\xf3\u007f\xf8T\xa2\xf1\xceS\x93D\x94wx\x92\xd3\xeceW\x82\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xcb\xfaj\xf6\u0083\xb0F\xe2w,`c\xb0\xb2\x15S\xc4\x01\x06\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xcb\xfav\xdb\x04\xce8\xfb

]7\xb8\xd3w\xcf\x13\x80\xda\x03\x17\x89M\x85<\x8f\x89\b\x98\x00\x00\u07d4\xcc\x03I\x85\xd3\xf2\x8c-
9\xb1\xa3K\xce\xd4\u04f2\xb6\xca#N\x89\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\u07d4\xcc\x04<C\x88\xd3E\xf8\x84\u0185^q\x14*\x9fA\xfdi5\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xcc\x1dn\xad\x01\xaa\xda>\x8d\u01f9]\xca%\xdf&\xee\xfac\x9d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xcc+_D\x8f5(\xd3\xfeA\xcc}\x1f\xa9\xc0\xdcv\xf1\xb7v\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\xcc-
\x04\xf0\xa4\x01q\x89\xb3@\xcaw\x19\x86A\xdc\xf6Ek\x91\x89\u0556{\xe4\xfc?\x10\x00\x00\xe0\x94\xccA\x9f\u0651+\x85\x13VY\xe7z\x93\xbc=\xf1\x82\xd4Q\x15\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xccE\xfb:U[\xad\x80{8\x8a\x03W\xc8U
_|u\xe8\x89.\xe4IU\b\x98\xe4\x00\x00\u07d4\xccHAM*\xc4\xd4*Yb\xf2\x9e\xeeD\x97\t/C\x13R\x89\b\xbaR\xe6\xfcE\xe4\x00\x00\u07d4\xccJ/,\xf8l\xf3\xe43u\xf3`\xa4sF\x91\x19_\x14\x90\x89I\x15\x05;\xd1)\t\x80\x00\u07d4\xccO\x0f\xf2\xae\xb6}T\xce;\xc8\xc6Q\v\x9a\xe8>\x9d2\x8b\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xccO\xaa\xc0\v\xe6b\x8f\x92\xefk\x8c\xb1\xb1\xe7j\xac\x81\xfa\x18\x89\v\"\xa2\xea\xb0\xf0\xfd\x00\x00\xe0\x94\xccO\xebr\u07d8\xff5\xa18\xe0\x17a\xd1
?\x9b~\xdf\n\x8a\x01{x\x83\xc0i\x16`\x00\x00\u07d4\xcc`oQ\x13\x97\xa3\x8f\u01c7+\u04f0\xbd\x03\xc7\x1b\xbdv\x8b\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xcc`\xf86\xac\xde\xf3T\x8a\x1f\xef\u0321>\u01a97\xdbD\xa0\x89\x04\xb0m\xbb\xb4\x0fJ\x00\x00\u07d4\xccl\x03\xbd`>\t\xdeT\xe9\xc4\u056cmA\xcb\xceqW$\x89\x05V\xf6L\x1f\xe7\xfa\x00\x00\u07d4\xccl-
\xf0\x0e\x86\xec\xa4\x0f!\xff\xda\x1ag\xa1i\x0fG|e\x89\xabM\xcf9\x9a:`\x00\x00\xe0\x94\xccm{\x12\x06\x1b\xc9m\x10M`me\xff\xa3+\x006\xeb\a\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xccs\xdd5kIy\xb5y\xb4\x01\xd4\xccz1\xa2h\xdd\xceZ\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xccu\x8d\a\x1d%\xa62\n\xf6\x8c]\xc9\xc4\xf6\x95[\xa9E
\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xcc{\x04\x81\xcc2\xe6\xfa\xef#\x86\xa0p\"\xbc\xb6\xd2\u00f4\xfc\x89\xabM\xcf9\x9a:`\x00\x00\xe0\x94\u0314;\xe1\",\xd1@\n#\x99\xdd\x1bE\x94E\xcfmT\xa9\x8a\x02\xa7@\xaee6\xfc\x88\x00\x00\u07d4\u0315\x19\xd1\xf3\x98_k%^\xad\xed\x12\xd5bJ\x97'!\xe1\x8965\u026d\xc5\u07a0\x00\x00\u0794\u031a\xc7\x15\xcdo&\x10\xc5+XgdV\x88B\x97\x01\x8b)\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\u0320{\xb7\x94W\x1dJ\xcf\x04\x1d\xad\x87\xf0\xd1\xef1\x85\xb3\x19\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u032b\xc6\x04\x8aSFD$\xfc\xf7n\xeb\x9en\x18\x01\xfa#\u0509\x02\xab{&\x0f\xf3\xfd\x00\x00\u07d4\u032e\r=\x85*}\xa3\x86\x0f\x066\x15L\nl\xa3\x16(\u0509\x05\xc6\xd1+k\xc1\xa0\x00\x00\u07d4\xcc\xca$\xd8\xc5mn,\a\xdb\bn\xc0~X[\xe2g\xac\x8d\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xcc\xd5!\x13-
\x98l\xb9hi\x84&\"\xa7\u0762l>\xd0W\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xcc\xf49u\xb7k\xfes_\xec<\xb7\xd4\xdd$\xf8\x05\xba\tb\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\xcc\xf6*f?\x13S\xba.\xf8\xe6R\x1d\xc1\xec\xb6s\xec\x8e\xf7\x89\b=lz\xabc`\x00\x00\u07d4\xcc\xf7\x11\r\x1b\u0667K\xfd\x1d}}-
\x9dU`~{\x83}\x890\xca\x02O\x98{\x90\x00\x00\u07d4\xcc\xfdrW`\xa6\x88#\xff\x1e\x06/L\xc9~\x13`\xe8\u0657\x89\x15\xacV\xed\xc4\xd1,\x00\x00\u07d4\xcd\x02\x0f\x8e\xdf\xcfRG\x98\xa9\xb7:d\x034\xbb\xf7/\x80\xa5\x89\a?u\u0460\x85\xba\x00\x00\u07d4\xcd\x06\xf8\xc1\xb5\u037d(\xe2\xd9kcF\xc3\xe8Z\x04\x83\xba$\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xcd\a.n\x183\x13y\x95\x19m{\xb1r_\xef\x87a\xf6U\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xcd\n\x16\x1b\xc3g\xae\t'\xa9*\xac\x9c\xf6\xe5\bg\x14\xef\u0289lk\x93[\x8b\xbd@\x00\x00\u07d4\xcd\n\xf3GN\"\xf0i\xec4\a\x87\r\xd7pD=[\x12\xb0\x89\x8e^\xb4\xeew\xb2\xef\x00\x00\u07d4\xcd\v\x02W\u70e3\xd2\xc2\u3e9dny\

xb7^\xf9\x80$\u0509\x9f\xad\x06$\x12y\x16\x00\x00\u07d4\xcd\x10,\xd6\xdb=\xf1J\u05af\x0f\x87\xc7$y\x86\x1b\xfc=$\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xcd\x1ef\xedS\x9d\xd9/\xc4\v\xba\xa1\xfa\x16\u078c\x02\xc1ME\x89\fw\xe4%hc\xd8\x00\x00\u07d4\xcd\x1e\xd2c\xfb\xf6\xf6\xf7\xb4\x8a\xef\x8fs=2\x9dC\x82\xc7\u01c9\x01\x00\xbd3\xfb\x98\xba\x00\x00\u07d4\xcd*6\xd7S\xe9\xe0\xed\x01*XMqh\aX{A\xd5j\x89\x0e+\xa7[\v\x1f\x1c\x00\x00\u07d4\xcd2\xa4\xa8\xa2\u007f\x1c\xc69T\xaacOxW\x05s4\u01e3\x89:\xd1fWlr\xd4\x00\x00\u07d4\xcd5\xff\x01\x0e\xc5\x01\xa7!\xa1\xb2\xf0z\x9c\xa5\x87}\xfc\xf9Z\x89\xd9o\u0390\u03eb\xcc\x00\x00\u07d4\xcdC\x06\xd7\xf6\x94z\xc1tMN\x13\xb8\xef2\xcbe~\x1c\x00\x89\x1b\x1a\xb3\x19\xf5\xecu\x00\x00\u07d4\xcdC%\x8bs\x92\xa90\x83\x9aQ\xb2\xef\x8a\xd24\x12\xf7Z\x9f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xcdI\xbf\x18^p\xd0E\a\x99\x9f\x92\xa4\xdeDU1('\u040965\u026d\xc5\u07a0\x00\x00\u07d4\xcdU\x10\xa2B\u07f0\x18=\xe9%\xfb\xa8f\xe3\x12\xfa\xbc\x16W\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4\xcdVj\u05f8\x83\xf0\x1f\u04d9\x8a\x9aX\xa9\xde\xe4rM\u0725\x89\x030\xae\x185\xbe0\x00\x00\xe0\x94\xcdY\xf3\xdd\xe7~\t\x94\v\xef\xb6\xeeX\x03\x19e\xca\xe7\xa36\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xcdr]p\xbe\x97\xe6w\xe3\xc8\xe8\\\v&\xef1\xe9\x95PE\x89Hz\x9a0E9D\x00\x00\xe0\x94\xcd~G\x90\x94d\xd8q\xb9\xa6\xdcv\xa8\xe9\x19]\xb3H^z\x8a\x02\x15\xf85\xbcv\x9d\xa8\x00\x00\u07d4\xcd~\xce\bkKa\x9b;6\x93R\xee8\xb7\x1d\xdb\x06C\x9a\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xcd\u007f\t\xd7\xedf\xd0\u00cb\u016dN2\xb7\xf2\xb0\x8d\xc1\xb3\r\x89>;\xb3M\xa2\xa4p\x00\x00\u07d4\u0355)I+\\)\xe4u\xac\xb9A@+=;\xa5\x06\x86\xb0\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\u0355\xfaB=o\xc1 'J\xac\xde\x19\xf4\xee\xb7f\xf1\x04 \x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\u035bL\xefs9\f\x83\xa8\xfdq\u05f5@\xa7\xf9\u03cb\x8c\x92\x89\x04\xe1\x00;(\xd9(\x00\x00\u07d4\u0361t\x11\t\xc0&[?\xb2\xbf\x8d^\xc9\u00b8\xa34kc\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u0361\xb8\x86\u39d5\u027aw\x91N\n/\xe5go\x0f\\\u03c9\x05\xbf`\xeaB\xc2\x04\x00\x00\u07d4\u0364S\x0fK\x9b\xc5\t\x05\xb7\x9d\x17\u008f\xc4o\x954\x9b\u07c93\x10\xe0I\x11\xf1\xf8\x00\x00\u07d4\u036bF\xa5\x90 \x80do\xbf\x95B\x04
J\xe8\x84\x04\x82+\x89\x1d\x8a\x96\xe5\xc6\x06\xeb\x00\x00\u07d4\u0375\x97)\x900\x18?n-
#\x853\xf4d*\xa5\x87T\xb6\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xcd\xd5\u0601\xa76,\x90p\a;\u07fcu\xe7$S\xacQ\x0e\x89-
\xa5\x18\xea\xe4\x8e\xe8\x00\x00\u07d4\xcd\xd6\rs\xef\xaa\xd8s\u027b\xfb\x17\x8c\xa1\xb7\x10Z\x81\xa6\x81\x89\x01\xbc\x16\xd6t\xec\x80\x00\x00\u07d4\xcd\xd9\xef\xacMm`\xbdq\xd9U\x85\xdc\xe5\u0557\x05\xc15d\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xcd\xe3m\x81\xd1(\u015d\xa1Ee!\x93\xee\u00bf\xd9e\x86\xef\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xcd\xea8o\x9d\x0f\xd8\x04\xd0(\x18\xf27\xb7\xd9\xfavF\xd3^\x89\xa3I\xd3m\x80\xecW\x80\x00\u07d4\xcd\xec\xf5gT3\u0370\xc2\xe5Zh\xdb]\x8b\xbexA\x9d\u0489\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xcd\xfd\x82\x173\x97%\xd7\xeb\xac\x11\xa66U\xf2e\xef\xf1\xcc=\x8a\x01\x0f\fid\x10\xe3\xa9\x00\x00\u07d4\xce\a\x9fQ\x88wt\xd8\x02\x1c\xb3\xb5u\xf5\x8f\x18\xe9\xac\xf9\x84\x89\t\xc2\x00vQ\xb2P\x00\x00\u07d4\xce\x18\x84\u077b\xb8\xe1\x0eM\xbanD\xfe\xee\u00a7\xe5\xf9/\x05\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xce\x1b\f\xb4j\xae\xcf\u05db\x88\f\xad\x0f-
\u068a\x8d\xed\u0431\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xce&\xf9\xa50_\x83\x81\tCT\xdb\xfc\x92fN\x84\xf9\x02\xb5\x89\fz\xaa\xb0Y\x1e\xec\x00\x00\u07d4\xce-
\xea\xb5\x1c\n\x9a\xe0\x9c\xd2\x12\xc4\xfaL\xc5+S\xcc\r\xec\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xce.\r\xa8\x93F\x99\xbb\x1aU>U\xa0\xb8\\\x16\x945\xbe\xa3\x8a\x01\x0f\fid\x10\xe3\xa9\x00\x

00\u07d4\xce:a\xf0F\x1b\x00\x93^\x85\xfa\x1e\xad\x82\xc4^Zd\u0508\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xceK\x06]\xbc\xb20G
2b\xfbH\xc1\x18\x83d\x97tp\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xceS\xc8\xcd\xd7B\x96\xac\xa9\x87\xb2\xbc\x19\u00b8u\xa4\x87I\u0409\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xce^\x04\xf0\x18Ci\xbc\xfa\x06\xac\xa6o\xfa\x91\xbfY\xfa\x0f\xb9\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xce^\xb6:{\xf4\xfb\xc2\xf6\u4ea0\u018a\xb1\xcbL\xf9\x8f\xb4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xceb\x12Z\xde\xc37\n\xc5!\x10\x95:Nv\v\xe9E\x1e;\x89\b=lz\xabc`\x00\x00\xe0\x94\xceq\bmL`%T\xb8-\xcb\xfc\xe8\x8d
cMS\xccM\x8a\t(\x96R\x9b\xad\u0708\x00\x00\u07d4\u038akmP3\xb1I\x8b\x1f\xfe\xb4\x1aAU\x04\x05\xfa\x03\xa2\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u0397\x86\xd3q/\xa2\x00\xe9\xf6\x857\xee\xaa\x1a\x06\xa6\xf4ZK\x89a\t=|,m8\x00\x00\u07d4\u039d!\u0192\xcd<\x01\xf2\x01\x1fP_\x87\x006\xfa\x8fl\u0489\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\u03a2\x89f#\xf4\x91\x02\x87\xa2\xbd\u017e\x83\xae\xa3\xf2\xe6\xde\b\x8a\x01\xfbZ7Q\xe4\x90\xdc\x00\x00\u07d4\u03a3JM\xd9=\u066e\xfd9\x90\x02\xa9}\x99z\x1bK\x89\u0349QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4\u03a4?pu\x81k`\xbb\xfc\u62d9:\xf0\x88\x12p\xf6\u0109lk\x93[\x8b\xbd@\x00\x00\u07d4\u03a8t3AS<\xb2\xf0\xb9\xc6\xef\xb8\xfd\xa8\rw\x16(%\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\u03b0\x89\xec\x8ax3~\x8e\xf8\x8d\xe1\x1bI\xe3\u0751\x0ft\x8f\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u03b3=x\xe7Tz\x9d\xa2\xe8}Q\xae\xc5\xf3D\x1c\x87\x92:\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u03b3\x898\x1dH\xa8\xaeO\xfcH:\u043b^
L\xfd\xb1\xec\x89('\xe6\xe4\xddb\xba\x80\x00\u07d4\xce\xc6\xfce\x85?\x9c\xce_\x8e\x84Fv6.\x15y\x01_\x02\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xce\xd3\u01fe\x8d\xe7XQ@\x95*\xebP\x1d\xc1\xf8v\ucbf0\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xce\xd8\x1e\xc3S?\xf1\xbf\xeb\xf3\xe3\x84>\xe7@\xad\x11u\x8d>\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xce\u0733\xa1\u0584?\xb6\xbe\xf6Ca}\xea\U000cf398\xdd_\x89\x19\xe2\xa4\xc8\x18\xb9\x06\x00\x00\u07d4\xce\xe6\x99\xc0pzx6%+)/\x04|\xe8\xad(\x9b/U\x89\x11\x9a\x1e!\xaaiV\x00\x00\u07d4\xce\xedG\xca[\x89\x9f\xd1b?!\xe9\xbdM\xb6Z\x10\u5c1d\x89\a8w@L\x1e\xee\x00\x00\u07d4\xce\xf7tQ\u07e2\xc6C\xe0\v\x15mlo\xf8N#s\xebf\x89\n1\x06+\xee\xedp\x00\x00\u07d4\xcf\x11i\x04\x1c\x17E\xe4[\x17$5\xa2\xfc\x99\xb4\x9a\xce+\x00\x89\x01\xbb\x88\xba\xab-
|\x00\x00\xe0\x94\xcf\x15v\x12vN\x0f\u0596\xc8\xcb_\xba\x85\xdfL\r\xdc<\xb0\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u0794\xcf\x1b\xdby\x9b.\xa6<\xe14f\x8b\xdc\x19\x8bT\x84\x0f\x18\v\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xcf\"\x88\xefN\xbf\x88\xe8m\xb1=\x8a\x0e\v\xf5*\x05e\x82\u00c9\x89Po\xbf\x97@t\x00\x00\u07d4\xcf&Ni%\x13\t\x06\xc4\xd7\xc1\x85\x91\xaaA\xb2\xa6\u007foX\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xcf&\xb4{\xd04\xbcP\x8elK\xcf\xd6\xc7\xd3\x004\x92Wa\x89a\x94\x04\x9f0\xf7
\x00\x00\xe0\x94\xcf.*\xd65\xe9\x86\x1a\xe9\\\xb9\xba\xfc\xca\x03kR\x81\xf5\u038a\at2!~h6\x00\x00\x00\u07d4\xcf.s@B\xa3U\xd0_\xfb.9\x15\xb1h\x11\xf4Zi^\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xcf4\x8f/\xe4{~A<\az{\xaf:u\xfb\xf8B\x86\x92\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xcf?\x91(\xb0r\x03\xa3\xe1\r}WU\xc0\u012b\xc6\xe2\xca\u008a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xcf?\xbf\xa1\xfd2\u05e6\xe0\xe6\xf8\xefN\xabW\xbe4\x02\\L\x899\xa1\xc0\xf7YMH\x00\x00\u07d4\xcfAftn\x1d;\xc1\xf8\xd0qK\x01\xf1~\x8ab\xdf\x14d\x896w\x03n\xdf\n\xf6\x00\x00\u07d4\xcfO\x118\xf1\xbdk\xf5\xb6\u0505\xcc\xe4\xc1\x01\u007f\u02c5\xf0}\x89/\u043cw\xc3+\xff\x00\x00\u07d4\xcfZo\x9d\xf7Uy\xc6D\xf7\x94q\x12\x15\xb3\rw\xa0\xce@\x89lk\x93[\x8b\xbd@\x

00\x00\u07d4\xcf^\x0e\xac\u0473\x9d\x06U\xf2\xf7u5\xeff\b\xeb\x95\v\xa0\x89lk\x93[\x8b\xbd@\x
00\x00\u07d4\xcfhM\xfb\x83\x04r\x93U\xb5\x83\x15\xe8\x01\x9b\x1a\xa2\xad\x1b\xac\x89\x17r$\x
aa\x84Lr\x00\x00\u07d4\xcfi@\x81\xc7m\x18\xc6L\xa7\x13\x82\xbe\\\xd6;<\xb4v\xf8\x8965\u026d\
xc5\u07a0\x00\x00\u07d4\xcfnR\xe6\xb7t\x80\xb1\x86~\xfe\xc6Dm\x9f\xc3\xcc5w\xe8\x89\f\t\x01\
xf6\xbd\x98y\x00\x00\u07d4\u03c8:
2\x96g\xea\"j\x1e<v]\xbbk\xab2!\x9f\x89\xa4\xbe5d\xd6\x16f\x00\x00\xe0\x94\u03c8\x825\x9c\x0f\
xb23\x87\xf5g@t\u0631z\xdeQ/\x98\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\u03c9\xf7F\v\xa
3\xdf\xe8<Z\x1d:\x01\x9e\xe1%\x0f$/\x0f\x895h\x13\xcd\xce\xfd\x02\x80\x00\u07d4\u03d2:]\x8f\xb
c=\x01\xaa\a\x9d\x1c\xfeKC\xce\a\x1b\x16\x11\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u03db\u9e6b\
x86\xc6kY\x96\x8eg\xb8\xd4\xdc\xffF\xb1\x81J\x89#\xc7W\a+\x8d\xd0\x00\x00\u07d4\u03e8\xb3q
'\x14\x9b\xdb\xfe\xe2\\4\xd8xQ\tQ\xea\x10\xeb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u03ec.\x1b\xf
32\x05\xb0U3i\x1a\x02&~\xe1\x9c\xd8\x186\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u03fb2\xb7\x
d0$5\x0e3!\xfa
\u0269\x14\x03Sr\xff\u0189\x15\xbeat\xe1\x91.\x00\x00\u07d4\xcf\xc4\xe6\xf7\xf8\xb0\x11AK\xfb\
xa4/#\xad\xfa\xa7\x8dN\xcc^\x89dI\xe8NG\xa8\xa8\x00\x00\u07d4\xcf\xd2r\x8d\xfb\x8b\xdb\xf3\xb
fsY\x8an\x13\xea\xf40R\xea+\x89\t79SM(h\x00\x00\u07d4\xcf\xd4t\x93\xc9\xf8\x9f\u603d\xa5uM\
xd7\xc9\xcf\xe7\xcb[\xbe\x89\x02\xf4sQ4H\xfe\x00\x00\u07d4\xcf\xde\x0f\xc7]o\x16\xc4C\xc3\x03\
x82\x177-
\x99\xf5\xd9\a\xf7\x89\x83\"^c\x96\xb5\xec\x00\x00\xe0\x94\xcf\xe2\u02af<\xec\x97\x06\x1d\t9t\x8
79\xbf\xfehJ\xe9\x1f\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xcf\xea\u02ae\xd5r\x85\xe
0\xacrh\xcejN5\xec\xfd\xb2B\u05c9:\xe4\xd4$\x01\x90`\x00\x00\u07d4\xcf\ucfa0|'\x00/e\xfeSK\xb8
\x84-
\t%\u01c4\x02\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xcf\xee\x05\u019d\x1f)\xe7qF\x84\u020
d\xe5\xa1`\x98\xe9\x13\x99\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xcf\xf6\xa6\xfe>\x9a\x92*\x12\x
f2\x1f\xaa\x03\x81V\x91\x8cO\u02dc\x89\x04E\x91\xd6\u007f\xec\xc8\x00\x00\u07d4\xcf\xf7\xf8\x
9aMB\x19\xa3\x82\x95%\x131V\x82\x10\xff\xc1\xc14\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\xcf
\xf8\xd0k\x00\xe3\xf5\f\x19\x10\x99\xadV\xbaj\xe2eq\u0348\x8965\u026d\xc5\u07a0\x00\x00\xe0\
x94\xcf\xfcI\xc1x~\ubcb5l\xab\xe9$\x04\xb66\x14}EX\x8a\x013\xe00\x8f@\xa3\u0680\x00\u07d4\x
d0\bQ;'`J\x89\xba\x17c\xb6\xf8L\u6233F\x94[\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd0\x0f\x06
r\x86\xc0\xfb\u0402\xf9\xf4\xa6\x10\x83\xecv\u07b3\xce\xe6\x8965\u026d\xc5\u07a0\x00\x00\u07
d4\xd0\x15\xf6\xfc\xb8M\xf7\xbbA\x0e\x8c\x8f\x04\x89J\x88\x1d\xca\xc27\x898E$\xccp\xb7x\x00\
x00\u07d4\xd0\x1a\xf9\x13O\xafRW\x17N\x8by\x18oB\xee5Nd-
\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd0!\b\u04ae<\xab\x10\xcb\xcf\x16W\xaf\">\x02|\x82\x10\
xf6\x89lm\x84\xbc\xcd\xd9\xce\x00\x00\u07d4\xd0*\xfe\u03ce.\u00b6*\u022d
Aa\xfd\x1f\xaew\x1d\x0e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd01\x919\xfb\xab.\x8e*\xcc\xc1\xd
9$\u0531\x1d\xf6ilZ\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd07\xd2\x15\xd1\x1d\x1d\xf3\xd5O\xbd2\x1c\u0495\xc5F^';\x89K\xe4\xe7&{j\xe0\
x00\x00\u07d4\xd0:-
\xa4\x1e\x86\x8e\xd3\xfe\xf5t[\x96\xf5\xec\xa4b\xffo\u0689\xa2\xa1]\tQ\x9b\xe0\x00\x00\xe0\x94\x
d0?\xc1eWj\xae\xd5%\xe5P,\x8e\x14\x0f\x8b.\x86\x969\x8a\x01sV\u0633%\x01\xc8\x00\x00\u07d
4\xd0C\xa0\x11\xecBp\xee~\u0239hsu\x15\xe5\x03\xf80(\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u
07d4\xd0K\x86\x1b=\x9a\xccV:\x90\x16\x89\x94\x1a\xb1\xe1\x86\x11a\xa2\x89\x01\x15\x8eF\t\x1

3\xd0\x00\x00\u07d4\xd0ZD|\x91\x1d\xbb'[\xfb.Z7\xe5\xa7\x03\xa5o\x99\x97\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xd0_\xfb+t\xf8g
O\xe51e;\x02H\xe2\x1c\x13TN\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd0bX\x81q\u03d9\xbb\xebX\xf1&\xb8p\xf9\xa3r\x8da\xec\x89\xf3\xf2\v\x8d\xfai\xd0\x00\x00\u07d4\xd0c\x8e\xa5q\x89\xa6\xa6\x99\x02J\u05ccq\xd99\xc1\xc2\xff\x8c\x89\x8e\xaeVg\x10\xfc
\x00\x00\xe0\x94\xd0d\x8aX\x1b5\b\xe15\xa2\x93]\x12\xc9epE\xd8q\u028a\x01\xb2\u07dd!\x9fW\x98\x00\x00\u07d4\xd0q\x19)f\xebi\xc3R\x0f\xca:\xa4\xdd\x04)~\xa0KN\x89\x05\xf6\x8e\x811\xec\xf8\x00\x00\u07d4\xd0q\x85 \xea\xe0\xa4\xd6-
p\xde\x1b\xe0\xcaC\x1c^\xea$\x82\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd0w]\xba*\xf4\xc3\n:x6Y9\xcdq\xc2\xf9\u0795\u0489i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xd0{\xe0\xf9\t\x97\xca\xf9\x03\u022c\x1dS\xcd\xe9\x04\xfb\x19\aA\x8968\x908\xb6\x99\xb4\x00\x00\u07d4\xd0~Q\x18d\xb1\u03d9i\xe3V\x06\x02\x82\x9e2\xfcNq\xf5\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\u0400\x94\x98\xc5H\x04z\x1e**\xa6\xa2\x9c\xd6\x1a\x0e\xe2h\xbd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u0402'_tZ,\xac\x02v\xfb\xdb\x02\u0532\xa3\xab\x17\x11\xfe\x89\x01\xa0Ui\r\x9d\xb8\x00\x00\u07d4\u040f\xc0\x9a\x000\xfd\t(\xcd2\x11\x98X\x01\x82\xa7j\xae\x9f\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u0413\xe8)\x81\x9f\xd2\xe2[\x978\x00\xbb=XA\xdd\x15-
\x05\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u0414J\xa1\x85\xa13pa\xae
\u071d\xd9l\x83\xb2\xbaF\x02\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\u0416V[|t\a\xd0e6X\x03U\xfd\xd6\xd29\x14J\xa1\x89\r\x8drkqw\xa8\x00\x00\u07d4\u041c\xb2\xe6\b-
i:\x13\xe8\xd2\xf6\x8d\xd1\u0744a\xf5X@\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u0426\xc6\xf9\xe9\u0133\x83\xd7\x16\xb3\x1d\xe7\x8dVAM\xe8\xfa\x91\x89\x10CV\x1a\x88)0\x00\x00\u07d4\u0427
\x9b\x80\xcf`\xdbb\xf5}\n]}R\x1ai`fU\x89\b\xacr0H\x9e\x80\x00\x00\xe0\x94\u0428\xab\xd8\n\x19\x9bT\xb0\x8be\xf0\x1d
\x9c'\xfe\xf0\x11[\x8a\x01a\xc6&\xdca\xa2\xef\x80\x00\xe0\x94\u042b\xccp\xc0B\x0e\x0e\x17/\x97\xd4;\x87\xd5\xe8\f3n\xa9\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u042es]\x91^\x94hf\xe1\xfe\xa7~^\xa4f\xb5\xca\xdd\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u0431\x1do+\u0394^\fjP\u00f5'S\xf8\x03\xf9\u0449\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xd0\xc1\x01\xfd\x1f\x01\xc6?k\x1d\x19\xbc\x92\r\x9f\x93#\x14\xb16\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xd0\xc5Z\xbf\x97o\xdc=\xb2\xaf\u9f99\u0519HMWl\x02\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd0\u0422\xadE\xf5\x9a\x9d\xcc\u0195\xd8_%\xcaF\xed1\xa5\xa3\x89-\x89W}}@
\x00\x00\u07d4\xd0\xd6,G\xea`\xfb\x90\xa3c\x92\t\xbb\xfd\xd4\xd93\x99\x1c\u0189\n\x84Jt$\xd9\xc8\x00\x00\u07d4\xd0\xdbEax
o\\D0\xfe\x00Pc\x90<=zI\xa7\x89&I\x1eE\xa7S\xc0\x80\x00\u07d4\xd0\xe1\x94\xf3K\x1d\xb6\t(\x85\t\xcc\xd2\xe7;a1\xa2S\x8b\x8965f3\xeb\xd8\xea\x00\x00\u07d4\xd0\xe3^\x04vF\xe7Y\xf4Qp\x93\xd6@\x86BQ\u007f\bM\x89\u054f\xa4h\x18\xec\u02c0\x00\u07d4\xd0\xeeM\x02\xcf$8,0\x90\xd3\xe9\x95`\xde6xs\\\u07c9\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4\xd0\xf0OR\x10\x9a\xeb\xec\x9a{\x1e\x932v\x1e\x9f\xe2\xb9{\xb5\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xd0\xf9Yx\x11\xb0\xb9\x92\xbb}7W\xaa%\xb4\xc2V\x1d2\xe2\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xd1\x03\x02\xfa\xa1\x92\x9a2i\x04\xd3v\xbf\v\x8d\xc9:\xd0LL\x89a\t=|,m8\x00\x00\xe0\x94\xd1\x10\r\xd0\x0f\x

e2\xdd\xf1\x81c\xad\x96M\vi\xf1\xf2\xe9e\x8a\x8a\x01C\x12\tU\xb2Pk\x00\x00\u07d4\xd1\x16\xf3\xdc\xd5\xdbtK\xd0\b\x88v\x87\xaa\x0e\xc9\xfdr\x92\xaa\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd1\x19A|Fs,\xf3M\x1a\x1a\xfby\xc3\xe7\xe2\u034e\xec\xe4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd1-w\xae\x01\xa9-
5\x11{\xacpZ\xac\u0642\xd0.t\xc1\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xd15yK\x14\x9a\x18\xe1G\xd1nb\x1ai1\xf0\xa4\n\x96\x9a\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xd1C%8\xe3[vd\x95j\u4563*\xbd\xf0A\xa7\xa2\x1c\x8a\x04+\xf0kx\xed;P\x00\x00\u07d4\xd1C\x82g#\x17\x04\xfcr\x80\xd5c\xad\xf4v8D\xa8\a\"\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd1S\x8e\x9a\x87\u5729\xec\x8eX&\xa5\xb7\x93\xf9\x9f\x96\xc4\u00c965\u026d\xc5\u07a0\x00\x00\xe0\x94\xd1d\x85\x03\xb1\xcc\u0178\xbe\x03\xfa\x1e\xc4\xf3\xee&~j\xdf{\x8a\x01;\xef\xbfQ\xee\xc0\x90\x00\x00\xe0\x94\xd1h,!Y\x01\x8d\xc3\xd0\u007f\b$\n\x8c`m\xafe\xf8\xe1\x8a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4\xd1q\xc3\xf2%\x8a\xef5\xe5\x99\xc7\xda\x1a\xa0s\x00#M\xa9\xa6\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd1w\x8c\x13\xfb\xd9h\xbc\b<\xb7\xd1\x02O\xfe\x1fl\xd0,\xaa\x89\xd9\xec\xb4\xfd
\x8eP\x00\x00\u07d4\xd1\u007f\xbe\"\xd9\x04b\xed7(\x06p\xa2\xea\v0\x86\xa0\xd6\u0589\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4\u0441\x1cU\x97i\x80\xf0\x83\x90\x1d\x8a\r\xb2i\"-
\xfb\\\xfe\x89T\x06\x923\xbf\u007fx\x00\x00\u07d4\u044e\xb9\xe1\u0485\u06be\x93\xe5\u053a\xe7k\xee\xfeC\xb5!\xe8\x89$=M\x18\"\x9c\xa2\x00\x00\u07d4\u0453\xe5\x83\xd6\a\x05c\xe7\xb8b\xb9aJG\u9509\xf3\xe5\x8965f3\xeb\xd8\xea\x00\x00\u07d4\u0457\x8f.4@\u007f\xab\x1d\xc2\x18=\x95\xcf\xdab`\xb3Y\x82\x89*\xb7\xb2`\xff?\xd0\x00\x00\u07d4\u045c\xaf9\xbb7\u007f\xdf,\xf1\x9b\xd4\xfbRY\x1c&1\xa6<\x8965\u026d\xc5\u07a0\x00\x00\u0794\u0463\x96\xdc\u06b2\xc7IA0\xb3\xfd0x 4\r\xfd\x8c\x1f\x88\xf9\"P\xe2\xdf\xd0\x00\x00\xe0\x94\u0467\x1b-
\bX\xe82p\b]\x95\xa3\xb1T\x96P\x03^#\x8a\x03'\xbb\t\xd0j\xa8P\x00\x00\u07d4\u046c\xb5\xad\xc1\x189s%\x8dk\x85$\xff\xa2\x8f\xfe\xb2=\xe3\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u0473\u007f\x03\xcb\x10t$\xe9\xc4\xddW\\\xcdOL\xeeW\xe6\u0349lk\x93[\x8b\xbd@\x00\x00\u07d4\u0475\xa4T\xac4\x05\xbbAy
\x8cl\x84\xde\x00k\u02db\xe9\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xd1\xc4YT\xa6+\x91\x1a\xd7\x01\xff.\x90\x13\x1e\x8c\xeb\x89\xc9\\\x89K\x91\xa2\xdeE~\x88\x00\x00\u07d4\xd1\xc9np\xf0Z\xe0\xe6\xcd`!\xb2\b7P\xa7q|\xdeV\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xd1\u0571\u007f\xfe-
{\xbby\xcc}y0\xbc\xb2\xe5\x18\xfb\x1b\xbf\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xd1\xda\f\x8f\xb7\xc2\x10\xe0\xf2\xeca\x8f\x85\xbd\xae}>sK\x1c\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xd1\xddy\xfb\x15\x81`\xe5\xb4\xe8\xe2?1.j\x90\u007f\xbcMN\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xd1\xdeZ\xad:_\xd8\x03\U00071bb6\x10<\xb8\xe1O\xe7#\xb7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xd1\xe1\xf2\xb9\xc1l0\x98t\xde\xe7\xfa\xc3&u\xaf\xf1)\u00d8\x89\x03\xf2M\x8eJ\x00p\x00\x00\xe0\x94\xd1\xe5\xe24\xa9\xf4Bf\xa4\xa6$\x1a\x84\u05e1\xa5Z\u0567\xfe\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xd1\xeaMr\xa6{[>\x0f1UY\xf5+\xd0aMq0i\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd1\xee\x90YW\xfe|\xc7\x0e\xc8\xf2\x86\x8bC\xfeG\xb1?\xeb\xff\x89\x02b\x9ff\xe0\xc50\x00\x00\u07d4\xd1\xf1iM\"g\x1bZ\xadj\x94\x99\\6\x9f\xbea3go\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd1\xf4\xdc\x1d\u06ca\xbb\x88H\xa8\xb1N%\xf3\xb5Z\x85\x91\xc2f\x89\r\x8drkqw\xa8\x00\x00\u07d4\xd1\xfe\u042e\xe6\xf5\xdf\xd7\xe2Wi%L<\xfa\xd1Z\xde\u032a\x89'\x92\xc8\xfcKS(\x00\x00\u07d4\xd2\x05\x1c\xb3\xcbg\x04\xf0T\x8c\u0210\xab\n\x19\xdb4\x15\xb4*\x89\x12\x1b.^ddx\x00\x

00\u07d4\xd2\x06\xaa\u07736\xd4^yr\xe9<\xb0uG\x1d\x15\x89{]\x89
\x86\xac5\x10R`\x00\x00\u07d4\xd2\tH+\xb5I\xab\xc4w{\xeam\u007fe\x00b\xc9\xc5z\x1c\x89\x11
e\x1a\xc3\xe7\xa7X\x00\x00\u07d4\xd2\r\xcb\vxh+\x94\xbc0\x00(\x14H\xd5W\xa2\v\xfc\x83\x890\
x84\x9e\xbe\x166\x9c\x00\x00\u07d4\xd2\x10{57&\u00e2\xb4ef\xea\xa7\xd9\xf8\v]!\xdb\xe3\x89\x
01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xd2\x11\xb2\x1f\x1b\x12\xb5\ta\x81Y\r\xe0~\xf8\x1a\x89S~\
xad\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd2\x18\xef\xb4\u06d8\x1c\xddjy\u007fK\u050c|&)<\xeb
@\x89\xa1Fk1\xc6C\x1c\x00\x00\xe0\x94\xd2\x1asA\xeb\x84\xfd\x15\x10T\xe5\u31fb%\xd3nI\x9c\
t\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\xe0\x94\xd2$\xf8\x80\xf9G\x9a\x89\xd3/\t\xe5+\u9432\x8
8\x13\\\xef\x8a\x03\xa9\u057a\xa4\xab\xf1\xd0\x00\x00\u07d4\xd2/\f\xa4\xcdG\x9ef\x17u\x05;\xccI
\xe3\x90\xf6p\u074a\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd21\x92\x975\x13!\x02G\x1b\xa5\x9
0\a\xb6dL\xc0\xc1\xde>\x8967\tlK\xcci\x00\x00\u07d4\xd25\xd1\\\xb5\xec\xee\xbba)\x9e\x0e\x82\
u007f\xa8'H\x91\x1d\x89\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xd2:$\xd7\xf9F\x83C\xc1C\x
a4\x1ds\xb8\x8f|\xbec\xbe^\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd2=z\xff\xac\xdc>\x9f=\xaez\xfc\xb4\x00oX\xf8\xa4F\x00\x89\xc3(\t>a\xee@\x00\
x00\u07d4\xd2C\x18L\x80\x1e]y\xd2\x06?5x\u06ee\x81\u7ce9\u02c9k\u0722h\x1e\x1a\xba\x00\x
00\u07d4\xd2KfD\xf49\xc8\x05\x1d\xfcd\u04c1\xb8\xc8lu\xc1u8\x89lk\x93[\x8b\xbd@\x00\x00\xe0
\x94\xd2K\xf1--\xdfE}\xec\xb1xt\xef\xde
R\xb6\\\xbbI\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\xe0\x94\xd2Q\xf9\x03\xae\x18rrY\xee\xe8A\x
a1\x89\xa1\xf5i\xa5\xfdv\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xd2R\x96\v\v\xf6\xb2\x
84\x8f\u07ad\x80\x13m\xb5\xf5\a\xf8\xbe\x02\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xd2X\x1aU\x
ce#\xab\x10\u062d\x8cD7\x8fY\a\x9b\xd6\xf6X\x8a\x01\xdd\f\x88_\x9a\r\x80\x00\x00\u07d4\xd2Z\
xec\xd7\xeb\x8b\xd64[\x06;]\xbd'\x1cw\xd3QD\x94\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xd2|#O
\xf7\xac\xca\xce=\x99g\b\xf8\xf9\xb0Ip\xf9}6\x89Hz\x9a0E9D\x00\x00\u07d4\u0482\x98RM\xf5\xe
cK$\xb0\xff\xb9\u07c5\x17\n\x14Z\x9e\xb5\x89\x0f\x98\xa3\xb9\xb37\xe2\x00\x00\xe0\x94\u0483\
xb8\xed\xb1\n%R\x8aD\x04\xde\x1ce\xe7A\r\xbc\xaag\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\u
0484\xa5\x03\x82\xf8:am9\xb8\xa9\xc0\xf3\x96\xe0\ubfe9]\x8966\xc2^f\xec\xe7\x00\x00\u07d4\u0
488\xe7\xcb{\xa9\xf6
\xab\x0ftR\xe5\bc=\x1cZ\xa2v\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\u049d\xc0\x8e\xfb\xb3
\xd7.&?x\xabv\x10\xd0\"m\xe7k\x00\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\u04a00\xac\x89R2
_\x9e\x1d\xb3x\xa7\x14\x85\xa2N\x1b\a\xb2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u04a4y@CG\xc
5T:\xab)*\xe1\xbbJo\x15\x83W\xfa\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\u04a5\xa0$#\nW\
xcc\xc6fv\v\x89\xb0\xe2l\xaf\u0449\u01ca\n\x96YZ\\n\x8a?\x80\x00\u07d4\u04a8\x03'\xcb\xe5\\L{\
xd5\x1f\xf9\xdd\xe4\xcad\x8f\x9e\xb3\xf8\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\u04a8O
ug\\b\xd8\f\x88ulB\x8e\xee+\xcb\x18T!\x89A\rXj
\xa4\xc0\x00\x00\u07d4\u04ab\xd8J\x18\x10\x93\xe5\xe2)\x13oB\xd85\xe8#]\xe1\t\x89\x05k\xe0<\
xa3\xe4}\x80\x00\u07d4\u04ac\r:X`^\x1d\x0f\x0e\xb3\xde%\xb2\xca\xd1)\xed`X\x89\xd8\xd7&\xb7\
x17z\x80\x00\x00\u07d4\u04bfg\xa7\xf3\xc6\xceV\xb7\xbeAg]\xbb\xad\xfe~\xa9:3\x89\x15\xaf\x1d
x\xb5\x8c@\x00\x00\u07d4\xd2\xdb\xeb\xe8\x9b\x03W\xae\xa9\x8b\xbe\x8eIc8\u07bb(\xe8\x05\x
89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xd2\xe2\x1e\xd5hh\xfa\xb2\x8e\tG\x92z\xda\xf2\x9f#\xe
b\xadl\x89l\x18O\x13U\xd0\xe8\x00\x00\u07d4\xd2\xe8\x17s\x8a\xbf\x1f\xb4\x86X?\x80\xc3P1\x8
b\xed\x86\f\x80\x89\r\x02\xce\xcf_]\x81\x00\x00\u07d4\xd2\xed\xd1\xdd\xd6\xd8m\xc0\x05\xba\xe
bT\x1d\"\xb6@\xd5\xc7\xca\xe5\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xd2\xf1\x99\x8e\x1c\x

b1X\f\xecOl\x04}\xcd=\xce\xc5L\xf7<\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd2\xf2A%]\xd7\xc3\xf7<\a\x040q\xec\b\xdd\xd9\xc5\xcd\xe5\x89\x1b\x1a\xe4\xd6
\xe2\xefP\x00\x00\u07d4\xd2\xffg
\x16\xf6;/\x859\x8fJo\xed\xbb`\xa5\r<\u0389\x12\x91$o[sJ\x00\x00\u07d4\xd3\rLC\xad\xcfU\xb2\xc
bS\u0583#&A4I\x8d\x89\u038965\u026d\xc5\u07a0\x00\x00\u07d4\xd3\x0e\xe9\xa1+Mh\xab\xac\
xe6\xba\u029a\u05ff\\\xd1\xfa\xf9\x1c\x89QO\xcb$\xff\x9cP\x00\x00\u07d4\xd3\x11\x8e\xa3\xc85\
x05\xa9\u0613\xbbg\xe2\xde\x14-
Sz>\xe7\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xd3\x11\xbc\u05eaN\x9bO8?\xf3\xd0\u05b6\x
e0~!\xe3p]\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd3\x15\xde\xea\x1d\x8c\x12q\xf9\xd11\x12c\xabG\xc0\a\xaf\xb6\xf5\x89\x03\xc8\
x1dNeK@\x00\x00\u07d4\xd3+,y\xc3dx\xc5C\x19\x01\xf6\xd7\x00\xb0M\xbe\x9b\x88\x10\x89\x15
w\x9a\x9d\xe6\xee\xb0\x00\x00\u07d4\xd3+EVF\x14Ql\x91\xb0\u007f\xa9\xf7-
\xcfx|\xceN\x1c\x89\x0f\xc6o\xae7F\xac\x00\x00\u07d4\xd30r\x811\xfe\x8e:\x15Hz4W<\x93E~*\xf
e\x95\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xd31\xc8#\x82Z\x9eRc\xd0R\u0611]M\xcd\xe0z\
\7\x89\x1e\x93\x12\x83\xcc\xc8P\x00\x00\u07d4\xd33btE\xf2\u05c7\x90\x1e\xf3;\xb2\xa8\xa3g^'\xf
f\xec\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xd3<\xf8+\xf1LY&@\xa0\x86\b\x91L#py\u057e4\
x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd3Mp\x8ds\x98\x02E3\xa5\xa2\xb20\x9b\x19\xd3\xc5Qq\xb
b\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xd3N\x03\xd3j+\xd4\u045a_\xa1b\x18\xd1\xd6\x1e?\
xfa\v\x15\x89\x11X\xe4`\x91=\x00\x00\x00\u07d4\xd3Pu\xcaa\xfeY\xd1#\x96\x9c6\xa8-
\x1a\xb2\xd9\x18\xaa8\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\xd3g\x00\x9a\xb6X&;b\xc23:\x1c\x9
eA@I\x8e\x13\x89\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd3g\x9aG\xdf-
\x99\xa4\x9b\x01\u024d\x1c>\f\x98|\xe1\xe1X\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\xe0\x94\u04cf\xa2\xc4\xcc\x14z\xd0j\u0562\xf7Uy(\x1f\"\xa7\xcc\x1f\x
8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\u04da]\xa4`9+\x94\v<i\xbc\x03u{\xf3\xf2\xe8$\x89\x8a\x
01|\x83\xa9}kl\xa5\x00\x00\u07d4\u04db|\xbc\x94\x00?\xc9H\xf0\xcd\xe2{\x10\r\xb8\xcc\xd6\xe0c\
x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\u04e1\x0e\u01e5\xc92I\x99\u075e\x9bk\xde|\x91\x1e
XK\u0689
\x86\xac5\x10R`\x00\x00\u07d4\u04e9A\xc9a\xe8\u028b\x10p\xf2<mm\r*u\x8aDD\x89\n\u05ce\xb
cZ\xc6 \x00\x00\u07d4\u04fbY\xfa1%\x8b\xe6/\x8e\xd22\xf1\xa7\xd4{J\vA\xee\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\u04fcs\t7\xfau\xd8E&\x16\xad\x1e\xf1\xfe\u007f\xff\xe0\xd0\xe7\x89\x04\x84\
xe4\xde\xd2\xea\xe3\x80\x00\u07d4\xd3\xc2MK:^\x0f\xf8\xa4b-
Q\x8e\xdds\xf1j\xb2\x86\x10\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xd3\xc6\xf1\xe0\xf5\x0e
\xc3\u04a6~k\xcd\x19>\u01ee8\xf1e\u007f\x8a\x01f\xc5H\b\x89\xdbw\x00\x00\xe0\x94\xd3\xd6\xe
9\xfb\x82T/\u049e\xd9\xea6\t\x89\x1e\x15\x13\x96\xb6\xf7\x8a\voX\x8a\xa7\xbc\xf5\xc0\x00\x00\x
e0\x94\xd3\xda\u0476\u040dE\x81\u032ee\xa8s-
\xb6\xaci\xf0\u019e\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xd3\xdf;S\xcb;GU\xdeT\xe1\x80
E\x1c\xc4L\x9e\x8a\u0a89#\u0114\t\xb9w\x82\x80\x00\u07d4\xd3\xf8s\xbd\x99V\x13W\x89\xab\x
00\xeb\xc1\x95\xb9\"\xe9K%\x9d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd4\x02\xb4\xf6\xa0\x99\xe
b\xe7\x16\xcb\x14\xdfOy\xc0\xcd\x01\xc6\a\x1b\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xd4\r\x00U
\xfd\x9a8H\x8a\xff\x92?\xd0=5\xecF\xd7\x11\xb3\x8a\x01\x0f\b\xed\xa8\xe5U\t\x80\x00\u07d4\xd4
\x0e\xd6j\xb3\xce\xff$\xca\x05\xec\xd4q\ufd12\xc1__\xfa\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u
07d4\xd4\x18\x87\v\xc2\xe4\xfa{\x8aa!\xae\br\xd5RG\xb6%\x01\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x

00\u07d4\xd4\x1d\u007f\xb4\x9f\xe7\x01\xba\xac%qpBl\u0273\x8c\xa3\xa9\xb2\x89\t\x8a}\x9b\x83\x14\xc0\x00\x00\u07d4\xd4
U\x92\x84@U\xb3\u01e1\xf8\f\xef\xe3\xb8\xebP\x9b\xcd\xe7\x89\t\xb3\xbf\xd3B\xa9\xfc\x80\x00\u07d4\xd4+
\xbd\x03\x11`\x8bf\xf8\xa6\xd1[*\x95\xe6\xde'\u017f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd44O}\\\xade\xd1~\\-
\x0es#\x94=ob\xfe\x92\x89\x0e~\xeb\xa3A\vt\x00\x00\u07d4\xd4>\xe48\xd8=\xe9\xa3ub\xbbN(l\xb1\xbd\x19\xf4\x96M\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd4C4\xb4\xe2:\x16\x9a\f\x16\xbd!\xe8f\xbb\xa5-
\x97\x05\x87\x89\x8c\xf2?\x90\x9c\x0f\xa0\x00\x00\xe0\x94\xd4M\x81\xe1\x8fF\xe2\u03f5\xc1\xfc\xf5\x04\x1b\xc8V\x97g\xd1\x00\x8a\a\xb4B\xe6\x84\xf6Z\xa4\x00\x00\u07d4\xd4OJ\xc5\xfa\xd7k\xdc\x157\xa3\xb3\xafdr1\x9bA\r\x9d\x89V\xbcu\xe2\xd61\x00\x00\x00\u07d4\xd4O^\xdf+\xcf$3\xf2\x11\xda\xdd\f\xc4P\xdb\x1b\x00\x8e\x14\x89\x0e~\xeb\xa3A\vt\x00\x00\xe0\x94\xd4Oj\u00d2;_\xd71\xa4\xc4YD\xecO~\xc5*j\xe4\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xd4[3A\xe8\xf1\\\x802\x93
\u00d7~;\x90\xe7\x82j~\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xd4]]\xaa\x13\x8d\xd1\xd3t\xc7\x1b\x90\x19\x91h\x11\xf4\xb2\nN\x89\x1f9\x9b\x148\xa1\x00\x00\x00\u07d4\xd4`\xa4\xb9\b\xdd+\x05gY\xb4\x88\x85\vf\xa88\xfcw\xa8\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xd4g\xcf\x06L\bq\x98\x9b\x90\u0632\xeb\x14\xcc\xc6;6\b#\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd4k\xaea\xb0'\xe5\xbbB.\x83\xa3\xf9\xc9?<\x8f\xc7}'\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd4o\x82#E)\x82\xa1\xee\xa0\x19\xa8\x81n\xfc-
o\xc0\ah\x89\amA\xc6$\x94\x84\x00\x00\u07d4\xd4uG\u007f\xa5c\x90\xd30\x17Q\x8dg\x11\x02\u007f\x05\U0008dfc9k\x11\x133\xd4\xfdL\x00\x00\u07d4\xd4|$.\xdf\xfe\xa0\x91\xbcT\xd5}\xf5\xd1\xfd\xb91\x01Gl\x89\x9d\xf7\u07e8\xf7`H\x00\x00\u07d4\xd4}\x86\x85\xfa\xee\x14|R\x0f\u0646p\x91u\xbf/\x88k\xef\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd4\u007fP\u07c9\xa1\xcf\xf9e\x13\xbe\xf1\xb2\xae:)q\xac\xcf,\x89-\x89W}}@
\x00\x00\u07d4\u0502\xe7\xf6\x8eA\xf28\xfeQx)\xde\x15G\u007f\xe0\xf6\xdd\x1d\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\u0507\x9f\xd1+\x1f:'\xf7\xe1\tv\x1b#\xca4<H\xe3\u0609$\x1a\x9bOaz(\x00\x00\u07d4\u050e?\x93W\xe3\x03Q8A\xb3\xf8K\u0683\xfc\x89ru\x87\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\u051au\xbb\x93?\xca\x1f\u029a\xa10:d\xb6\xcbD\xea0\xe1\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u0530\x85\xfb\bo=\rh\xbf\x12\x92k\x1c\xc3\x14,\xae\x87p\x89\u0213\u041c\x8fQP\x00\x00\u07d4\u0532\xff;\xae\x19\x93\xff\xeaM;\x18\x021\xdaC\x9fu\x02\xa2\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\u0533\x8a_\xdbc\xe0\x17\x14\xe9\x80\x1d\xb4{\u0250\xbdP\x91\x83\x8a\x01E4\xd9[\xef\x90\\\x00\x00\u07d4\u0538\xbd\xf3\u07daQ\xb0\xb9\x1d\x16\xab\xbe\xa0[\xb4x<\x86a\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd4\xc4\u0467\xc3\xc7I\x84\xf6\x85{/_\a\xe8\xfa\xceh\x05m\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd4\u01act.|\x85}J\x05\xa0L3\xd4\xd0\\\x14gW\x1d\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd4\xcb!\xe5\x90\u0160\xe0h\x016j\xff4,}}\xb1d$\x89\x1a\u01e0\x8e\xad\x02\xf8\x00\x00\u07d4\xd4\xd9,b\xb2\x80\xe0\x0fbm\x86W\xf1\xb8af\xcb\x1ft\x0f\x89\n\xd7\xf264\xcb\xd6\x00\x00\u07d4\xd4\ubc52\x9a#\x87\x1c\xf7\u007f\xe0I\xab\x96\x02\xbe\b\xbe\ns\x89g\x8a\x93b\xe4\x18\x00\x00\xe0\x94\xd4\xeeI\x19\xfb7\xf2\xbb\x97\f?\xffT\xaa\xf1\xf3\u0766\xc0?\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xd4\xfe\xed\x99\xe8\x91|\\TXc_6\x03\xec\xb7\xe8\x17\xa7\u0409

\x10C\xc4<\xde\x1d9\x80\x00\u07d4\xd4\xffF
>\xfa#\x06K\x1c\xaf\x00Qn(pJ\x82\xa4\xf8\x89Hz\x9a0E9D\x00\x00\u07d4\xd5\x00\xe4\xd1\u0242K\xa9\xf5\xb65\u03e3\xa8\xc2\u00cb\xbdL\xed\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xd5\b\u04dcp\x91oj\xbcL\xc7\xf9\x99\xf0\x11\xf0w\x10X\x02\x89\x05rM$\xaf\xe7\u007f\x00\x00\u07d4\xd5\x0f\u007f\xa0>8\x98v\u04d0\x8b`\xa57\xa6pc\x04\xfbV\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xd5\x13\xa4P\x80\xff/\xeb\xe6,\u0545J\xbe)\xeeDg\xf9\x96\x89\bN\x13\xbcO\xc5\xd8\x00\x00\u07d4\xd5'o\f\xd5\xff\xd5\xff\xb6?\x98\xb5p=U\x94\xed\xe0\x83\x8b\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xd5)KfbB0;m\xf0\xb1\u020d7B\x9b\xc8\xc9e\xaa\x89\x10M\r\x00\u04b7\xf6\x00\x00\u07d4\xd5*\xec\xc6I98\xa2\x8c\xa1\xc3g\xb7\x01\xc2\x15\x98\xb6\xa0.\x89;\xa1\x91\v\xf3A\xb0\x00\x00\u07d4\xd5<V\u007f\f?\xf2\xe0\x8b}Y\xe2\xb5\xc74\x85C\u007f\u014d\x89 \x86\xac5\x10R`\x00\x00\u07d4\xd5A\xac\x18z\xd7\xe0\x90R-
\xe6\xda2\x13\xe9\xa7\xf4C\x96s\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd5K\xa2\xd8V\x81\xdc\x13\x0e[\x9b\x02\xc4\xe8\xc8Q9\x1f\u0679\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\xd5U\b\xad\xbb\xbe\x9b\xe8\x1b\x80\xf9zn\xa8\x9a\xddh\xdagO\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xd5U\f\xaa\xf7C\xb07\xc5o\xd2U\x8a\x1c\x8e\xd25\x13\aP\x8a\x01!\xe4\u05106%[\x00\x00\u07d4\xd5Xm\xa4\u5543\xc8\xd8l\xcc\xf7\x1a\x86\x19\u007f\x17\x99gI\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd5\\\x1c\x8d\xfb\xe1\xe0,\xac\xbc\xa6\x0f\xdb\xdd@[\t\xf0\xb7_\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xd5a\u02fc\x05Q]\xe7:\xb8\u03de\xae\x13W4\x1e}\xfd\xf4\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xd5j\x14Mz\xf0\xae\x8d\xf6I\xab\xaeSZ\x15\x98:\xa0M\x02\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x94\xd5r0\x91i\xb1@.\xc8\x13\x1a\x17\xa6\xaa\xc3\"/\x89\xe6\xeb\x8a\x02\xec\x19x\xc4wf\xa0\x00\x00\u07d4\xd5xvh\xc2\xc5\x17[\x01\xa8\xee\x1a\xc3\xec\xc9\u0232\xab\xa9Z\x89Ij\xccg\u05f1\xd4\x00\x00\u07d4\u0548\u00e5\xdf\"\x81\x85\u064e\xe7\xe6\aH%\\\u07a6\x8b\x01\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u054aR\xe0x\xa8\x05Yk\rV\xeaJ\xe13Z\xf0\x1cf\xeb\x89\x0e~\xeb\xa3A\vt\x00\x00\xe0\x94\u0550>\x99x\xee
\xa3\x8c?I\x8dc\xd5\u007f1\xa3\x9fj\x06\x8a\x022\xb3o\xfcg*\xb0\x00\x00\u07d4\u05568\xd3\xc5\xfa\xa7q\x1b\xf0\x85t_\x9d[\xdc#\u0518\u0609lk\x93[\x8b\xbd@\x00\x00\xe0\x94\u055d\x92\xd2\xc8p\x19\x80\xcc\a<7]r\n\xf0dt<\f\x8a\x04\x05\xfd\xf7\u5bc5\xe0\x00\x00\u07d4\u0567\xbe\xc32\xad\xde\x18\xb3\x10KW\x92Tj\xa5\x9b\x87\x9bR\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u0571\x17\xec\x11n\xb8FA\x89a\xeb~\xdbb\x9c\xd0\xddi\u007f\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\u0572\x84\x04\x010\xab\xf7\xc1\xd1cq#q\xcc~(\xadf\u0689j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\u0579\xd2w\u062a\xd2\x06\x97\xa5\x1fv\xe2\tx\x99k\xff\xe0U\x89\a\xc3\xfe<\aj\xb5\x00\x00\u07d4\u057d^\x84U\xc10\x16\x93W\xc4q\xe3\u6077\x99jrv\x89-
\x9e(\x8f\x8a\xbb6\x00\x00\u07d4\xd5\u02e5\xb2k\xea]s\xfa\xbb\x1a\xba\xfa\xcd\xef\x85\xde\xf3h\u0309\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd5\xceU\u0476/YC<!&\xbc\xec\t\xba\xfc\x9d\xfa\xa5\x14\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xd5\xe5Q\x00\xfb\u0455k\xbe\xd2\xcaQ\x8dK\x1f\xa3v\x03+\v\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xd5\xe5\xc15\xd0\xc4\xc3094q\x19\x93\xd0\xd1o\xf9\u7ea0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd5\xe6V\xa1\xb9\x16\xf9\xbfE\xaf\xb0}\u062f\xafs\xb4\xc5oA\x89\x05B%:\x12l\xe4\x00\x00\xe0\x94\xd5\xeaG,\xb9F`\x18\x11\n\xf0\f7l[\\,q1\x12\x8a\x01\x0e\xeehl\x85OD\x00\x00\u07d4\xd5\xf0uR\xb5\u0193\xc2\x00g\xb3x\xb8\t\xce\xe8S\xb8\xf16\x89\x1bg\xc6\u07c8\xc6\xfa\x00\x00\u07d4\xd5\xf7\xc4\x1e\ar\x9d\xfam\xfcd\xc4B1`\xa2,`\x9f\u04c9a\t=|,m8\x00\x00\u07d4\xd6\x04\xab\xceC0\x84.=9l\xa7=\xdbU\x19\xed>\xc0?\x89\b\xe3\x1f\xe1h\x9d\x8a\x00\x00\u0

7d4\xd6\x06Q\xe3\x93x4#\xe5\xcc\x1b\xc5\xf8\x89\xe4N\xf7\xea$>\x89\x15\x9ev7\x11)\xc8\x00\x
00\u07d4\xd6\t\xbfO\x14n\xeak\r\xc8\xe0m\xdc\xf4D\x8a\x1f\xcc\xc9\xfa\x89lk\x93[\x8b\xbd@\x00
\x00\u07d4\xd6\t\xec\v\xe7\r\n\xd2ong\xc9\xd4v+R\xeeQ\x12,\x8965\u026d\xc5\u07a0\x00\x00\u0
7d4\xd6\nRX\a(R\r\xf7Tk\xc1\xe2\x83)\x17\x88\u06ee\f\x8964\x89\xef?\xf0\xd7\x00\x00\u07d4\xd
6\v$s!\xa3*Z\xff\xb9k\x1e'\x99'\xccXM\xe9C\x89z\xd0
\xd6\xdd\xd7v\x00\x00\u07d4\xd6\x11\x02v\xcf\xe3\x1eB\x82ZW\u007fkC]\xbc\xc1\f\xf7d\x8965\u
026d\xc5\u07a0\x00\x00\xe0\x94\xd6\x12Y{\xc3\x17C\u01c63\xf63\xf29\xb1\xe9Bk\xd9%\x8a\x10\
x17\xf7\u07d6\xbe\x17\x80\x00\x00\u07d4\xd6#J\xafE\xc6\xf2.f\xa2%\xff\xb9:\xddb\x9bN\xf8\x0f\x
8965\u026d\xc5\u07a0\x00\x00\u07d4\xd6.\u06d6\xfc\u259a\xaflT^\x96|\xf1\xc0\xbc\x80R\x05\x8
9\x04\xa5eSjZ\u0680\x00\u07d4\xd60\v2\x15\xb1\x1d\xe7b\xec\xdeKp\xb7\x92}\x01)\x15\x82\x89l
k\x93[\x8b\xbd@\x00\x00\u07d4\xd69]\xb5\xa4\xbbf\xe6\x0fL\xfb\xcd\xf0\x05{\xb4\xd9xb\xe2\x891
T\xc9r\x9d\x05x\x00\x00\xe0\x94\xd6J-
P\xf8\x85\x857\x18\x8a$\xe0\xf5\r\xf1h\x1a\xb0~\u05ca\b7Z*\xbc\xca$@\x00\x00\u07d4\xd6X\n\x
b5\xedL}\xfaPo\xa6\xfed\xad\\\xe1)pw2\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xd6Y\x8b\x13\
x86\xe9<\\\u02d6\x02\xffK\xbb\xec\xdb\xd3p\x1d\u0109\f%\xf4\xec\xb0A\xf0\x00\x00\u07d4\xd6d
M@\xe9\v\xc9\u007f\xe7\xdf\xe7\u02bd2i\xfdW\x9b\xa4\xb3\x89\b\x9e\x91y\x94\xf7\x1c\x00\x00\x
e0\x94\xd6g\f\x03m\xf7T\xbeC\xda\u074fP\xfe\xea(\x9d\x06\x1f\u058a\x01D\xa2\x904H\xce\xf7\x
80\x00\u07d4\xd6hR:\x90\xf0)=e\xc58\xd2\xddlWg7\x10\x19n\x89\x02$,0\xb8S\xee\x00\x00\u07d
4\xd6j\xb7\x92\x94\aL\x8bb}\x84-
\xabA\xe1}\xd7\f]\xe5\x8965\u026d\xc5\u07a0\x00\x00\u0794\xd6j\xcc\r\x11\xb6\x89\u03a6\xd9\x
ea_\xf4\x01L\"J]\xc7\u0108\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xd6m\xdf\x11Y\xcf\"\xfd\x8czK\xc
8\u0540wV\xd43\xc4>\x89wC\"\x17\xe6\x83`\x00\x00\u07d4\u0587\xce\xc0\x05\x90\x87\xfd\xc7\x
13\xd4\xd2\xd6^w\xda\xef\xed\xc1_\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\u0588\xe7\x85\u024f
\x00\xf8K:\xa1S3U\u01e2X\xe8yH\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\u05a2.Y\x8d\xab
\u04ce\xa6\xe9X\xbdy\u050d\u0756\x04\xf4\u07c965\u026d\xc5\u07a0\x00\x00\u07d4\u05a7\xac
M\xe7\xb5\x10\xf0\xe8\xdeQ\x9d\x97?\xa4\xc0\x1b\xa84\x00\x89e\xea=\xb7UF`\x00\x00\u07d4\u
05ac\xc2
\xba.Q\xdf\xcf!\xd4C6\x1e\xeav\\\xbd5\u0609\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u05ac\xff\u0
43f\u065c8.{\xd5o\xf0\xe6\x14J\x9eR\xb0\x8e\x89\b\xacr0H\x9e\x80\x00\x00\u07d4\xd6\xc0\u043
c\x93\xa6.%qtp\x0e\x10\xf0$\u0232?\x1f\x87\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xd6\xcf\\\x1b\
u03dd\xa6b\xbc\xea\"U\x90P\x99\xf9\xd6\xe8M\u030a\x01\u011eB\x01W\xd9\xc2\x00\x00\u07d4\
xd6\xd05r\xa4RE\xdb\xd46\x8cO\x82\xc9W\x14\xbd!g\xe2\x89?\x00\xc3\xd6f\x86\xfc\x00\x00\u07
d4\xd6\xd6wiX\xee#\x14:\x81\xad\xad\xeb\b8
\t\xe9\x96\u0089\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xd6\xd9\xe3\x0f\bB\x01*qv\xa9\x17\xd9\xd2\
x04\x8c\xa0s\x87Y\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xd6\xe0\x9e\x98\xfe\x13\x003!\x04
\xc1\xca4\xfb\xfa\xc5T6N\u0649lk\x93[\x8b\xbd@\x00\x00\u07d4\xd6\xe8\xe9z\u90db\x9e\xe5\a\x
ee\xdb(\xed\xfbtw\x03\x149\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd6\uea18\u052e+q\x80'\xa1\x9c
\xe9\xa5\xebs\x00\xab\xe3\u0289\x01}J\xce\xeec\u06c0\x00\xe0\x94\xd6\xf1\xe5[\x16\x94\b\x9e\x
bc\xb4\xfe}x\x82\xaaf\u0217av\x8a\x04<#\xbd\xbe\x92\x9d\xb3\x00\x00\u07d4\xd6\xf4\xa7\xd0N\x
8f\xaf
\xe8\xc6\ub15c\xf7\xf7\x8d\xd2=z\x15\x89\a$\xde\xd1\xc7H\x14\x00\x00\u07d4\xd6\xfc\x04F\u01a
8\xd4\n\xe3U\x1d\xb7\xe7\x01\xd1\xfa\x87nJI\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd7\x03\u01a4

\xf1\x1d`\x19Ey\u054c'f\xa7\xef\x16\xc3\n)\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd7\x05%\x19uj\xf4%\x90\xf1S\x91\xb7#\xa0?\xa5d\xa9Q\x89\xfa61H\r\x01\xfd\x80\x00\u07d4\xd7\na+\xd6\u0769\xea\xb0\xdd\xdc\xffJ\xafA\"\u04cf\xea\xe4\x89\x1dF\x01b\xf5\x16\xf0\x00\x00\u07d4\xd7\n\xd2\xc4\xe9\uefe67\xefV\xbdHj\u04a1\xe5\xbc\xe0\x93\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd7\x14\f\x8eZC\a\xfa\xb0\xcc'\xba\u0752\x95\x01\x8b\xf8yp\x89\x05\xf1\x01kPv\xd0\x00\x00\u07d4\xd7\x16J\xa2a\xc0\x9a\u0672\xb5\x06\x8dE>\xd8\xebj\xa10\x83\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xd7\x1eC\xa4Qw\xadQ\xcb\xe0\xf7!\x84\xa5\xcbP9\x17(Z\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd7\x1f\xb10\xf0\x15\fVRi\xe0\x0e\xfbC\x90+R\xa4U\xa6\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd7\"W8\xdc\xf3W\x848\xf8\xe7\u0233\x83~B\xe0J&/\x89\x18+\x8c\ubec3\xaa\x00\x00\u07d4\xd7'MP\x80M\x9cw\u0693\xfaH\x01V\xef\xe5{\xa5\x01\u0789i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xd71\xbbk_<79^\t\u03ac\xcd\x14\xa9\x18\xa6\x06\a\x89\x89\u0556{\xe4\xfc?\x10\x00\x00\xe0\x94\xd7>\xd2\u0645\xb5\xf2\x1bU\xb2td;\xc6\xda\x03\x1d\x8e\u074d\x8a\nm\xd9\f\xaeQ\x14H\x00\x00\u07d4\xd7D\xac~S\x10\xbeijc\xb0\x03\xc4\v\xd097\x05a\u0189Z\x87\xe7\xd7\xf5\xf6X\x00\x00\xe0\x94\xd7Jn\x8dj\xab4\u0385\x97h\x14\xc12{\xd6\xea\a\x84\u048a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xd7ZP*[gr\x87G\x0fe\u016aQ\xb8|\x10\x15\x05r\x8910\xb4dc\x85t\x00\x00\u07d4\xd7m\xba\xeb\xc3\rN\xf6{\x03\xe6\xe6\xec\xc6\xd8N\x00MP-
\x89mv\xb9\x18\x8e\x13\x85\x00\x00\u07d4\xd7q\xd9\xe0\u028a\b\xa1\x13wW1CN\xb3'\x05\x99\xc4\r\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xd7x\x8e\xf2\x86X\xaa\x06\xccS\xe1\xf3\xf0\xdeX\xe5\xc3q\xbex\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\xd7x\x92\xe2';#]v\x89\xe40\xe7\xae\ud73c\xe8\xa1\xf3\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u05c1\xf7\xfc\t\x18F\x11V\x85p\xb4\x98n,r\x87+~\u0409\x01\x15\x95a\x06]]\x00\x00\u07d4\u05c5\xa8\xf1\x8c8\xb9\xbcO\xfb\x9b\x8f\xa8\xc7r{\xd6B\xee\x1c\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u05ce\xcd%\xad\xc8k\xc2\x05\x1d\x96\xf6Sd\x86kB\xa4&\xb7\x89\xd20X\xbf/&\x12\x00\x00\xe0\x94\u05cf\x84\xe3\x89D\xa0\xe0%_\xae\xceH\xbaIP\u053d9\u048a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\u05d4\x83\xf6\xa8DO%I\xd6\x11\xaf\xe0,C-
\x15\xe1\x10Q\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u05d85\xe4\x04\xfb\x86\xbf\x84_\xba\t\rk\xa2^\f\x88f\xa6\x89\x82\x1a\xb0\xd4AI\x80\x00\x00\u07d4\u05da\xff\x13\xba-
\xa7]F$\f\xac\n$g\xc6V\x94\x98#\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4\u05dd\xb5\xabCb\x1az=\xa7\x95\xe5\x89)\xf3\xdd%\xafg\u0649lj\xccg\u05f1\xd4\x00\x00\u07d4\u05e1C\x1e\xe4S\xd1\xe4\x9a\x05P\xd1%hy\xb4\xf5\xd1\x02\x01\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4\u05ed\t\xc6\xd3&WhSU\xb5\xc6\uc39fW\xb4\ube42\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\u05f7@\xdf\xf8\xc4Wf\x8f\xdft\xf6\xa2f\xbf\xc1\u0737#\xf9\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xd7\u0080>\u05f0\xe0\x83sQA\x1a\x8ef7\xd1h\xbc[\x05\x8a\x06A\xda\xf5\xc9\x1b\xd95\x80\x00\u07d4\xd7\xc6&]\xea\x11\x87l\x90;q\x8eL\u062b$\xfe&[\u0789lk\x93[\x8b\xbd@\x00\x00\u07d4\xd7\xca\u007f\xdc\xfe\xbeE\x88\xef\xf5B\x1d\x15\"\xb6\x13(\xdf{\xf3\x89\xd8\xe6\x00\x1el0+\x00\x00\u07d4\xd7\u037dA\xff\xf2\r\xf7'\xc7\vbU\xc1\xbav\x06\x05Th\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd7\xd1W\xe4\xc0\xa9d7\xa6\u0485t\x1d\xd2>\xc46\x1f\xa3k\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd7\xd2\xc6\xfc\xa8\xad\x1fu9R\x10\xb5}\xe5\xdf\xd6s\x939\t\x89\x12nr\xa6\x9aP\xd0\x00\x00\xe0\x94\xd7\xd3\xc7Y
Y\x048\xb8,>\x95\x15\xbe.\xb6\xedz\x8b\x1a\x8a\f\xb4\x9bD\xba`-
\x80\x00\x00\u07d4\xd7\xd7\xf2\u02a4b\xa4\x1b;0\xa3J\xeb;\xa6\x10\x10\xe2bo\x89lk\x93[\x8b\xb

d@\x00\x00\u07d4\xd7\xe7J\xfd\xba\xd5^\x96\u03bcZ7O,\x8bv\x86\x80\xf2\xb0\x89\x05]\xe6\xa7
y\xbb\xac\x00\x00\xe0\x94\xd7\xeb\x901b'\x1c\x1a\xfa5\xfei\xe3s\"\u0224\u049b\x11\x8a\x02\x1e\
x19\xe0\u027a\xb2@\x00\x00\u07d4\xd7\xeb\u0779\xf99\x87w\x9bh\x01U7T8\xdbe\xaf\xcbj\x89\x
05t\x1a\xfe\xff\x94L\x00\x00\u07d4\xd7\xef4\x0ef\xb0\u05ef\xcc\xe2\n\x19\xcb{\xfc\x81\xda3\xd9N
\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xd7\xf3p\u053e\xd9\xd5|oI\u0259\xder\x9e\xe5i\xd3\xf4\x
e4\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd7\xfa_\xfb`H\xf9o\xb1\xab\xa0\x9e\xf8{\x1c\x11\xddp\x05\xe4\x8965\u026d\xc5\
u07a0\x00\x00\u07d4\xd8\x06\x9f\x84\xb5!I?G\x15\x03\u007f2&\xb2_3\xb6\x05\x86\x89g\x8a\x93
b\xe4\x18\x00\x00\u0794\xd8\x15\xe1\xd9\xf4\xe2\xb5\xe5~4\x82k|\xfd\x88\x81\xb8Th\x90\x88\xf
0\x15\xf2W6B\x00\x00\u07d4\xd8\x1b\xd5K\xa2\xc4Jok\xeb\x15a\u058b\x80\xb5DNm\u0189?\x1
7\r~\xe4<C\x00\x00\xe0\x94\xd8\"QEm\xc18\x0f\x8fV\x92\xf9b\x82\x86@\xab\x9f*\x03\x8a\x01\b\
x8bS\xb2\xc2\x02\xbe\x00\x00\u07d4\xd8,o\xed\xbd\xac\x98\xaf.\xed\x10\xb0\x0f2\xb0\x00V\xcaZ
m\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xd8/\xd9\xfd\xf6\x99k\xed\xad(C\x15\x9c\x06\xf3~\t$3}\x89[\x8c\xce\xdcZ\xa7\xb0\
x00\x00\u07d4\xd8:\xd2`\xe9\xa6\xf42\xfbn\xa2\x87C)\x9bJ\t\xade\x8c\x89lk\x93[\x8b\xbd@\x00\x
00\u0794\xd8C\xee\bc\u0393>\"\xf8\x9c\x80-
1({\x96q\xe8\x1c\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\xd8K\x92/xA\xfcWt\xf0\x0e\x14`J\xe0\x
dfB\xc8U\x1e\x89\xd9o\u0390\u03eb\xcc\x00\x00\u07d4\xd8U\xb0<\xcb\x02\x9awG\xb1\xf0s\x03\
xe0\xa6dy59\u0209lk\x93[\x8b\xbd@\x00\x00\u07d4\xd8_\u07af*a\xf9]\xb9\x02\xf9\xb5\xa5<\x9b\
x8f\x92f\u00ec\x89l\xf6Z~\x90G(\x00\x00\u07d4\xd8q^\xf9\x17o\x85\v.0\xeb\x8e8'\a\xf7w\xa6\xfb\
xe9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd8t\xb9\u07eeEj\x92\x9b\xa3\xb1\xa2~W,\x9b,\xec\u07f
3\x89\t79SM(h\x00\x00\u07d4\u0613\n9\xc7sW\xc3\n\u04e0`\xf0\v\x06\x04c1\xfdb\x89,s\xc97t,P\x
00\x00\u07d4\u061b\xc2q\xb2{\xa3\xabib\xc9JU\x90\x06\xae8\xd5\xf5j\x89lk\x93[\x8b\xbd@\x00\x
00\u07d4\u0637}\xb9\xb8\x1b\xbe\x90B{b\xf7\x02\xb2\x01\xff\u009f\xf6\x18\x892m\x1eC\x96\xd4\
\\x00\x00\u07d4\xd8\xcdd\xe0(N\xecS\xaaF9\xaf\xc4u\b\x10\xb9\u007f\xabV\x89\x01\x15\x8eF\t\x
13\xd0\x00\x00\u07d4\xd8\xd6C\x84$\x9bwg\x94\x06;V\x98x\xd5\xe3\xb50\xa4\xb2\x89\t\xa0C\u0
432\xf9V\x80\x00\u07d4\xd8\xd6T
\xc1\x8c#'\xccZ\xf9t%\xf8W\xe4\xa9\xfdQ\xb3\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\xd8\xe5\x
c9g^\xf4\xde\xed&k\x86\x95o\xc4Y\x0e\xa7\u0522}\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xd8\
xe8GB\x92\xe7\xa0Q`L\xa1d\xc0pw\x83\xbb(\x85\xe8\x8a\x02\xd4\xca\x05\xe2\xb4<\xa8\x00\x00\
u07d4\xd8\xebxP>\xc3\x1aT\xa9\x016x\x1a\xe1\t\x00Lt2W\x8965\u026d\xc5\u07a0\x00\x00\u07d
4\xd8\xee\xf4\xcfK\xeb\x01\xee
\xd1\x11t\x8ba\xcbM?d\x1a\x01\x89\x94\x89#z\u06daP\x00\x00\u07d4\xd8\xf4\xba\xe6\xf8M\x91\r
m}Z\xc9\x14\xb1\xe6\x83r\xf9A5\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xd8\xf6
6\xf0;v5\xb8X\xf1\x10?\x8a\x1d\x90\x19\xa8\x92\xb6\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u0
7d4\xd8\xf6e\xfd\x8c\xd5\u00bc\xc6\xdd\xc0\xa8\xaeR\x1eM\u01aa``\x89\\(=A\x03\x94\x10\x00\x0
0\u07d4\xd8\xf9$\fU\xcf\xf05R<m[\xd3\x00\xd3p\u070f\f\x95\x89\x0fs+f\x01ZT\x00\x00\xe0\x94\xd
8\xf9EyIg%\xb5\xcbS\u05d8\\\x98\x97I\xaf\xf8I\xc0\x8a\x03\x99\x92d\x8a#\u0220\x00\x00\u07d4\
xd8\xfd\xf5FgG8\u0244\xd8\xfa\xb8W\x88\v>B\x80\xc0\x9e\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u
07d4\xd8\xfe\b\x8f\xff\u0394\x8fQ7\xee#\xb0\x1d\x95\x9e\x84\xacB#\x89\f[T\xa9O\xc0\x17\x00\x0
0\u07d4\xd9\x0f0\t\xdbC~N\x11\u01c0\xbe\u0209os\x8de\xef\r\x89\xd8\xd7&\xb7\x17z\x80\x00\x0
0\u07d4\xd9\x10;\xb6\xb6zU\xa7\xfe\xce-\x1a\xf6-

E|!x\x94m\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd9\x13\xf0w\x19Iu<G&\xac\xaa+\xd3a\x9c\\
\xffw\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xd9\x1d\x88\x91dG\x9c\xe46\xec\xe5\x17c\xe2,\xda\
x19\xb2-
k\x89\xb6m\x88\x12h\x00\x88\x00\x00\u07d4\xd9)\xc6]i\u057b\xae\xa5\x97bf.\xf4\x18\xbc!\xad\x9
2J\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xd90\xb2zx\x87d\x85\xd0\xf4\x8bp\xddS6T\x96y\u028f
\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xd91\xac&h\xbaj\x84H\x1a\xb19sZ\xec\x14\xb7\xbf\xb
a\xbf\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd98=Km\x17\xb3\xf9\xcdBn\x10\xfb\x94@\x15\xc0\xd4
K\xfb\x89+^:\xf1k\x18\x80\x00\x00\xe0\x94\xd9B\xdeG\x84\xf7\xa4\x87\x16\xc0\xfdK\x9dT\xa6\xe
5L_/>\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xd9D\u0226\x9f\xf2\xca\x12Ii\f\x12)\xc7\x19/6%\x
10b\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xd9JW\x88*Rs\x9b\xbe*\x06G\xc8\f$\xf5\x8a+O\x1c\x8
9H\xb5N*\xdb\xe1+\x00\x00\xe0\x94\xd9SB\x95<\x8a!\xe8\xb65\xee\xfa\u01c1\x9b\xea0\xf1pG\x8
a\x13\xf0l\u007f\xfe\xf0]@\x00\x00\u07d4\xd9\\\x90\xff\xbeT\x84\x86G\x80\xb8gIJ\x83\u0212V\xd
6\xe4\x89X\xe7\x92n\xe8X\xa0\x00\x00\u07d4\xd9g\x11T\x0e.\x99\x83C\xd4\xf5\x90\xb6\xfc\x8f\x
ac;\xb8\xb3\x1d\x89_Z@h\xb7\x1c\xb0\x00\x00\u07d4\xd9j\xc2Pt\t\u01e3\x83\xab.\xee\x18\"\xa5\
xd78\xb3kV\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xd9m\xb3;{Z\x95\f>\xfa-
\xc3\x1b\x10\xba\x10\xa52\uf1c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xd9wYe\xb7\x16Gfu\xa8\xd5\
x13\xeb\x14\xbb\xf7\xb0|\xd1J\x8a\x01\x13.m-
#\xc5\xe4\x00\x00\u07d4\xd9{\xc8J\xbdG\xc0[\xbfE{.\xf6Y\xd6\x1c\xa5\xe5\u43c9\x06\x9d\x17\x1
1\x9d\u0168\x00\x00\u07d4\xd9\u007fE&\u07a9\xb1c\xf8\xe8\xe3:k\u03d2\xfb\x90}\xe6\xec\x89\x
0feJ\xafM\xb2\xf0\x00\x00\u07d4\xd9\u007f\xe6\xf5?*X\xf6\xd7mu*\xdft\xa8\xa2\xc1\x8e\x90t\x89\
x10\xcd\xf9\xb6\x9aCW\x00\x00\u07d4\u0659\x99\xa2I\r\x94\x94\xa50\xca\xe4\xda\xf3\x85T\xf4\x
ddc>\x89\x06\x81U\xa46v\xe0\x00\x00\u07d4\u065d\xf7B\x1b\x93\x82\xe4,\x89\xb0\x06\xc7\xf0\x
87p*\aW\xc0\x89\x1a\x05V\x90\xd9\u06c0\x00\x00\xe0\x94\u0677\x83\xd3\x1d2\xad\xc5\x0f\xa3\
xea\u02a1]\x92\xb5h\xea\xebG\x8a\a3\xaf\x907L\x1b(\x00\x00\u07d4\xd9\xd3p\xfe\xc65v\xab\x15
\xb3\x18\xbf\x9eX6M\u00a3U*\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xd9\xd4/\xd1>\xbdK\xf6\x9c\xac^\x9c~\x82H:\xb4m\xd7\xe9\x8a\x01!\xeah\
xc1\x14\xe5\x10\x00\x00\u07d4\xd9\xe2~\xb0}\xfcq\xa7\x06\x06\f\u007f\a\x928\u0293\xe8\x859\x
8965\u026d\xc5\u07a0\x00\x00\u07d4\xd9\xe3\x85~\xfd\x1e
*D\x17p\xa7w\xa4\x9d\xccE\xe2\xe0\u04c9\f\x1d\xaf\x81\u0623\xce\x00\x00\u07d4\xd9\xec.\xfe\x
99\xff\\\xf0\r\x03\xa81{\x92\xa2J\xefD\x1f~\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xd9\xec\x8f\xe6\x
9bw\x16\xc0\x86Z\xf8\x88\xa1\x1b+\x12\xf7
\xed3\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xd9\xf1\xb2d\b\xf0\xecg\xad\x1d\ro\xe2.\x85\x15
\xe1t\x06$\x89\x01M\x11
\u05f1`\x00\x00\u07d4\xd9\xf5G\xf2\xc1\xde\x0e\u064aS\xd1a\xdfWc]\xd2\x1a\x00\xbd\x89\x05V\
xf6L\x1f\xe7\xfa\x00\x00\u07d4\xd9\xff\x11]\x01&l\x9fs\xb0c\xc1\xc28\xef5e\xe6;6\x89$\xdc\xe5M
4\xa1\xa0\x00\x00\u07d4\xda\x06\x04N)<e,F\u007f\xe7AF\xbf\x18[!3\x8a\x1c\x8965\u026d\xc5\u0
7a0\x00\x00\u07d4\xda\vH\xe4\x89\xd3\x02\xb4\xb7\xbf
O\x95|\x1c\x9b\u30f0\u07c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xda\rK~\xf9\x1f\xb5Z\xd2e\xf2Q\x1
4
g\xf1\x03v\xce\u058a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xda\x10\x97\x8a9\xa4o\U0003b10c\xf
6]\xd9\xc7u\t\xa6\xd7\x0e\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xda\x16\xdd\\=\x1a'\x145\x8f\xe3
u,\xaeS\u06eb+\u930a\x04\x1b\xad\x15^e\x12

\x00\x00\u07d4\xda!L\x02>#&\xffil\x0091h\xceF\xff\xac9\xec\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xda*\x14\xf9r@\x15\u05d0\x14\xed\x8eY\th\x1dYaH\xf1\x89\x02\xa1\x0f\x0f\x8a\x91\xab\x80\x00\u07d4\xda*\u054ew\xde\xdd\xed\xe2\x18vF\xc4e\x94Z\x8d\xc3\xf6A\x89#\xc7W\a+\x8d\xd0\x00\x00\u07d4\xda0\x17\xc1P\xdd\r\xce\u007f\u03c8\x1b\nH\xd0\xd1\xc7V\xc4\u01c9\x05k\xf9\x1b\x1ae\xeb\x00\x00\u07d4\xda4\xb2\xea\xe3\v\xaf\xe8\xda\xec\xcd\xe8\x19\xa7\x94\u0349\xe0\x95I\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xdaJ_U\u007f;\xab9\n\x92\xf4\x9b\x9b\x90\n\xf3\fF\xae\x80\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xdaPU7S\u007f\xfb3\xc4\x15\xfe\xc6Ni\xba\xe0\x90\xc5\xf6\x0f\x89\b\xacr0H\x9e\x80\x00\x00\u07d4\xdai\x8dd\xc6\\\u007f+,rS\x05\x9c\xd3\u0441\u0619\xb6\xb7\x89\x10\x04\xe2\xe4_\xb7\xee\x00\x00\u07d4\xdaw2\xf0/.'.\xaf(\u07d7.\xcc\r\xde\xed\x9c\xf4\x98\x89\v
\xbf\xbfig\x89\x00\x00\u07d4\xdaz\xd0%\xeb\xde%\xd2\"C\u02c3\x0e\xa1\xd3\xf6JVc#\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\u0685]SG\u007fP^\xc4\xc8\xd5\u8ed1\x80\u04c6\x81\x11\x9c\x8a\x01/\x93\x9c\x99\xed\xab\x80\x00\x00\u07d4\u0687^N/<\xab\xe4\xf3~\x0e\xae\xd7\xd1\xf6\xdc\xc6\xff\xefC\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u068b\xbe\xe1\x82\xe4U\xd2\t\x8a\xcb3\x8amE\xb4\xb1~\u0636\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u0698.\x96C\xff\xec\xe7#\aZ@\xfewnZ\xce\x04\xb2\x9b\x89\b\xb8\xb6\u0259\x9b\xf2\x00\x00\u07d4\u069fUF\tF\u05ff\xb5p\xdd\xecu|\xa5w;XB\x9a\x89\x1b\x84]v\x9e\xb4H\x00\x00\u07d4\u06a1\xbdz\x91H\xfb\x86\\\xd6\x12\xdd5\xf1b\x86\x1d\x0f;\u0709\xa68\xabr\xd9,\x13\x80\x00\xe0\x94\u06a6<\xbd\xa4]\u0507\xa3\xf1\xcdJtj\x01\xbb^\x06\v\x90\x8a\x01\x04\x16\u0670*\x89$\x00\x00\u07d4\u06a7v\xa6uDi\u05f9&z\x89\xb8g%\xe7@\xda\x0f\xa0\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\u06ac\x91\xc1\xe8Y\xd5\xe5~\xd3\bKP
\x0f\x97f\xe2\xc5+\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\u06ac\xda\xf4\"&\xd1\\\xb1\u03d8\xfa\x15\x04\x8c\u007fL\xee\xfei\x89\x10CV\x1a\x88)0\x00\x00\xe0\x94\u06b6\xbc\u06c3\xcf$\xa0\xae\x1c\xb2\x1b;[\x83\xc2\xf3\x82I'\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\u06bb\b\x89\xfc\x04)&\xb0^\xf5{%
\x91\n\xbcKAI\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u06bc\"PB\xa6Y,\xfa\x13\xeb\xe5N\xfaA\x04\bx\xa5\xa2\x89\x0e\x11\xfa\xd5\xd8\\\xa3\x00\x00\u07d4\xda\xc0\xc1w\xf1\x1c\\>>x\xf2\xef\xd6c\xd12!H\x85t\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xda\xd16\xb8\x81x\xb4\x83zlx\x0f\xeb\xa2&\xb9\x85i\xa9L\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xda\xdb\xfa\xfd\x8bb\xb9*$\xef\xd7RV\u0743\xab\xdb\u05fb\u06c9\x01\x11du\x9f\xfb2\x00\x00\u07d4\xda\xdc\x00\xaby'`</\xcf1\xce\xe3R\xf8\x0elMcQ\x89lf\xe9\xa5Sx\xb8\x00\x00\u07d4\xda\xe0\xd3>\xaa4\x15i\xfa\x9f\xf5\x98&\x84\x85JJ2\x8an\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xda\xe7
\x1e\xab\x8c\x063\x02\x93\ri9)\xd0\u007f\x95\xe7\x19b\x89\x91\xae\xc0(\xb4\x19\x81\x00\x00\u07d4\xda\xed\u052d\x10{'\x1e\x89Hl\xbf\x80\xeb\xd6!\u0757Ex\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdb\x04\xfa\xd9\u011f\x9e\x88\v\xeb\x8f\xcf\x1d:8\x90\u4cc4o\x89CZ\xe6\xcc\fX\xe5\x00\x00\u07d4\xdb\f\u01cft\u0642{\u070ads'n\xb8O\u0717b\x12\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdb\x12\x93\xa5\x06\xe9\f\xad*Y\xe1\xb8V\x1f^f\x96\x1ag\x88\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdb\x19\xa3\x98\"06\x8f\x01w!\x9c\xb1\f\xb2Y\u0372%|\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdb#\xa6\xfe\xf1\xaf{X\x1ew,\xf9\x18\x82\u07b2Qo\xc0\xa7\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xdb$O\x97\xd9\xc4K\x15\x8a@\xed\x96\x06\xd9\xf7\xbd8\x9131\x89\x05\x87\x88\u02d4\xb1\xd8\x00\x00\u07d4\xdb(\x8f\x80\xff\xe22\u00baG\u0314\xc7c\xcfo\u0278+\r\x89\x04\x

9b\x9c\xa9\xa6\x944\x00\x00\u07d4\xdb*\f\x9a\xb6M\xf5\x8d\u07f1\u06ec\xf8\xba\r\x89\xc8[1\xb4\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xdb4t^\u0785v\xb4\x99\xdb\x01\xbe\xb7\xc1\xec\u0685\xcfJ\xbe\x89\x04V9\x18$O@\x00\x00\u07d4\xdb?%\x8a\xb2\xa3\xc2\xcf3\x9cD\x99\xf7ZK\xd1\xd3G.\x9e\x89QP\xae\x84\xa8\xcd\xf0\x00\x00\u07d4\xdbK\xc8;\x0ek\xaa\xdb\x11V\xc5\xcf\x06\xe0\xf7!\x80\x8cR\u01c9/\xb4t\t\x8fg\xc0\x00\x00\u07d4\xdbc\x12-
\xe7\x03}\xa4\x97\x151\xfa\u9bc5\x86x\x86\u0192\x89\x0f\x04%\xb0d\x1f4\x00\x00\u07d4\xdbl*s\xda\xc7BJ\xb0\xd01\xb6ga\x12%f\xc0\x10C\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xdbnV\f\x9b\xc6 \u053e\xa3\xa9MG\xf7\x88\v\xf4\u007f-
_\x89\x04\xda\x0f\xdf\xcf\x05v\x00\x00\u07d4\xdbo\xf7\x1b=\xb0\x92\x8f\x83\x9e\x05\xa72;\xfbW\u049c\x87\xaa\x891T\xc9r\x9d\x05x\x00\x00\u07d4\xdbsF\vY\xd8\xe8PE\xd5\xe7R\xe6%Y\x87^BP.\x8963\x03\"\xd5#\x8c\x00\x00\u07d4\xdbw\xb8\x8d\xcbq/\xd1~\xe9\x1a[\x94t\x8dr\f\x90\xa9\x94\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdb}@7\b\x1fle\xf9Gk\x06\x87\xd9\u007f\x1e\x04M\n\x1d\x89#\xc7W\a+\x8d\xd0\x00\x00\xe0\x94\u06c8.\xac\xed\xd0\xef\xf2cQ\x1b1*\u06fcY\u01b8\xb2[\x8a\x01\xedO\xdez\"6\xb0\x00\x00\u07d4\u06d3q\xb3\fL\x84NY\xe0>\x92K\xe6\x06\xa98\xd1\xd3\x10\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u06e4ym\f\xebM:\x83k\x84\xc9o\x91\n\xfc\x10?[\xa0\x89\t\b\xf4\x93\xf77A\x00\x00\u07d4\u06ed\xc6\x1e\xd5\xf0F\n\u007f\x18\xe5\x1b/\xb2aM\x92d\xa0\xe0\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\u06f6\xacH@'\x04\x16B\xbb\xfd\x8d\x80\xf9\xd0\xc1\xcf3\xc1\xeb\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u06fc\xbby\xbfG\x9aB\xadq\xdb\u02b7{Z\u07ea\x87,X\x89]\u0212\xaa\x111\xc8\x00\x00\u07d4\xdb\xc1\xce\x0eI\xb1\xa7\x05\xd2.
7\xae\xc8x\xee\ru\xc7\x03\x89\r\x8drkqw\xa8\x00\x00\u07d4\xdb\xc1\xd0\xee+\xabS\x11@\xde\x13w\"\xcd6\xbd\xb4\xe4q\x94\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xdb\u015e\u0609s\u07ad1\b\x84\":\xf4\x97c\xc0P0\xf1\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u0794\xdb\xc6ie\xe4&\xff\x1a\xc8z\xd6\xebx\xc1\xd9Rq\x15\x8f\x9f\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xdb\xcb\xcdzW\ua7724\x9b\x87\x8a\xf3K\x1a\xd6B\xa7\xf1\u0449\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xdb\xd5\x1c\xdf,;\xfa\xcd\xff\x10b!\xde.\x19\xadmB\x04\x14\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\xdb\xd7\x1e\xfaK\x93\u0209\xe7e\x93\xde`\x9c;\x04\u02ef\xbe\b\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xdb\xf5\xf0a\xa0\xf4\x8e^ia\x879\xa7}.\xc1\x97h\xd2\x01\x89\b=lz\xabc`\x00\x00\u07d4\xdb\xf8\xb19g\xf5Q%'-
\xe0V%6\xc4P\xbaVU\xa0\x89n\xf5x\xf0n\f\xcb\x00\x00\u07d4\xdb\xfb\x1b\xb4d\xb8\xa5\x8eP\r.\xd8\u0797,E\xf5\xf1\xc0\xfb\x89V\xbcu\xe2\xd61\x00\x00\x00\xe0\x94\xdc\x06~\xd3\xe1-
q\x1e\xd4u\xf5\x15n\xf7\xe7\x1a\x80\xd94\xb9\x8a\x02\x05\xb4\u07e1\xeetx\x00\x00\u07d4\xdc\b\u007f\x93\x90\xfb\x9e\x97j\xc2:\xb6\x89TJ\tB\xec
!\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xdc\x1e\xb9\xb6\xe6CQ\xf5d$P\x96E\xf8>y\xee\xe7l\xf4\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xdc\x1f\x19ya_\b!@\xb8\xbbx\xc6{'\xa1\x94'\x13\xb1\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\xdc#\xb2`\xfc\xc2n}\x10\xf4\xbd\x04J\xf7\x94W\x94`\xd9\u0689\x1b\x1bk\u05efd\xc7\x00\x00\u07d4\xdc)\x11\x97E\xd23s
\xdaQ\xe1\x91\x00\xc9H\u0640\xb9\x15\x89\b\xacr0H\x9e\x80\x00\x00\u07d4\xdc-
\x15\xa6\x9fk\xb3;$j\xef@E\aQ\xc2\xf6uj\u0489l4\x10\x80\xbd\x1f\xb0\x00\x00\u07d4\xdc=\xaeY\xed\x0f\xe1\x8bXQ\x1eo\xe2\xfbi\xb2\x19h\x94#\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xdc?\x0evr\xf7\x1f\xe7R[\xa3\v\x97U\x18:
\xb9\x16j\x8a\x02\b\x9c\xf5{[>\x96\x80\x00\xe0\x94\xdcCE\u0581.\x87\n\xe9\fV\x8cg\xd2\xc5g\u0

3f4\xf0<\x8a\x01k5-
\xa5\xe0\xed0\x00\x00\u07d4\xdcD'[\x17\x15\xba\xea\x1b\x03EsZ)\xacB\xc9\xf5\x1bO\x89?\x19\xbe\xb8\xdd\x1a\xb0\x00\x00\u07d4\xdcF\xc13%\u034e\xdf\x020\xd0h\x89d\x86\xf0\a\xbfN\xf1\x89Hz\x9a0E9D\x00\x00\u07d4\xdcQ\xb2\u071d$z\x1d\x0e[\xc3l\xa3\x15oz\xf2\x1f\xf9\xf6\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xdcS\x05\xb4\x02\n\x06\xb4\x9de||\xa3L5\xc9\x1c_,V\x8a\x01}\xf6\xc1\r\xbe\xba\x97\x00\x00\u07d4\xdcW4[8\xe0\xf0g\u0263\x1d\x9d\xea\xc5'Z\x10\x94\x93!\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xdcWG}\xaf\xa4/p\\\u007f\xe4\x0e\xae\x9c\x81un\x02%\xf1\x89\x1b\x1b\x81(\xa7An\x00\x00\u07d4\xdc_Z\xd6c\xa6\xf2c2}d\xca\xc9\xcb\x13=,\x96\x05\x97\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdcp:_7\x94\xc8Ml\xb3TI\x18\xca\xe1J5\u00fdO\x89dI\xe8NG\xa8\xa8\x00\x00\xe0\x94\xdcs\x8f\xb2\x17\u03ad/iYL\b\x17\r\xe1\xaf\x10\xc4\x19\xe3\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xdcv\xe8[\xa5\v\x9b1\xec\x1e&
\xbc\xe6\xe7\xc8\x05\x8c\x0e\xaf\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\u0703\xb6\xfd\rQ!1G\a\xea\xf7.\xa0\xc8\u027e\xf9v\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u070c)\x12\xf0\x84\xa6\u0444\xaasc\x85\x13\u033c2n\x01\x02\x89F3\xbc6\xcb\xc2\xdc\x00\x00\u07d4\u0711\x1c\xf7\xdc]\u04016Vg\x05(\xe93\x8eg\x03G\x86\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u0730;\xfal\x111#NV\xb7\xea|Or\x14\x87Tkz\x89Hz\x9a0E9D\x00\x00\xe0\x94\u0736M\xf47X\xc7\u03d7O\xa6`HO\xbbq\x8f\x8cg\xc1\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xdc\xc5-
\x8f\x8d\x9f\xc7B\xa8\xb8'g\xf0US\x87\xc5c\xef\xff\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xdc\xcb7\x0e\u058a\xa9\"(0C\xef|\xad\x1b\x9d@?\xc3J\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xdc\u0324
E\xec>\x16P\x8b`?\xd96\xe7\xfd}\xe5\xf3j\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4\xdc\xd1\fU\xbb\x85OuD4\xf1!\x9c,\x9a\x98\xac\xe7\x9f\x03\x89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4\xdc\u057c\xa2\x00S\x95\xb6u\xfd\xe5\x03VY\xb2k\xfe\xfcI\xee\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xdc\u06fdN&\x04\xe4\x0e\x17\x10\xccg0(\x9d\xcc\xfa\u04c9-
\x89\xf9]\xd2\xec'\xcc\xe0\x00\x00\u07d4\xdc\xe3\f1\xf3\xcafr\x1e\xcb!<\x80\x9a\xabV\x1d\x9bR\xe4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdc\xf39eS\x13\x80\x161h\xfc\x11\xf6~\x89\xc6\xf1\xbc\x17\x8a\x89\x12'v\x854\x06\xb0\x80\x00\u07d4\xdc\xf6\xb6W&n\x91\xa4\xda\xe6\x03=\xda\xc1S2\u074d+4\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\xdc\xf9q\x9b\xe8|oFum\xb4\x89\x1d\xb9\xb6\x11\xd2F\x9cP\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xdc\xff\xf3\xe8\xd2<*4\xb5k\u0473\xbdE\u01d3tC\"9\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xdd\x04\xee\xe7N\v\xf3\f?\x8dl,\u007fR\xe0Q\x92\x10\u07d3\x89\x04V9\x18$O@\x00\x00\xe0\x94\xdd&\xb4)\xfdC\xd8N\xc1y\x82S$\xba\u057f\xb9\x16\xb3`\x8a\x01\x16\xbf\x95\xbc\x842\x98\x00\x00\u07d4\xdd*#:\xde\xdef\xfe\x11&\xd6\xc1h#\xb6*\x02\x1f\xed\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xdd+\u07e9\x17\xc1\xf3\x10\xe6\xfa5\xaa\x8a\xf1i9\xc23\xcd}\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xdd5\xcf\xdb\u02d93\x95Sz\xec\xc9\xf5\x90\x85\xa8\xd5\u0776\xf5\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xddG\x18\x9a>d9qg\xf0b\x0eHEe\xb7b\xbf\xbb\xf4\x89dI\xe8NG\xa8\xa8\x00\x00\u07d4\xddM\xd6\xd3`3\xb0co\u030d\t8`\x9fM\xd6OJ\x86\x89\x03@\xaa\xd2\x1b;p\x00\x00\u07d4\xddO_\xa2\x11\x1d\xb6\x8fk\xde5\x89\xb60)9[i\xa9-
\x89\b\x96=\xd8\xc2\xc5\xe0\x00\x00\xe0\x94\xddc\x04/%\xed2\x88J\xd2n:\xd9Y\xeb\x94\xea6\xbfg\x8a\x04\x84\xd7\xfd\xe7\u0553\xf0\x00\x00\u07d4\xdde\xf6\xe1qc\xb5\xd2\x03d\x1fQ\xcc{$\xb0\x0f\x02\xc8\xfb\x89\n\u05ce\xbcZ\xc6

\x00\x00\u07d4\xddl\x06!\x93\xea\xc2=/\xdb\xf9\x97\xd5\x06:4k\xb3\xb4p\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xdd{\u0366Y$\xaa\xa4\x9b\x80\x98J\xe1su\x02X\xb9(G\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xdd\u007f\xf4A\xbao\xfe6q\xf3\xc0\u06bb\xff\x18#\xa5\x043p\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u0742T\x12\x1an\x94/\xc9\b(\xf2C\x1fQ\x1d\xad\u007f2\u6263\x9b)\xe1\xf3`\xe8\x00\x00\xe0\x94\u074a\xf9\xe7vR#\xf4DoD\xd3\xd5\t\x81\x9a==\xb4\x11\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\u0755\xdb\xe3\x0f\x1f\x18w\xc5\xddv\x84\xae\xef0*\xb6\x88Q\x92\x8a\x01\xc5\xd8\xd6\xeb>2P\x00\x00\xe0\x94\u0756|L_\x8a\xe4~&o\xb4\x16\xaa\u0456N\xe3\xe7\xe8\u00ca\x01\xa4
\xdb\x02\xbd}X\x00\x00\u07d4\u075bHZ;\x1c\xd3:j\x9cb\xf1\xe5\xbe\xe9'\x01\x85m%\x89\f3\x83\xed\x03\x1b~\x80\x00\xe0\x94\u0763q\xe6\x00\xd3\x06\x88\xd4q\x0e\b\x8e\x02\xfd\xf2\xb9RM_\x8a\x01w\"J\xa8D\xc7 \x00\x00\u07d4\u0764\xed*X\xa8\xdd
\xa72u4{X\rq\xb9[\xf9\x9a\x89\x15\xa1<\xc2\x01\xe4\xdc\x00\x00\xe0\x94\u0764\xff}\xe4\x91\u0187\xdfEt\xdd\x1b\x17\xff\x8f$k\xa3\u044a\x04&\x84\xa4\x1a\xbf\xd8@\x00\x00\u07d4\u076bkQ\xa9\x03\v@\xfb\x95\xcf\vt\x8a\x05\x9c$\x17\xbe\u01c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94\u076bu\xfb/\xf9\xfe\u02c8\xf8\x94vh\x8e+\x00\xe3g\xeb\xf9\x8a\x04\x1b\xad\x15^e\x12
\x00\x00\xe0\x94\u076b\xf1<<\x8e\xa4\xe3\xd7=x\xecqz\xfa\xfaC\x0eTy\x8a\b\xcf#\xf9\t\xc0\xfa\x00\x00\x00\u07d4\u076c1*\x96UBj\x9c\f\x9e\xfa?\xd8%Y\xefE\x05\xbf\x89\x15\xbeat\xe1\x91.\x00\x00\u07d4\u076ck\xf4\xbb\xdd}Y}\x9chm\x06\x95Y;\xed\xcc\xc7\xfa\x89.\xe4IU\b\x98\xe4\x00\x00\xe0\x94\u077d+\x93,v;\xa5\xb1\xb7\xae;6.\xac>\x8d@\x12\x1a\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u077d\xdd\x1b\xbd8\xff\xad\xe00]0\xf0
(\xd9.\x9f:\xa8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u077e\xe6\xf0\x94\xea\xe64
\xb0\x03\xfbGW\x14*\xeal\xd0\xfd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdd\u059c[\x9b\xf5\xebZ9\xce\xe7\xc34\x1a\x12\r\x97?\xdb4\x89k\xc1K\x8f\x8e\x1b5\x00\x00\xe0\x94\xdd\xdd{\x9en\xab@\x9b\x92&:\xc2r\u0680\x1bfO\x8aW\x8ai\xe1\r\xe7fv\u0400\x00\x00\u07d4\xdd\xe6p\xd0\x169fuv\xa2-
\xd0]2F\xd6\x1f\x06\xe0\x83\x89\x01s\x17\x90SM\xf2\x00\x00\xe0\x94\xdd\xe7zG@\xba\b\xe7\xf7?\xbe:\x16t\x91)1t.\xeb\x8a\x044\xfeMC\x82\xf1\u0500\x00\u07d4\xdd\xe8\xf0\xc3\x1bt\x15Q\x1d\xce\xd1\xcd}F2>K\xd1\"2\x89WG=\x05\u06ba\xe8\x00\x00\u07d4\xdd\xe9i\xae\xf3N\xa8z\u0099\xb7Y~)+J\x01U\u030a\x89\x102\xf2YJ\x01s\x80\x00\u07d4\xdd\xf0\xcc\xe1\xfe\x99m\x91v5\xf0\a\x12\xf4\x05
\x91\xdf\xf9\xea\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xdd\xf3\xadv58\x10\xbej\x89\xd71\xb7\x87\xf6\xf1q\x88a+\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xdd\xf5\x81\n\x0e\xb2\xfb.22;\xb2\u0255\t\xab2\x0f$\xac\x8a\x03\xca\\f\u067cD0\x00\x00\xe0\x94\xdd\xf9\\\x1e\x99\xce/\x9fV\x98\x05|\x19\xd5\xc9@'\xeeJn\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u0794\xdd\xfa\xfd\xbc|\x90\xf12\x0eT\xb9\x8f7F\x17\xfb\xd0\x1d\x10\x9f\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\xdd\xfc\xca\x13\xf94\xf0\u03fe#\x1d\xa109\xd7\x04u\xe6\xa1\u040968\"\x16`\xa5\xaa\x80\x00\u07d4\xde\x02~\xfb\xb3\x85\x03\"n\xd8q\t\x9c\xb3\v\xdb\x02\xaf\x135\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xde\x06\xd5\xeawzN\xb1G^`]\xbc\xbfCDN\x807\xea\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\xde\a\xfb[zFN;\xa7\xfb\xe0\x9e\x9a\xcb'\x1a\xf53\x8cX\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\xde\x11!\x82\x9c\x9a\b(@\x87\xa4?\xbd/\xc1\x14*23\xb4\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xde\x17kR\x84\xbc\xee:\x83\x8b\xa2Og\xfc|\xbfg\u05ce\xf6\x89\x02\t\xce\b\xc9b\xb0\x00\x00\u07d4\xde!\"\x93\xf8\xf1\xd21\xfa\x10\xe6\tG\rQ,\xb8\xff\xc5\x12\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xde0\xe4\x

9eZ\xb3\x13!M/\x01\u072b\u0389@\xb8\x1b\x1cv\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xde3
\xd7\b\xa3\xb8\x9e\x90\x9e\xafe;0\xfd\u00e5\xd5\u0334\xb3\x89\t\x9c\x88\"\x9f\xd4\xc2\x00\x00\u
07d4\xde7B\x99\xc1\xd0}ySs\x85\x19\x0fD.\xf9\xca$\x06\x1f\x89\a?u\u0460\x85\xba\x00\x00\u07
d4\xdeB\xfc\xd2L\xe4#\x93\x830CgY_\x06\x8f\fa\a@\x89\x02r*p\xf1\xa9\xa0\x00\x00\u07d4\xdeP\
x86\x8e\xb7\xe3\xc7\x197\xecs\xfa\x89\u074b\x9e\xe1\rE\xaa\x8965\u026d\xc5\u07a0\x00\x00\u0
7d4\xdeU\xde\x04X\xf8P\xb3~Mx\xa6A\xdd.\xb2\u074f8\u0389\xd8\xd7&\xb7\x17z\x80\x00\x00\u
07d4\xde[\x00_\xe8\u06ae\x8d\x1f\x05\xde>\xda\x04
f\xc6\xc4i\x1c\x89;\xa1\x91\v\xf3A\xb0\x00\x00\u07d4\xdea-
\a$\xe8N\xa4\xa7\xfe\xaa=!B\xbd^\xe8-
2\x01\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xdem61\x06\xccb8\xd2\xf0\x92\xf0\xf07!6\xd1\
xcdP\u018a\x01!\xeah\xc1\x14\xe5\x10\x00\x00\u07d4\xde}\xee\"\x0f\x04W\xa7\x18}V\xc1\xc4\x1f
.\xb0\n\xc5`!\x89\"%\xf3\x9c\x85\x05*\x00\x00\u07d4\u0782\u030dJ\x1b\xb1\xd9CC\x92\x96[>\x80
\xba\xd3\xc0=O\x89P\x18nu\u0797\xa6\x00\x00\u07d4\u0797\xf43\a\x00\xb4\x8cImC|\x91\xca\x1
d\xe9\u0130\x1b\xa4\x89\x9d\xcc\x05\x15\xb5n\f\x00\x00\u07d4\u079e\xffLy\x88\x11\xd9h\xdc\xc
bF\r\x9b\x06\x9c\xf3\x02x\xe0\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\u07b1\xbc4\xd8mJM\xd
e%\x80\u063e\xaf\aN\xb0\xe1\xa2D\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\u07b2I]j\xca{*j-
\x13\x8bn\x1aB\xe2\xdc1\x1f\u0749lk\x93[\x8b\xbd@\x00\x00\u07d4\u07b9rTGL\r/Zyp\xdc\xdb/R\
xfb\x10\x98\xb8\x96\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u07b9\xa4\x9aC\x870
\xf0u\x91\x85\xe2\v\xbbL\U000c1ecf\x89\vx\xed\xb0\xbf.^\x00\x00\u07d4\u07bb\u0743\x1e\x0f
\xaen7\x82R\xde\xcd\xf9/|\xf0\xc6X\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xde\xc3\xee\xc2d\nu,Fn+
~~\u616f\xe9\xacA\xf4\x89G\u0257SYk(\x80\x00\u07d4\xde\xc8#s\xad\xe8\xeb\xcf*\xcbo\x8b\xc2
AM\u05eb\xb7\rw\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xde\u0221\xa8\x98\xf1\xb8\x95\xd80\x1f\xe6J\xb3\xad]\xe9A\xf6\x89\x89*\xb4\xf6
~\x8as\x0f\x80\x00\u07d4\xde\u025e\x97/\xcaqwP\x8c\x8e\x1aG\xac\"\xd7h\xac\xab|\x89lk\x93[\x
8b\xbd@\x00\x00\u07d4\xde\xd8w7\x84\a\xb9Nx\x1cN\xf4\xaf|\xfc[\xc2
\xb5\x16\x89\x141y\xd8i\x11\x02\x00\x00\u07d4\xde\xe9B\xd5\xca\xf5\xfa\xc1\x14!\xd8k\x01\vE\x
8e\\9)\x90\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xde\xee&\x89\xfa\x90\x06\xb5\x9c\xf2\x85#
}\xe5;:\u007f\xd0\x148\x89\x18ey\xf2\x9e
%\x00\x00\u07d4\xde\xfd\xdf\u055b\x8d,\x15N\xec\xf5\xc7\xc1g\xbf\v\xa2\x90]>\x89\x05\x12\xcb^
&GB\x00\x00\u07d4\xde\xfe\x91A\xf4pE\x99\x15\x9d{\"=\xe4+\xff\xd8\x04\x96\xb3\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xdf\t\x8f^N=\xff\xa5\x1a\xf27\xbd\xa8e,Os\ud726\x89\x1b6\xa6DJ>\x18\x00\
x00\xe0\x94\xdf\r\ba{\xd2R\xa9\x11\u07cb\xd4\x1a9\xb8=\u07c0\x96s\x8a\x02\x1e\x19\xe0\u027a
\xb2@\x00\x00\u07d4\xdf\x0f\xf1\xf3\xd2z\x8e\xc9\xfb\x8fk\f\xb2T\xa6;\xba\x82$\xa5\x89\xec\xc5
)E\xd0\x02\x00\x00\u07d4\xdf\x1f\xa2\xe2\x0e1\x98^\xbe,\x0f\f\x93\xb5L\x0f\xb6z&K\x89\n\u05ce\
xbcZ\xc6
\x00\x00\u07d4\xdf!\x1c\xd2\x12\x88\xd6\xc5o\xaef\xc3\xffTb]\u0531T'\x89\x87\x86\xcdvN\x1f,\x0
0\x00\u07d4\xdf#k\xf6\xab\xf4\xf3)7\x95\xbf\f(q\x8f\x93\u3c73k\x89Hz\x9a0E9D\x00\x00\u07d4\xd
f1\x02_VI\xd2\xc6\xee\xa4\x1e\u04fd\xd3G\x1ay\x0fu\x9a\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe
0\x94\xdf7\xc2.`:\xed\xb6\nbrS\xc4}\x8b\xa8f\xf6\xd9r\x8a\x05\x15\n\xe8J\x8c\xdf\x00\x00\x00\u07
d4\xdf;r\u017dq\u0501N\x88\xa6#!\xa9=@\x11\xe3W\x8b\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u0
7d4\xdf?W\xb8\xeed4\xd0G\"=\xeft\xb2\x0fc\xf9\xe4\xf9U\x89\r\x94b\xc6\xcbKZ\x00\x00\u07d4\xd
fD\xc4\u007f\xc3\x03\xacv\xe7O\x97\x19L\xcag\xb5\xbb<\x02?\x89

\t\xc5\u023fo\xdc\x00\x00\u07d4\xdfG\xa6\x1brSQ\x93\xc5a\xcc\xccu\xc3\xf3\xce\b\x04\xa2\x0e\x89\x15\x93\\\vN=x\x00\x00\u07d4\xdfG\xa8\xef\x95\xf2\xf4\x9f\x8eoX\x18AT\x14]\x11\xf7'\x97\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\xdfS\x003F\xd6\\^zdk\xc04\xf2\xb7\xd3/\xcb\xe5j\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xdfW5:\xaf\xf2\xaa\xdb\n\x04\xf9\x01N\x8d\xa7\x88N\x86X\x9c\x89\bH\x86\xa6nO\xb0\x00\x00\u07d4\xdf`\xf1\x8c\x81*\x11\xedN'v\xe7\xa8\x0e\xcf^S\x05\xb3\u05890\xca\x02O\x98{\x90\x00\x00\u07d4\xdfd\x85\xc4)z\xc1R\xb2\x89\xb1\x9d\xde2\xc7~\xc4\x17\xf4}\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xdff\n\x91\u06b9\xf70\xf6\x19\rP\xc89\x05aP\aV\u0289lk\x93[\x8b\xbd@\x00\x00\u07d4\xdfn\xd6\x00jj\xbe\x88n\xd3=\x95\xa4\xde(\xfc\x12\x189'\x891T\xc9r\x9d\x05x\x00\x00\u07d4\u07c5\x10y>\xee\x81\x1c-
\xab\x1c\x93\xc6\xf4G?0\xfb\xef[\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u07cdH\xb1\xeb\a\xb3\xc2\x17y\x0el-
\xf0M\xc3\x19\xe7\xe8H\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\u07e6\xb8\xb8\xad1\x84\xe3W\xda()Q\u05d1a\u03f0\x89\xbc\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\u07ef1\xe6\"\xc0=\x9e\x18\xa0\u0778\xbe`\xfb\xe3\xe6a\xbe\n\x8a\x02\x1e\x17\x1a>\xc9\xf7,\x00\x00\u07d4\u07f1bn\xf4\x8a\x1d}uR\xa5\xe0)\x8f\x1f\xc2:;H-
\x89\\\xe8\x95\u0754\x9e\xfa\x00\x00\xe0\x94\u07f4\u052d\xe5/\u0301\x8a\xccz,k\xb2\xb0\x02$e\x8fx\x8a\x01\xa4
\xdb\x02\xbd}X\x00\x00\u07d4\u07fdB2\xc1|@z\x98\r\xb8\u007f\xfb\u036060\xe5\xc4Y\x89\x1d\xfc\u007f\x92I#S\x00\x00\u07d4\xdf\xcb\xdf\tEN\x1a^J@\xd3\xee\xf7\xc5\xcf\x1c\xd3\u0794\x86\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xdf\xdb\xce\xc1\x01K\x96\xda!X\xcaQ>\x9c\x8d;\x9a\xf1\xc3\u0409lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xdf\xde\xd2WK'\xd1a:}\x98\xb7\x15\x15\x9b\r\x00\xba\xab(\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xdf\xdfC9<d\x9c\xae\xbe\x1b\xb1\x80Y\xde\xcb9\xf0\x9f\xb4\xe8\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xdf\xe3\xc5*\x92\xc3\x03\x96\xa4\xe3:P\x17\r\xc9\x00\xfc\xf8\xc9\u03c9\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\xdf\xe5I\xfe\x840\xe5R\xc6\xd0|\u00f9,\xcdC\xb1/\xb5\x0f\x89\x04\x88u\xea\xf6V*\x00\x00\u07d4\xdf\xe9)\xa6\x1c\x1b8\xed\xdb\xe8,%\xc2\xd6u<\xb1\xe1-h\x89\x15\xd1\xcfAv\xae\xba\x00\x00\u07d4\xdf\xf1\xb2
\xde=\x8e\x9c\xa4\xc1\xb5\xbe4\xa7\x99\xbc\xde\xd4\xf6\x1c\x89\x14\xe4\xe3S\xea9B\x00\x00\xe0\x94\xdf\xf4\x00y1xe\x93\xb2)\xef\u555f:N!\x9eQ\xaf\x8a\x01\n\xfc\x1a\xde;N\xd4\x00\x00\u07d4\xdf\xfc\xeaT!\xec\x15\x90\fn\xcf\xc7w\x18N\x14\x0e
\x9e$\x89\x01\x15G8$4N\x00\x00\u07d4\xe0\x01\xab\xa7|\x02\xe1r\bl\x19P\xff\xfb\u02a3\v\x83H\x8f\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xe0\x04\x84x\x8d\xb5\x0f\u01a4\x8e7\x9d\x12>P\x8b\x0fnZ\xb1\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe0\x06\x04b\xc4\u007f\xf9g\x9b\xae\xf0qY\xca\xe0\x8c)\xf2t\xa9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe0\r\x15;\x106\x91C\xf9\u007fT\xb8\xd4\xca\"\x9e\xb3\xe8\xf3$\x89\b=lz\xabc`\x00\x00\u07d4\xe0\x12\xdbE8'\xa5\x8e\x16\xc16V\b\xd3n\xd6Xr\x05\a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe0\x15G\xbaB\xfc\xaf\xaf\x93\x93\x8b\xec\xf7i\x9ft)\n\xf7O\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe0\x16\xdc\x13\x8e%\x81[\x90\xbe?\xe9\xee\xe8\xff\xb2\xe1\x05bO\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xe0\x18Y\xf2B\xf1\xa0\xec`/\xa8\xa3\xb0\xb5v@\xec\x89\a^\x89\x1e\x16,\x17{\xe5\xcc\x00\x00\xe0\x94\xe0
\xe8cb\xb4\x87u(6\xa6\xde\v\xc0,\xd8\u061a\x8bj\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xe0#\xf0\x9b(\x87a,|\x9c\xf1\x98\x8e::`+3\x94\u0249lk\x93[\x8b\xbd@\x00\x00\u07d4\xe0'\"\x13\xe8\xd2\xfd>\x96\xbdb\x17\xb2KK\xa0\x1bapy\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xe0+t\xa4v(

\xbe1[\x1fv\xb3\x15\x05J\xd4J\xe9qo\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xe02
\u0197\xbc\u048f&\xef\vt@J\x8b\xeb\x06\xb2\xba{\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xe
05/\u07c1\x9b\xa2e\xf1L\x06\xa61\\J\xc1\xfe\x13\x1b.\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe0
8\x8a\xed\xdd?\xe2\xadV\xf8WH\xe8\x0eq\n4\xb7\xc9.\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07
d4\xe0<\x00\xd0\x03\x88\xec\xbfO&=\n\xc7x\xbbA\xa5z@\u064966\xc9yd6t\x00\x00\u07d4\xe0I
\xdcn\xcc\x1dn\xcc\bO\x88\xaa\n\xf5\u06d7\xbf\x89:\x89\t\xdd\xc1\xe3\xb9\x01\x18\x00\x00\u07d
4\xe0Ir\xa8<\xa4\x11+\xc8q\xc7-
J\xe1al/\a(\u06c9\x0e\x81\xc7\u007f)\xa3/\x00\x00\u07d4\xe0O\xf5\xe5\xa7\u2bd9]\x88W\xce\x02\
x90\xb5:+\x0e\xda]\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xe0P)\xac\xeb\axg[\xef\x17A\xab,\u0
493\x1e\xf7\xc8K\x8a\x01\x0f\r\xba\xe6\x10\tR\x80\x00\u07d4\xe0V\xbf?\xf4\x1c&%o\xefQqf\x12\
xb9\u04da\u0799\x9c\x89\x05k\xe7W\xa1.\n\x80\x00\u07d4\xe0a\xa4\xf2\xfcw\xb2\x96\u045a\xda
#\x8eI\xa5\u02ce\xcb\xfap\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xe0f>\x8c\xd6g\x92\xa6A\xf
5nP\x03f\x01G\x88\x0f\x01\x8e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe0f\x8f\xa8,\x14\xd6\xe8\xd
9:S\x11>\xf2\x86/\xa8\x15\x81\xbc\x89//9\xfclT\x00\x00\x00\u07d4\xe0i\xc0\x173R\xb1\v\xf6\x83G
\x19\xdb[\xed\x01\xad\xf9{\xbc\x89\x01\x064\xf8\xe52;\x00\x00\u07d4\xe0l)\xa8\x15\x17\xe0\u050
7\xb6\u007f\xb0\xb6\xaa\xbcOW6\x83\x88\x89\x15\xbeat\xe1\x91.\x00\x00\u07d4\xe0l\xb6)G\x04\
xee\xa7C|/\xc3\xd3\as\xb7\xbf8\x88\x9a\x89\x01\x16\xdc:\x89\x94\xb3\x00\x00\u07d4\xe0q7\xae\r
\x11m\x0353\xc4\uad16\xf8\xa9\xfb\tV\x9c\x89K\xe4\xe7&{j\xe0\x00\x00\xe0\x94\xe0v\xdb0\xabH
oy\x19N\xbb\xc4]\x8f\xab\x9a\x92B\xf6T\x8a\x01\x06`~4\x94\xba\xa0\x00\x00\u07d4\xe0~\xbb\xc
7\xf4\xdaAnB\xc8\xd4\xf8B\xab\xa1b3\xc1%\x80\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe0\x81\xca
\x1fH\x82\xdb`C\u0569\x19\a\x03\xfd\xe0\xab;\xf5m\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\x
e0\x83\xd3Hc\xe0\xe1\u007f\x92ky(\xed\xff1~\x99\x8e\x9cK\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\
u07d4\xe0\x8b\x9a\xbak\xd9\u048b\xc2\x05gy\xd2\xfb\xf0\xf2\x85Z=\x9d\x89lk\x93[\x8b\xbd@\x0
0\x00\u07d4\xe0\x8b\u009c+H\xb1i\xff+\xdc\x16qLXnl\xb8\\\u03c9\x01\x15\x8eF\t\x13\xd0\x00\x0
0\u07d4\xe0\x8c`11\x06\xe3\xf93O\xe6\xf7\xe7bM!\x110\xc0w\x89\x02+\x1c\x8c\x12'\xa0\x00\x00
\u07d4\xe0\x9ch\xe6\x19\x98\xd9\xc8\x1b\x14\xe4\xee\x80+\xa7\xad\xf6\xd7L\u06c9\xd8\xd7&\xb
7\x17z\x80\x00\x00\u07d4\xe0\x9f\xeauZ\xee\x1aD\xc0\xa8\x9f\x03\xb5\u07b7b\xba3\x00o\x89;\x
a2\x89\xbc\x94O\xf7\x00\x00\xe0\x94\xe0\xa2T\xac\t\xb9r[\xeb\xc8\xe4`C\x1d\xd0s.\xbc\xab\xbf\x
8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xe0\xaai6UU\xb7?(#3\xd1\xe3\f\x1b\xbd\a(T\xe8\x8a
\x01{x\x83\xc0i\x16`\x00\x00\u07d4\xe0\xba\u064e\ue598\xdb\xf6\xd7`\x85\xb7\x92=\xe5uN\x90m
\x89\t\r\x97/22<\x00\x00\u07d4\xe0\u012b\x90r\xb4\xe6\xe3eJI\xf8\xa8\xdb\x02jK3\x86\xa9\x89lk\
x93[\x8b\xbd@\x00\x00\u07d4\xe0\u0380\xa4a\xb6H\xa5\x01\xfd\v\x82F\x90\u0206\x8b\x0eM\xe
8\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xe0\xcfi\x8a\x053'\xeb\xd1k}w\x00\t/\xe2\xe8T$F\
x89\x05*4\u02f6\x1fW\x80\x00\xe0\x94\xe0\xd21\xe1D\xec\x91\a8l|\x9b\x02\xf1p,\xea\xa4\xf7\x00
\x8a\x01\x0f\r\xba\xe6\x10\tR\x80\x00\u07d4\xe0\xd7kqf\xb1\xf3\xa1+@\x91\xee+)\u078c\xaa}\a\u
06c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xe0\xe0\xb2\xe2\x9d\xdes\xafu\x98~\xe4Dl\x82\x9a\x18\x9c
\x95\xbc\x89\b\x13\xcaV\x90m4\x00\x00\xe0\x94\xe0\xe9xu=\x98/\u007f\x9d\x1d#\x8a\x18\xbdH\x
89\xae\xfeE\x1b\x8a\x02\r\u058a\xaf2\x89\x10\x00\x00\u07d4\xe0\xf3r4|\x96\xb5_}C\x06\x03K\xe
b\x83&o\xd9\tf\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xe0\xf9\x03\xc1\xe4\x8a\xc4!\xabHR\x8
f=J&H\b\x0f\xe0C\x897\b\xba\xed=h\x90\x00\x00\u07d4\xe0\xff\v\xd9\x15D9\u0125\xb7#>)\x1d}\x
86\x8a\xf5?3\x89\x15y!jQ\xbb\xfb\x00\x00\xe0\x94\xe1\n\xc1\x9cTo\xc2T|a\xc19\xf5\xd1\xf4Zff\u0
570\x8a\x01\x02\xdao\xd0\xf7:<\x00\x00\xe0\x94\xe1\fT\x00\x88\x11?\xa6\xec\x00\xb4\xb2\u0202

O\x87\x96\xe9n\u010a2\x0fE\t\xab\x1e\xc7\xc0\x00\x00\xe0\x94\xe1\x17:$})\xd8#\x8d\xf0\x92/M\xf2Z\x05\xf2\xafw\u00ca\bx\xc9]V\x0f0G\x80\x00\xe0\x94\xe1 >\xb3\xa7#\xe9\x9c\"
\x11|\xa6\xaf\xebf\xfaBOa\x8a\x02\x00\uf49e2V\xfe\x00\x00\xe0\x94\xe11\xf8~\xfc^\xf0~C\xf0\xf2\xf4\xa7G\xb5Q\xd7P\xd9\xe6\x8a\x04<%\xe0\xdc\xc1\xbd\x1c\x00\x00\u07d4\xe13N\x99\x83y\xdf\xe9\x83\x17pby\x1b\x90\xf8\x0e\xe2-
\x8d\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xe15@\xec\xee\x11\xb2\x12\xe8\xb7u\u070eq\xf3t\xaa\xe9\xb3\xf8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe1;=+\xbf\u073c\x87r\xa23\x15rL\x14%\x16|V\x88\x897\xf3y\x14\x1e\xd0K\x80\x00\u07d4\xe1D=\xbd\x95\xccA#\u007fa:HEi\x88\xa0Oh2\x82\x89\xd8\xd8X?\xa2\xd5/\x00\x00\u07d4\xe1F\x17\xf6\x02%\x01\xe9~{>-\x886\xaaa\xf0\xff-\xba\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe1I\xb5rl\xafm^\xb5\xbf*\xccA\xd4\xe2\xdc2\x8d\u1089i*\xe8\x89p\x81\xd0\x00\x00\xe0\x94\xe1T\xda\xea\xdbTX8\xcb\u01aa\fUu\x19\x02\xf5(h*\x8a\x01\n\xfc\x1a\xde;N\xd4\x00\x00\u07d4\xe1l\xe3Ya\xcdt\xbdY\r\x04\u012dJ\x19\x89\xe0V\x91\u0189\a\xea(2uw\b\x00\x00\u07d4\xe1r\xdf\xc8\xf8\f\xd1\xf8\u03459\xdc&\b
\x14\xf5\xa8\xe3\u8262\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xe1w\xe0\xc2\x01\xd35\xba9V\x92\x9cW\x15\x88\xb5\x1cR#\xae\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe1x\x12\xf6l^e\x94\x1e\x18lF\x92+n{/\x0e\xebF\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xe1\x80\u079e\x86\xf5{\xaf\xac\u05d0O\x98&\xb6\xb4\xb2c7\xa3\x89-
\x04\x1dpZ,`\x00\x00\xe0\x94\xe1\x92H\x9b\x85\xa9\x82\xc1\x882F\xd9\x15\xb2)\xcb\x13
\u007f8\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xe1\x95<n\x97X\x14\xc5q1\x1c4\xc0\xf6\xa9\x9c\xdfH\xab\x82\x89\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\xe1\xae\x02\x9b\x17\xe3s\xcd\xe3\xdeZ\x91R
\x1a\x14\xca\xc4\xe1\x19\x89\x05kU\xaeX\xca@\x00\x00\u07d4\u1cac\xa1T\xb8\xe0vlN\xba0\xbc\x10\xc7\xf3P6\xf3h\x89\x01\x15G8$4N\x00\x00\u07d4\u1cdb\x88\u0650\r\xbcJl\xdcH\x1e\x10`\b\n\x8a\xec<\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe1\xb62\x01\xfa\xe1\xf1)\xf9\\z\x11k\xd9\xdd\xe5\x15\x9cl\u068a\x04\xd6\x05s\xa2\xf0\xc9\xef\x00\x00\u07d4\u1feaZE\xc5\x04B\x89#\u0126\x11\x92\xa5[\x14\x00\xb4]\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\xe1\xc6\a\xc0\xa8\xa0`\u068f\x02\xa8\xeb8\xa0\x13\xea\x8c\xda[\x8c\x89+\xa3\x9e\x82\xed]t\x00\x00\xe0\x94\xe1\u02c3\xec^\xb6\xf1\xee\xb8^\x99\xb2\xfcc\x81/\u0795q\x84\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xe1\xd9\x1b\tT\xce\xde\"\x1do$\u01d8_\u0159e\xfb\x98\xb8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe1\u07f5\u0309\x0e\u8c87~\x88]&|%a\x87\xd0\x19\xe6\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xe1\xe8\xc5\v\x80\xa3R\xb2@\xcesB\xbb\xfd\xf5i\f\xc8\xcb\x14\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\xe1\xf6>\xbb\xc6,{tD\x04\x0e\xb9\x96#\x96Ovg\xb3v\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xe2\x06\xfbs$\xe9\u07b7\x9e\x19\x904\x96\u0596\x1b\x9b\xe5f\x03\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xe2\aW\x8e\x1fM\u06cf\xf6\u0546{9X-
q\xb9\x81*\u0149\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\xe2\b\x81*h@\x98\xf3\xdaN\xfej\xba%bV\xad\xfe?\xe6\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe2\tT\xd0\xf4\x10\x8c\x82\xd4\u0732\x14\x8d&\xbb\xd9$\xf6\xdd$\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xe2\v\xb9\xf3\x96d\x19\xe1K\xbb\xaa\xaag\x89\xe9$\x96\u03e4y\x89\xbb\xd8%\x03\aRv\x00\x00\u07d4\xe2\r\x1b\xcbq(m\xc7\x12\x8a\x9f\xc7\xc6\xed\u007fs8\x92\xee\xf5\x896d\xf8\xe7\xc2J\xf4\x00\x00\u0794\xe2\x19\x12\x15\x98?3\xfd3\xe2,\u0522I\x00T\xdaS\xfd\u0708\xdbD\xe0I\xbb,\x00\x00\u07d4\xe2\x19\x8c\x8c\xa1\xb3\x99\xf7R\x15a\xfdS\x84\xa7\x13/\xbaHk\x897\b\xba\xed=h\x90\x00\x00\xe0\x9

4\xe2\x1cw\x8e\xf2\xa0\xd7\xf7Q\xea\x8c\aM\x1f\x81\"C\x86>N\x8a\x01\x1f\xc7\x0e,\x8c\x8a\xe1\x80\x00\xe0\x94\xe2)\xe7F\xa8?,\xe2S\xb0\xb0>\xb1G$\x11\xb5~W\x00\x8a\x016\x9f\xb9a(\xacH\x00\x00\u07d4\xe2+
\xc7x\x94F;\xafwL\xc2V\u057d\u06ff}\xdd\t\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe20\xfe\x1b\xff\x03\x18m\x02\x19\xf1]LH\x1b}Y\xbe(j\x89\x01\xfdt\x1e\x80\x88\x97\x00\x00\u07d4\xe27\xba\xa4\xdb\u0252n2\xa3\xd8]\x12d@-
T\xdb\x01/\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe2A\t\xbe/Q=\x87I\x8e\x92j(d\x99uO\x9e\u051e\x890\x0e\xa8\xad\x1f'\xca\x00\x00\u07d4\xe2Fh<\u025d\xb7\u0125+\u02ec\xaa\xb0\xb3/k\xfc\x93\u05c9lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe2Z\x16{\x03\x1e\x84am\x0f\x01?1\xbd\xa9]\xcccP\xb9\x8a\x02<upr\xb8\xdd\x00\x00\x00\u07d4\xe2[\x9fv\xb8\xad\x02?\x05~\xb1\x1a\xd9BW\xa0\x86.N\x8c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe2fW\xf0\xed
\x1e\xa29,\x92\"\xb8\np\x03`\x8d\xdf0\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xe2k\xf3\"wN\x18(\x87i\xd6~1\a\u07b7Dw\a\xb8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe2r\x8a>\x8c*\xaa\u0243\xd0]\u0187st\xa8\xf4F\xee\xe9\x89\n\xb6@9\x12\x010\x00\x00\u07d4\xe2\x8b\x06\"Y\xe9n\xeb<\x8dA\x04\x94?\x9e\xb3%\x89<\xf5\x89Hz\x9a0E9D\x00\x00\xe0\x94\u237c\x8e\xfd^Ajv.\xc0\xe0\x18\x86K\xb9\xaa\x83({\x8a\x051\xf2\x00\xab>\x03\n\x80\x00\u07d4\xe2\x90K\x1a\xef\xa0V9\x8bb4\xcb5\x81\x12\x88\xd76\xdbg\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\u274a\xe4R\xdc\xf3\xb6\xacd^c\x04\t8UQ\xfa\xae\n\x89\x04Z\r\xa4\xad\xf5B\x00\x00\u07d4\xe2\xbb\xf8FA\xe3T\x1fl3\xe6\xedh:cZp\xbd\xe2\xec\x89\x1bA<\xfc\xbfY\xb7\x80\x00\u07d4\xe2\xcf6\n\xa22\x9e\xb7\x9d+\xf7\xca\x04\xa2z\x17\xc52\xe4\u0609\x05\x87\x88\u02d4\xb1\xd8\x00\x00\u07d4\xe2\xdf#\xf6\xea\x04\xbe\xcfJ\xb7\x01t\x8d\xc0\x961\x84U\\\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xe2\xe1\\`\xdd8\x1e:K\xe2Pq\xab$\x9aL\\Rd\u0689\u007fk\u011b\x81\xb57\x00\x00\u07d4\xe2\xe2nN\x1d\xcf0\xd0H\xccn\u03ddQ\xec\x12\x05\xa4\xe9&\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xe2\xeei\x1f#~\xe6R\x9beW\xf2\xfc\xdd=\xcf\fY\xecc\x8a\x01'r\x9c\x14h|
\x00\x00\u07d4\xe2\xef\xa5\xfc\xa7\x958\xce`h\xbf1\xd2\xc5\x16\xd4\xd5<\b\xe5\x89\a\x1c\xc4\b\xdfc@\x00\x00\xe0\x94\xe2\xef\u0429\xbc@~\xce\x03\xd6~\x8e\xc8\xe9\u0483\xf4\x8d*I\x8a\x02\x99\xb3;\xf9\u0144\xe0\x00\x00\u07d4\xe2\xf4\r5\x8f^?\xe7F>\xc7\x04\x80\xbd.\u04d8\xa7\x06;\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xe2\xf98=X\x10\xea{C\x18+\x87\x04\xb6+'\xf5\x92]9\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xe2\xff\x9e\xe4\xb6\xec\xc1AA\xcct\xcaR\xa9\xe7\xa2\xee\x14\xd9\b\x89K\xe4\xe7&{j\xe0\x00\x00\xe0\x94\xe3\x02\x12\xb2\x01\x1b\xb5k\xdb\xf1\xbc5i\x0f:N\x0f\xd9\x05\xea\x8a\x01\xb2\u07dd!\x9fW\x98\x00\x00\u07d4\xe3\x03\x16\u007f=I`\xfe\x88\x1b2\x80\n+J\xef\xf1\xb0\x88\u0509lk\x93[\x8b\xbd@\x00\x00\u07d4\xe3\x04\xa3/\x05\xa87btJ\x95B\x97o\xf9\xb7#\xfa1\xea\x89Ur\xf2@\xa3F
\x00\x00\u07d4\xe3\bCR\x04y7d\xf5\xfc\xbee\xebQ\x0fZtJeZ\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe3\t\x97L\xe3\x9d`\xaa\xdf.ig2Q\xbf\x0e\x04v\n\x10\x89\r\xc5_\xdb\x17d{\x00\x00\u07d4\xe3\x1bN\xef\x18L$\xab\t\x8e6\xc8\x02qK\xd4t=\xd0\u0509\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe3!\xbbJ\x94j\xda\xfd\xad\xe4W\x1f\xb1\\\x00C\u04de\xe3_\x89Udu8+L\x9e\x00\x00\u07d4\xe3&<\xe8\xafm\xb3\xe4gXE\x02\xedq\t\x12^\xae\"\xa5\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe3+\x1cG%\xa1\x87TI\u93d7\x0e\xb3\xe5@b\xd1X\x00\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe3/\x95vmW\xb5\xcdK\x172\x89\u0587o\x9edU\x81\x94\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xe38@\u063c\xa7\u0698\xa6\xf3\u0416\xd8=\xe7\x8bp\xb7\x1e\xf8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe38\xe8Y\xfe.\x8c\x15UHH\xb7\\\xae\u0368w\xa0\xe82\x89a\xac\xff

\x81\xa7\x8a\xd4\x00\x00\u07d4\xe3=\x98\x02
\xfa\xb2Y\xafj\x1fK8\xcf\x0e\xf3\xc6\xe2\xea\x1a\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe3=\xf4\u0380\u0336*v\xb1+\xcd\xfc\xec\xc4b\x89\x97:\xa9\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xe3?\xf9\x87T\x1d\xde\\\xde\u0a29m\xcc?3\xc3\xf2L\u008a*Z\x05\x8f\u0095\xed\x00\x00\x00\u07d4\xe3A\v\xb7U|\xf9\x1dy\xfai\xd0\xdf\xea\n\xa0u@&Q\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe3Ad-
@\u04af\xce.\x91\a\xc6py\xacz&`\bl\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xe3TS\xee\xf2\xcc<z\x04M\n\xc14\xbaaY\b\xfa\x82\xee\x89\a\xff\x1c\xcbua\xdf\x00\x00\u07d4\xe3j\x8e\xa8\u007f\x1e\x99\xe8\xa2\xdc\x1b&\b\xd1ff|\x9d\xfa\x01\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xe3q'\x01a\x9c\xa7b<U\xdb:\n\xd3\x0e\x86}\xb0\x16\x8b\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xe3\u007f_\xdcn\xc9}/\x86j\x1c\xfd\r:M\xa48{\"\xb5\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u31cf\x91\u0286\x05?\xce\xd5DF\x86\xa30\xe0\x9c\u00c8\xfb\x89\n\x84Jt$\xd9\xc8\x00\x00\u07d4\u32d1\xb3Q\x90\xb6\xd9\xde\xed\x02\x1c0\xaf\tK\x95?\u072a\x89\x01\u03afy[k\x86\x00\x00\u07d4\xe3\x8e\xf2\x8a^\u0644\xa7\xdb$\xa1\xaex-
\xfb\x87\xf3\x97\xdf\u0189\a\xc0\x86\x0eZ\x80\xdc\x00\x00\u07d4\xe3\x92U\t\xc8\u0432\xa6s\x8c_jr\xf3S\x14I\x12H\u03896\xe9\xa8f\x9aDv\x80\x00\xe0\x94\xe3\x93=a\xb7}\xcd\xc7\x16@\u007f\x82P\xbc\x91\xe4\xff\xae\xb0\x9d\x8a\x12V\x98l\x95\x89\x1c
\x00\x00\u07d4\xe3\x95\x1d\xe5\xae\xfa\xf0E\x87h\xd7t\xc2T\xf7\x15w5\xe5\x05\x89V\xc9]\xe8\xe8\xca\x1d\x00\x00\u07d4\xe3\x99\xc8\x1a\x1dp\x1bD\xf0\xb6o3\x99\xe6k'Z\xaa\xf8\xc1\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe3\x9b\x11\xa8\xab\x1f\xf5\xe2.Z\xe6Qr\x14\xf7<[\x9bU\u0709lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe3\x9eF\xe1]\"\xceV\xe0\xc3/\x18w\xb7\u0462d\u03d4\xf3\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xe3\xa4b\x1bf\x00E\x88\xe3\x12\x06\xf7\x18\xcb\x00\xa3\x19\x88\x9c\xf0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe3\xa4\xf8<9\xf8Z\xf9\u0231\xb3\x12\xbf\xe5\xfc4#\xaf\xa64\x89\x01\x8d\x99?4\xae\xf1\x00\x00\u07d4\u3a1a\x19'\xccN-
C\xfb\u0361\xe4\x14\xd3$\xa7\xd9\xe0W\x89\v#\xe2\xa96\xde\xc6\x00\x00\u07d4\xe3\xab<\xa9\xb8p\xe3\xf5HQs\x06\xbb\xa4\xde%\x91\xaf\xaf\u0089A\x0e4\xae\u030c\xd3\x00\x00\u07d4\xe3\xb3\xd2\u027fW\v\xe6\xa2\xf7*\u0721\x86,1\t6\xa4<\x89\x05m*\xa3\xa5\xc0\x9a\x00\x00\u07d4\xe3\xc0\xc1(2z\x9a\xd8\x01H\x13\x9e&\x97sB\x8ec\x8c\xb0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe3\xc8\x12sz\xc6\x06\xba\xf7R*\xd8\x17B\x8a6\x05\x0ez4\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xe3\xcf\xfe#\x9cd\xe7\xe2\x03\x88\xe6\"\x11s\x910\x1b)\x86\x96\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xe3\xd3\ua899\x88xeV\x9e\x88\xbe!\x9b\xe5\a\x18\x9b\xe1\u0249\x18\xbao\xa9.\x93\x16\x00\x00\u07d4\xe3\u063fN\xfe\x84\xb1am\x1b\x89\xe4'\xdd\xc6\u0203\x06\x85\xae\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe3\xd9\x15\xed\xa3\xb8%\xd6\xeeJ\xf92\x8d2\xac\x18\xad\xa3T\x97\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xe3\xdaO2@\x84L\x9bc#\xb4\x99i!
q\"EC\x99\x8a\x02q\x90\xa9R\xdfK\xe5\x80\x00\u07d4\xe3\xeb,\n\x13*ROr\xcc\xc0\xd6\x0f\xee\x8bAh]9\xe2\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xe3\xec\x18\xa7N\xd48U@\x9a&\xad\xe7\x83\r\xe8\xe4&\x85\xef\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4\xe3\xec\xe1\xf62q\x1d\x13\xbf\xff\xa1\xf8\xf6\x84\bq\xeeX\xfb'\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xe3\xf8\v@\xfb\x83\xfb\x97\xbb\rR0\xafOn\u055b\x1c|\u0209Hz\x9a0E9D\x00\x00\u07d4\xe3\xff\xb0,\xb7\xd9\xeaRCp\x16\x89\xaf\xd5\xd4\x17\xd7\xed.\u0389\x04:w\xaa\xbd\x00x\x00\x00\u07d4\xe4\x00\xd6Q\xbb?-
#\xd5\xf8I\xe6\xf9-
\x9cW\x95\xc4:\x8a\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\xe4\x06\xf5\xddr\u02b6m\x8an\xcb\u0

5bf\xb4\x94\xa7\xb6\xb0\x9a\xfe\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xe4\b\xaa\x99\x83S\a\ue926\xc5\xeb\x80\x1f\xe6\x94\x11\u007fp}\x89\x1b\x
1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xe4\b\xfc\ua879\x8f<d\x0fH\xfc\xba9\xf0V\x06mc\b\x8a\x02\
x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xe4\n|\x82\xe1WT\n\v\x00\x90\x1d\xbb\x86\xc7\x16\xe1\
xa0b\u0689\x02\xb3\x1d$%\xf6t\x00\x00\u07d4\xe4\x1a\xea%\v\x87}B:c\xba+\xce/:a\xc0$\x8dV\x
89\x0e\x189\x8ev\x01\x90\x00\x00\u07d4\xe40\xc0\x02O\xdb\xf7:\x82\xe2\x1f\xcc\xf8\xcb\u04118
B\x1c!\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xe42I\x12\xd6N\xa3\xae\xf7k</\xf9\u07c2\xc7\x
e1:\u9449lk\x93[\x8b\xbd@\x00\x00\u07d4\xe46\x8b\xc1B\v5\xef\u0695\xfa\xfb\xc70\x90R\x19\x1
6\xaa4\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xe47\xac\xbe\x0fb'\xb0\xe3o6\xe4\xbc\xf7\xcfa
35\xfbh\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xe4Krd\u0743k\ue387\x97\x03@\xed+\x9a\xed\x8e\u0425\x8a\x018\xe7\xfa\xa0
\x1a\x80:\x00\x00\u07d4\xe4N\xa5\x10c@QT\xaa\xe76\xbe+\xf1\xee;\x9b\xe69\xae\x89\xd8\xd7&
\xb7\x17z\x80\x00\x00\u07d4\xe4bU\x01\xf5+z\xf5+\x19\xeda.\x9dT\xfd\xd0\x06\xb4\x92\x89\vZ\x
90ZV\xdd\xd0\x00\x00\u07d4\xe4qYV\xf5/\x150n\xe9Pk\xf8+\xcc\xc4\x06\xb3\x89^\x89\x0e\xe7\x
9dOH\xc5\x00\x00\x00\u07d4\xe4\u007f\xba\xed\x99\xfc \x99b`N\xbd
\xe2@\xf7OE\x91\xf1\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe4\x82\xd2U\xed\xe5k\x04\xc3\xe8\xd
f\x15\x1fV\xe9\xcab\xaa\xa8\u0089\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xe4\x8ee\x12T!\x88\
rB\xbd\xf1\x01\x8a\xb9w\x8d\x96\x92\x8f?\x89\u3bb5sr@\xa0\x00\x00\u07d4\u4481\x8a\xa6\x84\
xe5\xa6vV\x1br]B\xf3\xccV\xaeQ\x98\x89+^:\xf1k\x18\x80\x00\x00\u07d4\xe4\x996\xa9*\x8c\xcfq\
x0e\xaa\xc3B\xbcEK\x9b\x14\xeb\ucc49lk\x93[\x8b\xbd@\x00\x00\u07d4\xe4\x9a\xf4\xf3J\u06a23
\v\x0eI\xdct\xec\x18\xab/\x92\xf8'\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u46e0\u0356\x81lF\aw<\xf
8\xa5\x97\v\xb5\xbc\x16\xa1\xe6\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\u07d4\xe4\xa4~93$l?\xd6)y\
xa1\xea\x19\xff\u07ccr\xef7\x89\b\t\xb3\x83\xea}~\x80\x00\u07d4\u4dae\"\xc7s_[\x89\xf3M\xd7z\u
0417_\n\u0311\x81\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe4\xca\nR8VM\xfc\x91\xe8\xbf\"\xba\
xde)\x01a\x9a\x1c\u050965\u026d\xc5\u07a0\x00\x00\u07d4\xe4\xca\xfbr\u007f\xb5\u01b7\v\xb2
u3\xb8\xa9\xcc\xc9\xefh\x88\xe1\x89\x10I{\xf4\xafL\xaf\x80\x00\u07d4\xe4\xdc\"\xedY[\xf0\xa37\x
c0\x1e\x03\xcck\xe7D%_\xc9\xe8\x89\nZ\xa8P\t\xe3\x9c\x00\x00\u07d4\xe4\xfb&\xd1\xca\x1e\xec
\xba=\x82\x98\xd9\xd1H\x11\x9a\u00bb\xf5\x80\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xe4\xf
c\x13\xcf\u02ec\x1b\x17\xcew\x83\xac\xd4#\xa8E\x94?k:\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07
d4\xe5\vFJ\xc9\xde5\xa5a\x8b|\xbf%Ft\x18+\x81\xb9~\x89\xdeB\xee\x15D\u0750\x00\x00\u07d4\
xe5\x10,;q\x1b\x81\x03D\x19t\x19\xb1\u034apY\xf1>2\x89\x10CR\x8d\t\x84i\x80\x00\xe0\x94\xe5\
x10\xd6y\u007f\xba=f\x93\x83Z\x84N\xa2\xadT\x06\x91\x97\x1b\x8a\x03\xae9\xd4s\x83\xe8t\x00\
x00\u07d4\xe5\x14!\xf8\xee\"\x10\xc7\x1e\xd8p\xfea\x82v\u0215J\xfb\xe9\x89Hz\x9a0E9D\x00\x0
0\u07d4\xe5\x1e\xb8~\u007f\xb71\x1fR(\xc4y\xb4\x8e\u0247\x881\xacL\x89lk\x93[\x8b\xbd@\x00\
x00\u07d4\xe5!V1\xb1BH\xd4Z%R\x96\xbe\xd1\xfb\xfa\x030\xff5\x89G\x03\xe6\xebR\x91\xb8\x00
\x00\xe0\x94\xe5(\xa0\xe5\xa2g\xd6g\xe99:e\x84\xe1\x9b4\u071b\xe9s\x8a\x01/\x93\x9c\x99\xed\
xab\x80\x00\x00\u07d4\xe54%\xd8\xdf\x1f\x11\xc3A\xffX\xae_\x148\xab\xf1\xcaS\u03c9\x11t\xa5\
xcd\xf8\x8b\xc8\x00\x00\u07d4\xe5<hyb\x12\x03>No\x9c\xffV\xe1\x9cF\x1e\xb4T\xf9\x8965\u026
d\xc5\u07a0\x00\x00\u07d4\xe5A\x02SM\xe8\xf2>\xff\xb0\x93\xb3\x12B\xad;#?\xac\xfd\x89\xd8\x
d7&\xb7\x17z\x80\x00\x00\u07d4\xe5E\xee\x84\xeaH\xe5d\x16\x1e\x94\x82\u055b\xcf@j`,\xa2\x8
9dI\xe8NG\xa8\xa8\x00\x00\u07d4\xe5H\x1a\u007f\xedB\xb9\x01\xbb\xed
x\x9b\u052d\xe5\r_\x83\xb9\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe5Y\xb5\xfd3{\x9cUr\xa9\xbf\x9

e\x0f%!\xf7\xd4F\xdb\xe4\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe5\\\x80R\n\x1b\x0fu[\x9a,\xd3\xce!Ov%e>\x8a\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe5mC\x13$\xc9)\x11\xa1t\x9d\xf2\x92p\x9c\x14\xb7ze\u034a\x01\xbc\x85\xdc*\x89\xbb
\x00\x00\u07d4\xe5})\x95\xb0\xeb\xdf?<\xa6\xc0\x15\xeb\x04&\r\xbb\x98\xb7\u0189lk\x93[\x8b\xbd@\x00\x00\u07d4\u51f1j\xbc\x8at\b\x1e6\x13\xe1CB\xc03u\xbf\bG\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe5\x89\xfav\x98M\xb5\xec@\x04\xb4n\u8954\x92\xc3\aD\u0389\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\xe5\x8d\xd228\xeen\xa7\xc2\x13\x8d8]\xf5\x00\xc3%\xf3v\xbe\x89b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xe5\x95?\xeaIq\x04\xef\x9a\xd2\xd4\xe5\x84\x1c'\x1f\a5\x19\u0089&)\xf6n\fS\x00\x00\x00\xe0\x94\u5587\x97F\x8e\xf7g\x10\x1bv\x1dC\x1f\xce\x14\xab\xff\u06f4\x8a\x01\xb3\xd9i\xfaA\x1c\xa0\x00\x00\u07d4\xe5\x97\xf0\x83\xa4i\xc4Y\x1c=+\x1d,w'\x87\xbe\xfe'\xb2\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\u07d4\xe5\x9b;\xd3\x00\x89?\x97#>\xf9G\xc4or\x17\xe3\x92\xf7\xe9\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe5\xa3e4<\xc4\xeb\x1ew\x03h\xe1\xf1\x14Jw\xb82\xd7\xe0\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xe5\xa3\xd7\xeb\x13\xb1\\\x10\x01w#m\x1b\xeb0\xd1~\xe1T
\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe5\xaa\v\x83;\xb9\x16\xdc\x19\xa8\xddh?\x0e\xde$\x1d\x98\x8e\xba\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\u5def\x14i\x86\xc0\xff\x8f\x85\xd2.l\xc34\a}\x84\xe8$\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe5\xb8&\x19l\x0e\x1b\xc1\x11\x9b\x02\x1c\xf6\xd2Y\xa6\x10\u0256p\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe5\xb9o\u026c\x03\xd4H\xc1a:\xc9\x1d\x15\x97\x81E\xdb\xdf\u0449\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\u5e40\u048e\xec\xe2\xc0o\xcal\x94s\x06\x8b7\u0526\xd6\xe9\x89%\xaf\u058c\xac+\x90\x00\x00\u07d4\u5eb4\xf0\xaf\u0629\u0463\x81\xb4Wa\xaa\x18\xf3\xd3\xcc\xe1\x05\x89Q\xbf\xd7\xc18x\xd1\x00\x00\u07d4\xe5\xbc\u020c;%on\xd5\xfeU\x0eJ\x18\x19\x8b\x943V\xad\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe5\xbd\xf3OL\xccH>L\xa50\xcc|\xf2\xbb\x18\xfe\xbe\x92\xb3\x89\x06\xd85\xa1\v\xbc\xd2\x00\x00\u07d4\xe5\u0713l\xcbR\xe1a\x19a\"\u03c7\xa3\x896\xe2\xc5\u007f4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe5\xe38\x00\xa1\xb2\xe9k\xde\x101c\n\x95\x9a\xa0\a\xf2nQ\x89Hz\x9a0E9D\x00\x00\u07d4\xe5\xe3~\x19@\x8f,\xfb\xec\x834\x9d\u0501S\xa4\xa7\x95\xa0\x8f\x89\u3bb5sr@\xa0\x00\x00\u07d4\xe5\xed\xc7>bo]4A\xa4U9\xb5\xf7\xa3\x98\u0153\xed\xf6\x89.\xe4IU\b\x98\xe4\x00\x00\u07d4\xe5\xed\xf8\x12?$\x03\xce\x1a\x02\x99\xbe\xcfz\xactM\a_#\x89\n\xdaUGK\x814\x00\x00\u07d4\xe5\xf8\xefm\x97\x066\xb0\u072aO
\x0f\xfd\xc9\xe7Z\xf1t\x1c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe5\xfb1\xa5\xca\xeej\x96\xde9;\xdb\xf8\x9f\xbee\xfe\x12[\xb3\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe5\xfb\xe3I\x84\xb67\x19o3\x1cg\x9d\f\fG\xd84\x10\xe1\x89llD\xfeG\xec\x05\x00\x00\u07d4\xe6\tU\xdc\v\xc1V\xf6\xc4\x18I\xf6\xbdwk\xa4K\x0e\xf0\xa1\x89\x10C\x16'\xa0\x93;\x00\x00\u07d4\xe6\nU\xf2\u07d9m\u00ee\xdbil\b\xdd\xe09\xb2d\x1d\xe8\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe6\x11[\x13\xf9y_~\x95e\x02\xd5\aEg\u06b9E\xcek\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xe6\x1f(\t\x15\xc7t\xa3\x1d\"<\xf8\f\x06\x92f\xe5\xad\xf1\x9b\x89/\xb4t\t\x8fg\xc0\x00\x00\u07d4\xe6/\x98e\a\x12\xeb\x15\x87S\xd8)r\xb8\u9723\xf6\x18w\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe6/\x9d|d\xe8\xe2cZ\xeb\x88=\xd7;\xa6\x84\xee|\x10y\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xe6>xt\x14\xb9\x04\x84x\xa5\a35\x9e\xcd\xd7\xe3dz\xa6\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\xe6FfXr\xe4\v\rz\xa2\xff\x82r\x9c\xaa\xba[\xc3\u8789\x15\

xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xe6N\xf0\x12e\x8dT\xf8\xe8`\x9cN\x90#\xc0\x9f\xe8e\xc8;\x89\x01\x84\x93\xfb\xa6N\xf0\x00\x00\u07d4\xe6On\x1dd\x01\xb5l\akd\xa1\xb0\x86}\v/1\rN\x89\x02\u02edq\xc5:\xe5\x00\x00\u07d4\xe6g\xf6R\xf9W\u008c\x0ef\u04364\x17\xc8\f\x8c\x9d\xb8x\x89\x9d\x92/RY\xc5\x00\x00\xe0\x94\xe6w\xc3\x1f\xd9\xcbr\x00u\u0724\x9f\x1a\xbc\xcdY\xec3\xf74\x8a\x01\xa6\u05be\xb1\xd4.\xe0\x00\x00\u07d4\xe6|,\x16e\u02038h\x81\x87b\x9fl\xe9\x9b`\xb2\u04fa\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xe6\x9al\xdb:\x8a}\xb8\xe1\xf3\f\x8b\x84\xcds\xba\xe0+\xc0\xf8\x8a\x03\x94\xfd\xc2\xe4R\xf6q\x80\x00\u07d4\xe6\x9d\x1c7\x8bw\x1e\x0f\xef\xf0Q\xdbi\xd9f\xacgy\xf4\xed\x89\x1d\xfaj\xaa\x14\x97\x04\x00\x00\u07d4\xe6\x9f\xcc&\xed\"_{.7\x984\xc5$\xd7\f\x175\u5f09lk\x93[\x8b\xbd@\x00\x00\u07d4\xe6\xa3\x01\x0f\x02\x01\xbc\x94\xffg\xa2\xf6\x99\xdf\xc2\x06\xf9\xe7gB\x89/\xa7\xcb\xf6dd\x98\x00\x00\u07d4\xe6\xa6\xf6\xddop\xa4V\xf4\xec\x15\xefz\xd5\xe5\u06f6\x8b\xd7\u0709\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe6\xb2\x0f\x98\n\xd8S\xad\x04\xcb\xfc\x88|\xe6`\x1ck\xe0\xb2L\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\u6cec?]M\xa5\xa8\x85}\v?0\xfcK+i+w\u05c9O%\x91\xf8\x96\xa6P\x00\x00\u07d4\xe6\xb9T_~\u0406\xe5R\x92F9\xf9\xa9\xed\xbb\xd5T\v>\x89\xcb\xd4{n\xaa\x8c\xc0\x00\x00\xe0\x94\xe6\xbc\xd3\n\x8f\xa18\xc5\xd9\xe5\xf6\xc7\xd2\u0680i\x92\x81-
\u034a7\x0e\xa0\xd4|\xf6\x1a\x80\x00\x00\u07d4\xe6\xc8\x1f\xfc\xec\xb4~\xcd\xc5\\\vq\xe4\x85_>^\x97\xfc\x1e\x89\x12\x1e\xa6\x8c\x11NQ\x00\x00\u07d4\xe6\xcb&\vqmL\n\xb7&\xee\xeb\a\xc8pr\x04\xe2v\xae\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe6\xcb?1$\xc9\xc9\xcc84\xb1'K\xc33dV\xa3\x8b\xac\x89\x17+\x1d\xe0\xa2\x13\xff\x00\x00\xe0\x94\xe6\xd2\"\t\xff\u0438u\t\xad\xe3\xa8\xe2\xefB\x98y\u02c9\xb5\x8a\x03\xa7\xaa\x9e\x18\x99\xca0\x00\x00\u07d4\xe6\u051f\x86\xc2(\xf4sg\xa3^\x88l\xaa\xcb'\x1eS\x94)\x89\x16^\xc0\x9d\xa7\xa1\x98\x00\x00\u07d4\xe6\xe6!\xea\xab\x01\xf2\x0e\xf0\x83k|\xadGFL\xb5\xfd<\x96\x89\x11!\x93B\xaf\xa2K\x00\x00\u07d4\xe6\xe8\x861{jf\xa5\xb4\xf8\x1b\xf1d\xc58\xc2d5\x17e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe6\u98ddu\x0f\xe9\x949N\xb6\x82\x86\xe5\xeab\xa6\x99x\x82\x89
\x86\xac5\x10R`\x00\x00\xe0\x94\xe6\xec\\\xf0\u011b\x9c1~\x1epc\x15\uf7b7\xc0\xbf\x11\xa7\x8a\x03\xa4i\xf3F~\x8e\xc0\x00\x00\u07d4\xe6\xf5\xebd\x9a\xfb\x99Y\x9cAK'\xa9\xc9\xc8U5\u007f\xa8x\x89\x90\xf54`\x8ar\x88\x00\x00\xe0\x94\xe6\xfe\n\xfb\x9d\xce\xdd7\xb2\xe2,E\x1b\xa6\xfe\xabg4\x803\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xe7\x10\xdc\u041b\x81\x01\xf9C{\xd9}\xb9\ns\xef\x99=\v\xf4\x89\x14\xee6\xc0Z\xc2R\x00\x00\u07d4\xe7'\xe6~\xf9\x11\xb8\x1fl\xf9\xc7?\xcb\xfe\xbc+\x02\xb5\xbf\u0189lk\x93[\x8b\xbd@\x00\x00\u07d4\xe7.\x1d3\\\u009a\x96\xb9\xb1\xc0/\x00:\x16\xd9q\xe9\v\x9d\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\xe71\x1c\x953\xf0\t,rH\xc9s\x9b[,\x86J4\xb1\u0389\x97\xf9}l\xc2m\xfe\x00\x00\u07d4\xe7;\xfe\xad\xa6\xf0\xfd\x01o\xbc\x84>\xbc\xf6\xe3p\xa6[\xe7\f\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xe7<\xcfCg%\xc1Q\xe2U\xcc\xf5!\f\xfc\xe5\xa4?\x13\xe3\x89\x01\x15NS!}\xdb\x00\x00\u07d4\xe7B\xb1\xe6\x06\x9a\x8f\xfc<Gg#]\xef\xb0\u051c\xbe\xd2\"\x89+^:\xf1k\x18\x80\x00\x00\u07d4\xe7F\b\xf5\x06\x86j\xdak\xfb\xfd\xf2\x0f\xeaD\v\xe7i\x89\xef\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xe7S>'\f\xc6\x1f\xa1d\xac\x15SE\\\x10]\x04\x88~\x14\x89\x06\x96\xd8Y\x00
\xbb\x00\x00\u07d4\xe7\\\x1f\xb1w\b\x9f>X\xb1\x06y5\xa6Yn\xf1s\u007f\xb5\x89\x05j\x87\x9f\xa7uG\x00\x00\u07d4\xe7\\;8\xa5\x8a?3\xd5V\x90\xa5\xa5\x97f\xbe\x18^\x02\x84\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xe7a\xd2\u007f\xa3P,\xc7k\xb1\xa6\bt\x0e\x14\x03\u03dd\xfci\x89\x0f
-

\xc7\xd4\u007f\x15`\x00\x00\u07d4\xe7f\xf3O\xf1o<\xfc\xc9s!r\x1fC\xdd\xf5\xa3\x8b\f\xf4\x89T\x06
\x923\xbf\u007fx\x00\x00\u07d4\xe7m\x94Z\xa8\x9d\xf1\xe4W\xaa4+1\x02\x8a^\x910\xb2\u03897
\b\xba\xed=h\x90\x00\x00\u07d4\xe7s^\xc7e\x18\xfcj\xa9-
\xa8qZ\x9e\xe3\xf6%x\x8f\x13\x89lM\x16\v\xaf\xa1\xb7\x80\x00\xe0\x94\xe7z\x89\xbdE\xdc\x04\x
ee\xb4\xe4\x1d{Ykp~nQ\xe7L\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xe7}}\uac96\u0234\xfa\a\
xca;\xe1\x84\x16=Zm`l\x89\x05\x049\x04\xb6q\x19\x00\x00\u07d4\xe7\u007f\xeb\xab\xdf\b\x0f\x0f
]\xca\x1d?Wf\xf2\xa7\x9c\x0f\xfa|\x89K\"\x9d(\xa8Ch\x00\x00\xe0\x94\u7025c\x06\xba\x1ek\xb31\
x95,\"S\x9b\x85\x8a\xf9\xf7}\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\xe7\x81\xecs-
@\x12\x02\xbb\x9b\xd18`\x91\r\xd6\u009a\xc0\xb6\x89C8t\xf62\xcc`\x00\x00\u07d4\xe7\x84\xdc\x
c8s\xaa\x8c\x15\x13\xec&\xff6\xbc\x92\xea\xc6\xd4\xc9h\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe7\x91-
L\xf4V,W=\xdc[q\xe3s\x10\xe3x\xef\x86\u0249\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\xe7\x91\u0
545\xb8\x996\xb2])\x8f\x9d5\xf9\xf9\xed\xc2Z)2\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe7\x924\x9
c\xe9\xf6\xf1O\x81\xd0g@\x96\xbe\xfa\x1f\x92!\xcd\xea\x89[]#J\r\xb48\x80\x00\u07d4\xe7\x96\xfd
N\x83\x9bL\x95\xd7Q\x0f\xb7\xc5\xc7+\x83\xc6\xc3\xe3\u01c9\x1b\xc43\xf2?\x83\x14\x00\x00\xe
0\x94\xe7\xa4/Y\xfe\xe0t\xe4\xfb\x13\xea\x9eW\xec\xf1\xccH(\"I\x8a\x04<3\xc1\x93ud\x80\x00\x0
0\u07d4\xe7\xa4V\f\x84\xb2\x0e\x0f\xb5LIg\f)\x03\xb0\xa9lB\xa4\x89
j\xea\u01e9\x03\x98\x00\x00\u07d4\xe7\xa8\xe4q\xea\xfby\x8fET\xccnRg0\xfdV\xe6,}\x8965\u026
d\xc5\u07a0\x00\x00\u07d4\u7f82\xc6Y<\x1e\xed\xdd*\xe0\xb1P\x01\xff
\x1a\xb5{/\x89\x01\t\x10\xd4\xcd\xc9\xf6\x00\x00\u07d4\xe7\u01b5\xfc\x05\xfct\x8e[C\x81rdI\xa1\x
c0\xad\x0f\xb0\xf1\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe7\xd1u$\xd0\v\xad\x82I|\x0f"\x15jd\u007
f\xf5\x1d'\x92\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xe7\xd2\x13\x94\u007f\u02d0J\xd78H\v\
x1e\xed/\\2\x9f'\xe8\x89\x01\x03\u00f1\xd3\xe9\xc3\x00\x00\u07d4\xe7\xd6$\x06
\xf4,^\u06f2\xed\xe6\xae\xc4=\xa4\xed\x9bWW\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe7\xda`\x
9d@\xcd\xe8\x0f\x00\xce[O\xfbj\xa9\u04304\x94\xfc\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe7\x
f0oi\x9b\xe3\x1cD\vC\xb4\xdb\x05\x01\xec\x0e%&\x16D\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u
07d4\xe7\xf4\xd7\xfeoV\x1f\u007f\xa1\xda0\x05\xfd6TQ\xad\x89\u07c9\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe7\xfd\x8f\xd9Y\xae\xd2v~\xa7\xfa\x96\f\xe1\xdbS\xaf\x80%s\x8965\u026d\xc5\u
07a0\x00\x00\u07d4\xe8\x0e\u007f\xef\x18\xa5\xdb\x15\xb0\x14s\xf3\xadkx\xb2\xa2\xf8\xac\u064
9\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xe8\x13\u007f\xc1\xb2\xec|\xc7\x10:\xf9!\x89\x9bJ9\xe
1\xd9Y\xa1\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4\xe8\x1c-
4l\n\xdfL\xc5g\b\xf69K\xa6\xc8\u0226J\x1e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe8,X\xc5yC\x1b
g5F\xb5:\x86E\x9a\xca\xf1\u079b\x93\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xe84\xc6C\x18
\\\xa7\xddJ!\xab\xcb\b&l\xb2\x1f\xf0,\x8965\xc6
G9\u0640\x00\u07d4\xe86\x04\xe4\xffk\xe7\xf9o`\x18\xd3\xec0r\xecR]\xffk\x89\t\xdd\xc1\xe3\xb9\
x01\x18\x00\x00\xe0\x94\xe8E\xe3\x87\xc4\xcb\u07d8\"\x80\xf6\xaa\x01\xc4\x0eK\xe9X\u0772\x8
a\x05K@\xb1\xf8R\xbd\xa0\x00\x00\u07d4\xe8H\xca~\xbf\xf5\xc2O\x9b\x9c1g\x97\xa4;\xf7\xc3V)-
\x89\x06.\x11\\\x00\x8a\x88\x00\x00\u07d4\xe8KU\xb5%\xf1\x03\x9etK\x91\x8c\xb33$\x92\xe4^\x
caz\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe8O\x80v\xa0\xf2\x96\x9e\xcd3>\xef\x8d\xe4\x10B\x98b\x91\xf2\x89\x17k4O*x\x
c0\x00\x00\u07d4\xe8d\xfe\xc0~\xd1!Je1\x1e\x11\xe3)\xde\x04\r\x04\xf0\xfd\x89Y\u0283\xf5\xc4\x
04\x96\x80\x00\u07d4\xe8}\xba\xc66\xa3w!\xdfT\xb0\x8a2\xefIY\xb5\xe4\xff\x82\x89lk\x93[\x8b\xb

d@\x00\x00\xe0\x94\xe8~\x9b\xbf\xbb\xb7\x1c\x1at\ft\xc7#Bm\xf5]\x06=\u064a\x01\xb1\x92\x8c\x00\u01e68\x00\x00\u07d4\xe8~\xacm`+A\t\xc9g\x1b\xf5{\x95\f,\xfd\xb9\x9dU\x89\x02\xb4\xf2\x19r\xec\xce\x00\x00\xe0\x94\u807b\xbeir-
\x81\xef\xec\xaaH\u0455*\x10\xa2\xbf\xac\x8f\x8a\x03c\\\x9a\xdc]\xea\x00\x00\x00\u07d4\xe8\x92Is\x8b~\xce\xd7\xcbfjf<I\xcb\xf6\u0783C\xea\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe8\x9c\"\xf1\xa4\xe1\xd4tn\u03eaY\xed8o\xee\x12\xd5\x1e7\x89\x02o\x8e\x87\xf0\xa7\xda\x00\x00\u07d4\u8769n\x06\xbe\xafk\u0600\xb3x\xf0h\fC\xfd.\x9d0\x89
\x9a\x1a\x01\xa5o\xec\x00\x00\u07d4\xe8\xa9\x1d\xa6\xcf\x1b\x9de\xc7J\x02\xec\x1f\x96\xee\xcbm\xd2A\xf3\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\u8a64\x17@\xf4OT\xc3h\x8bS\xe1\xdd\xd4.C\xc9\xfe\x94\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\u8c8a\u0369qrWi\u06cfV=(fmA\u076bl\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xe8\xbe$\xf2\x89D>\xe4s\xbcv\x82/U\t\x8d\x89\xb9\x1c\u0149lk\x93[\x8b\xbd@\x00\x00\u07d4\xe8\xc3\u04f0\xe1\u007f\x97\xd1\xe7V\xe6\x84\xf9N\x14p\xf9\x9c\x95\xa1\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\xe8\xc3\xf0E\xbb}8\xc9\xd2\U000d5c3a\x84\x92\xb2S#\t\x01\x8a\x01\xe7\xe4\x17\x1b\xf4\u04e0\x00\x00\u07d4\xe8\xccC\xbcO\x8a\xcf9\xbf\xf0N\xbf\xbfB\xaa\xc0j2\x84p\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xe8\xd9B\xd8/\x17^\xcb\x1c\x16\xa4\x05\xb1\x01C\xb3\xf4k\x96:\x89\x1e\xd2\xe8\xffm\x97\x1c\x00\x00\u07d4\xe8\u077e\xd72\xeb\xfeu@\x96\xfd\xe9\bk\x8e\xa4\xa4\xcd\xc6\x16\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe8\xder^\xca]\xef\x80_\xf7\x94\x1d1\xac\x1c.4-
\xfe\x95\x89\x85~\ro\x1d\xa7j\x00\x00\u07d4\xe8\xe9\x85\x05\x86\xe9OR\x99\xabIK\xb8!\xa5\xf4\f\x00\xbd\x04\x89\xcf\x15&@\xc5\xc80\x00\x00\xe0\x94\xe8\xea\u047b\x90\xcc\u00ee\xa2\xb0\xdc\u0175\x80VUFU\xd1\u054a\x01\xa4\xab\xa2%\xc2\a@\x00\x00\u07d4\xe8\xea\xf1)D\t-
\xc3Y\x9b9S\xfa|\xb1\xc9v\x1c\xc2F\x89a\x94\x04\x9f0\xf7
\x00\x00\xe0\x94\xe8\xedQ\xbb\xb3\xac\xe6\x9e\x06\x02K3\xf8hD\xc4sH\u06de\x8a\"\xf9\xea\x89\xf4\xa7\xd6\xc4\x00\x00\u07d4\xe8\xef\x10\r|\xe0\x89X2\xf2g\x8d\xf7-
J\u03cc(\xb8\xe3\x89\x1b\x1bk\u05efd\xc7\x00\x00\u07d4\xe8\xf2\x99i\xe7\\e\xe0\x1c\xe3\xd8aT}\n\x9e|v\xf2\x89\xa2/\xa9\xa7:'\x19\x80\x00\u07d4\xe8\xfc6\xb0\x13\x1e\xc1
\xac\x9e\x85\xaf\xc1\f\xe7\vV\u0636\xba\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe9\n5L\xec\x04\u059e]\x96\xdd\xc0\xc5\x13\x8d=3\x15\n\xa0\x89\x1b\x1a}\u03caD\u04c0\x00\xe0\x94\xe9\x13>}1\x84]_+f\xa2a\x87\x92\xe8i1\x1a\xcff\x8a\x05\x17\xc0\xcb\xf9\xa3\x90\x88\x00\x00\u07d4\xe9\x1d\xac\x01\x95\xb1\x9e7\xb5\x9bS\xf7\xc0\x17\xc0\xb29[\xa4L\x89e\xea=\xb7UF`\x00\x00\u07d4\xe9\x1f\xa0\xba\xda\u0779\xa9~\x88\xd3\xf4\xdb|U\u05bbt0\xfe\x89\x14b\fW\xdd\xda\xe0\x00\x00\u07d4\xe9#\xc0aw\xb3B~\xa4H\xc0\xa6\xff\x01\x9bT\xccT\x8d\x95\x89\x01\xf7\x80\x01Fg\xf2\x80\x00\xe0\x94\xe9=G\xa8\u0288]T\fNRo%\xd5\xc6\xf2\xc1\b\u0138\x8a\x17\xda:\x04\u01f3\xe0\x00\x00\x00\u07d4\xe9E\x8fh\xbb',\xb5g:\x04\xf7\x81\xb4\x03Uo\u04e3\x87\x89\x03N\x8b\x88\xce\xe2\xd4\x00\x00\u07d4\xe9IA\xb6\x03`\x19\xb4\x01j0\xc1\x03}Zi\x03\xba\xba\xad\x89*H\xac\xabb\x04\xb0\x00\x00\u07d4\xe9I[\xa5\x84'(\xc0\ud5fe7\xd0\xe4\"\xb9\x8di
,\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe9M\xed\x99\u0735r\xb9\xbb\x1d\u02e3/m\xee\x91\xe0W\x98N\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\xe0\x94\xe9QyR}\uc951l\xa9\xa3\x8f!\\\x1e\x9c\xe77\xb4\u024a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\xe9U\x91\x85\xf1f\xfc\x95\x13\xccq\x11aD\xce-
\xeb\x0f\x1dK\x8a\x04<3\xc1\x93ud\x80\x00\x00\u0794\xe9^\x92\xbb\xc6\xde\a\xbf:f\x0e\xbf_\xeb

\x1c\x8a5'\xe1\u0148\xfc\x93c\x92\x80\x1c\x00\x00\xe0\x94\xe9e\u06a3@9\xf7\xf0\xdfb7Z7\u5ac
ar\xb3\x01\xe7\x8a\x01\x03\xfd\xde\u0373\xf5p\x00\x00\u07d4\xe9i\xea\x15\x95\xed\xc5\u0127\a\
xcf\xde8\t)c2Q\xa2\xb0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe9k\x18N\x1f\x0fT\x92J\xc8t\xf6\v\xbfDptF\xb7+\x89\x9d\xcc\x05\x15\xb5n\f\x00\
x00\xe0\x94\xe9m}L\xdd\x15U:NM1mmd\x80\xca<\xea\x1e8\x8a\x02\x95]\x02\xe1\xa15\xa0\x00\x
00\u07d4\xe9n-
8\x13\xef\xd1\x16_\x12\xf6\x02\xf9\u007fJb\x90\x9d<f\x89|\xae\xe9v\x13\xe6p\x00\x00\u07d4\xe9
\u007f\xde\vgqc%\xcf\x0e\xcc\u8851\xa3v\x1b,y\x1d\x896w\x03n\xdf\n\xf6\x00\x00\u07d4\xe9\x82
\xe6\xf2\x8cT\x8f_\x96\xf4^c\xf7\xabp\x87$\xf5?\xa1\x89\x15z\xe8)\xa4\x1f;\x00\x00\u07d4\xe9\x8
6L\x1a\xfc\x8e\xaa\xd3\u007f;\xa5o\xcbtw\xccb
\t\xb7\x89\x04HXap\xa7\xdc\x00\x00\u07d4\xe9\x87\xe6\x13\x9eaF\xa7\x17\xfe\xf9k\xc2I4\xa5D\u
007f\xe0]\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe9\x89s<\xa1\u054d\x9e{P)\xba]DHX\xbe\xc01r\x
89\x1f\x87@\x83\x13\xdfO\x80\x00\u07d4\u9311\xca\u0752L\x92W\x9e\x11\xb4\x12\x17\xb2\x82\
x95l\u06a1\x89\a\\\x9a\x84\x802\f\x00\x00\u07d4\xe9\x9a\xec\xe9\x05A\xca\xe2$\xb8}\xa6s\x96^\
n\xeb)j\xfd\x891\u07d0\x95\xa1\x8f`\x00\x00\u07d4\xe9\x9d\xe2X\xa4\x17<\xe9\xac8\xed\xe2l\v;\x
ea<\ts\u0549Y\u0438\x05\xe5\xbb0\x00\x00\u07d4\u98b4\x91N\x85S\xbf\r|\x00\xcaS#i\xb8y\xf91\
xbf\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\u98da\x8b\xac\x0f\x01\xc3I\xc6L\xed\xb6\x98\x97\xf63#N
\u0489\xd7\xc1\x98q\x0ef\xb0\x00\x00\u07d4\u996e<\x9e\x05\x97}\xd1\x06\x9e\x9f\xd9\u04ee\xf
b\xae\x04\xb8\u07c9j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xe9\xac67n\xfa\x06\x10\x9d@rc\a\xdd\x1a
W\xe2\x13\uaa49\n\x84Jt$\xd9\xc8\x00\x00\u07d4\xe9\xb1\xf1\xfc\xa3\xfaG&\x9f!\xb0a\xc3S\xb7\
xf5\xe9m\x90Z\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94\xe9\xb3o\xe9\xb5\x14\x12\xdd\xca
\x1aR\x1dn\x94\xbc\x90\x12\x13\u0768\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\u9d24\
x855w\xa9\xdb\xcc.y[\xe01\r\x1b\xed(d\x1a\x8965\u026d\xc5\u07a0\x00\x00\u07d4\u9da7\x90\x0
0\x9b\xc1fB\xc8\xd8
\xb7\xcd\xe0\xe9\xfd\x16\xd8\xf5\x89\xc5S%\xcat\x15\xe0\x00\x00\u07d4\u9e62tu\x10\xe3\x10$\x
1d.\u0398\xf5k3\x01\xd7W\xe0\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xe9\xc3\\\x91<\xa1\xfc\xea\
xb4aX/\u1941Qd\xb4\xfd!\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xe9\xc6\u07ee\x97\xf7\t\x9f\
xc5\xf4\xe9KxM\xb8\x02\x92:\x14\x19\x89\x02\xa5<mrO\x10\x00\x00\u07d4\xe9\xc7X\xf8\xdaA\x
e34nCP\xe5\xac9v4\\l\x10\x82\x89h\xa0\xd3\t(&\xad\x00\x00\u07d4\xe9\xca\xf8'\xbe\x9d`y\x15\xb
3e\xc8?\r;~\xa8\u01dbP\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xe9\xca\xfeA\xa5\xe8\xbb\xd9\v\
xa0-
\x9e\x06X[N\xb5F\xc5\u007f\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xe9\u0559Ek%C\xe6\u06c0\xea
\x9b!\x0e\x90\x80&\xe2\x14n\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xe9\xe1\xf7\xcb\x00\xa1\x10\xed\xd0\xeb\xf8\xb3w\xef\x8a{\xb8V\x11\u007f\x89\n
\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xea\x14\xbf\xda\nnvf\x8f\x87\x882\x1f\a\xdf7\x82N\xc5\u07ca*Z\x05\x8f\u0095\x
ed\x00\x00\x00\u07d4\xea\x1e\xa0\u0159\xaf\xb9\xcd6\u02ac\xbb\xb5+[\xbb\x97Ysw\x899\xfb\xa
e\x8d\x04-
\xd0\x00\x00\u07d4\xea\x1e\xfb<\u727e\xde\xc3\xd6|>\x1b;\xc0\xe9\xaa\"\u007f\x90\x89'\xcaK\xd
7\x19\xf0\xb8\x00\x00\u07d4\xea,\x19}&\xe9\x8b\r\xa8>\x1br\u01c7a\x8c\x97\x9d=\xb0\x89\x01\x
11du\x9f\xfb2\x00\x00\xe0\x94\xea7y\xd1J\x13\xf6\u01c5f\xbc\xde@5\x91A:b9\u06ca)\xb7d2\xb9
DQ

\x00\x00\u07d4\xeaN\x80\x9e&j\xe5\xf1<\xdb\u33dd\x04V\xe68m\x12t\x89\xf3\xf2\v\x8d\xfai\xd0\x00\x00\xe0\x94\xeaS\xc9T\xf4\xed\x97\xfdH\x10\x11\x1b\u06b6\x9e\xf9\x81\xef%\xb9\x8a\x03\xa9\u057a\xa4\xab\xf1\xd0\x00\x00\u07d4\xeaS\xd2ed\x85\x9d\x9e\x90\xbb\x0eS\xb7\xab\xf5`\xe0\x16,8\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xea`Ci\x12\xdek\xf1\x87\u04e4r\xff\x8fS3\xa0\xf7\xed\x06\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4\xea`T\x9e\xc7U?Q\x1d!I\xf2\xd4fl\xbd\x92C\xd9<\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xeaf\xe7\xb8M\u037f6\xee\xa3\xe7[\x858*u\xf1\xa1]\x96\x89]\xbc\x91\x91&o\x11\x80\x00\u07d4\xeahlPW\t<\x17\x1cf\u06d9\xe0\x1b\x0e\xce\xcb0\x86\x83\x89\x14\u0768],\xe1G\x80\x00\u07d4\xeaj\xfe,\xc9(\xac\x83\x91\xeb\x1e\x16_\xc4\x00@\xe3t!\u7262\u007f\xa0c\xb2\xe2\xe6\x80\x00\u07d4\xeay\x05}\xab\xef^d\xe7\xb4O\u007f\x18d\x8e~S7\x18\u0489\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xea|Mm\xc7)\xcdk\x15|\x03\xad#|\xa1\x9a
\x93F\u00c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xea\x81h\xfb\xf2%\xe7\x86E\x9c\xa6\xbb\x18\xd9c\xd2kPS\t\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xea\x81\u02868T\f\xd9\xd4\xd7=\x06\x0f,\xeb\xf2$\x1f\xfc>\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xea\x83\x17\x19yYB@A\xd9\xd7\xc6z>\xce\x1d\xbbx\xbbU\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\xea\x85'\xfe\xbf\xa1\xad\xe2\x9e&A\x93)\u04d3\xb9@\xbb\xb7\u0709lj\xccg\u05f1\xd4\x00\x00\u07d4\xea\x8f0\xb6\xe4\xc5\xe6R\x90\xfb\x98d%\x9b\u0159\x0f\xa8\ue289\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xea\x94\xf3(\b\xa2\uf29b\xf0\x86\x1d\x1d$\x04\xf7\xb7\xbe%\x8a\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xea\xa4\\\xea\x02\xd8},\xc8\xfd\xa9CN-
\x98[\xd4\x03\x15\x84\x89h\x1f\xc2\xccn+\x8b\x00\x00\xe0\x94\uac3d\x14\x83\t\x18l\xf8\xcb\xd1;r2\xd8\tZ\u02c3:\x8a\x02C\x9a\x88\x1cjq|\x00\x00\u07d4\uaed0\xd3y\x89\xaa\xb3\x1f\xea\xe5G\xe0\xe6\xf3\x99\x9c\xe6\xa3]\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xea\xc0\x82~\xff\fn?\xf2\x8a}JT\xf6\\\xb7h\x9d{\x99\x89\x9a\xd9\u67ddGR\x00\x00\u07d4\xea\xc1H(&\xac\xb6\x11\x1e\x19\xd3@\xa4_\xb8QWk\xed`\x89\x01\xbe\x8b\xab\x04\u067e\x80\x00\xe0\x94\xea\xc1{\x81\xedQ\x91\xfb\b\x02\xaaT3s\x13\x83A\a\xaa\xa4\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xea\u00efW\x84\x92\u007f\u9958\xfcN\xec8\xb8\x10/7\xbcX\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xea\u01b9\x88BT.\xa1\v\xb7O&\xd7\xc7H\x8fi\x8bdR\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xea\xc7h\xbf\x14\xb8\xf9C.i\xea\xa8*\x99\xfb\xeb\x94\xcd\f\x9c\x8a\x14\u06f2\x19\\\xa2(\x90\x00\x00\u07d4\xea\xd2\x1c\x1d\xec\u03ff\x1c\\\xd9f\x88\xa2Gki\xba\a\xceJ\x89\x03\xf2M\x8eJ\x00p\x00\x00\u07d4\xea\xd4\xd2\xee\xfbv\xab\xaeU3\x96\x1e\xdd\x11@\x04\x06\xb2\x98\xfc\x89\xd2U\xd1\x12\xe1\x03\xa0\x00\x00\u07d4\xea\xd6Rb\xed]\x12-
\xf2\xb2u\x14\x10\xf9\x8c2\xd1#\x8fQ\x89\x05\x83\x17\xedF\xb9\xb8\x00\x00\u07d4\xea\xd7P\x16\u3801Pr\xb6\xb1\b\xbc\xc1\xb7\x99\xac\xf08>\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xea\xea#\xaa\x05r\x00\xe7\xc9\xc1^\x8f\xf1\x90\xd0\xe6l\f\x0e\x83\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xea\xed\x16\xea\xf5\u06ab[\xf0)^^\a\u007fY\xfb\x82U\x90\v\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xea\xed\xcck\x8bib\xd5\xd9(\x8c\x15lW\x9dG\xc0\xa9\xfc\xff\x89\x04\x9b\x9c\xa9\xa6\x944\x00\x00\u07d4\xea\xf5#\x88Tn\xc3Z\xcaolc\x93\xd8\xd6\t\xde:K\xf3\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xeb\x10E\x8d\xac\xa7\x9eJk$\xb2\x9a\x8a\x8a\xdaq\x1b\u007f.\xb6\x89\u063beI\xb0+\xb8\x00\x00\u07d4\xeb\x1c\xea{E\u047dM\x0e*\x00{\u04ff\xb3Tu\x9e,\x16\x89\n\xbb\xcdN\xf3wX\x00\x00\u07d4\xeb%H\x1f\u035c\"\x1f\x1a\xc7\xe5\xfd\x1e\u0353\a\xa1b\x15\xb8\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\xe0\x94\xeb.\xf3\u04cf\xe6R@<\xd4\xc9\xd8^\xd7\xf0h,\xd7\xc2\u078a\t\x0fSF\b\xa7(\x80\x00\x00\xe0\x94\xeb;\xddY\xdc\u0765\xa9\xbb*\xc1d\x1f\xd0!\x80\xf5\xf3e`\x8a\x01e\xc9fG\xb3\x8a

\x00\x00\u07d4\xeb<\xe7\xfc8\x1cQ\xdb}_\xbdi/\x8f\x9e\x05\x8aLp=\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xebE?Z:\xdd\u074a\xb5gP\xfa\xdb\x0f\xe7\xf9M\x9c\x89\xe7\x89\x01\x15\x8eF\t\x
13\xd0\x00\x00\u07d4\xebO\x00\xe2\x836\xea\t\x94%\x88\ueb12\x18\x11\xc5\"\x14<\x89lk\x93[\x
8b\xbd@\x00\x00\u07d4\xebR\xab\x10U4\x922\x9c\x1cT\x83:\xe6\x10\xf3\x98\xa6[\x9d\x89\b=lz\x
abc`\x00\x00\u07d4\xebW\r\xba\x97R'\xb1\xc4-
n\x8d\xea,V\u026d\x96\x06p\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xebc\x94\xa7\xbf\xa4\u0489\x1
1\u0565\xb2>\x93\xf3^4\f\"\x94\x89\x04:w\xaa\xbd\x00x\x00\x00\u07d4\xebh\x10i\x1d\x1a\xe0\u0
45eG\xbd\"\u03be\u0cfa'\xf8\x8a\x89\x87\x85c\x15\xd8x\x15\x00\x00\u07d4\xebvBL\x0f\u0557\xd
3\xe3A\xa9d*\xd1\xee\x11\x8b+W\x9d\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xeb|
+F+|\u0145]t\x84u_n&\xefC\xa1\x15\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xeb\x83\\\x1a\x91\x18\x
17\x87\x8a3\xd1gV\x9e\xa3\xcd\u04c7\xf3(\x8965\u026d\xc5\u07a0\x00\x00\u07d4\ub268\x82g\t\t
\xcf7~\x9ex(n\xe9{\xa7\x8dF\u0089+|\xc2\xe9\xc3\"\\\x00\x00\xe0\x94\xeb\x90\u01d3\xb3S\x97a\x
e1\xc8\x14\xa2\x96q\x14\x86\x92\x19>\xb4\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xeb\x9c\xc9
\xfe\bi\xd2\u06b5,\u01ea\xe8\xfdW\xad\xb3_\x9f\xeb\x89j\x93\xbb\x17\xaf\x81\xf8\x00\x00\xe0\x9
4\ub8c8\xb0\xda'\xc8{\x1c\xc0\xea\xc6\xc5{,Z\vE\x9c\x1a\x8a\x01p\xa0\xf5\x04\x0eP@\x00\x00\u
07d4\xeb\xaa!m\xe9\xccZC\x03\x17\a\xd3o\xe6\u057e\xdc\x05\xbd\xf0\x89j\xc5\xc6-
\x94\x86\a\x00\x00\u07d4\xeb\xac+D\b\xefT1\xa1;\x85\b\xe8bP\x98!\x14\xe1E\x89\xd8\xd7&\xb7\
x17z\x80\x00\x00\u07d4\xeb\xb6,\xf8\xe2,\x88K\x1b(\xc6\xfa\x88\xfb\xbc\x17\x93\x8a\xa7\x87\x89
+By\x84\x03\u0278\x00\x00\u07d4\xeb\xb7\xd2\xe1\x1b\u01b5\x8f\n\x8dE\xc2\xf6\xde0\x10W\n\u
0211\x89\x01s\x17\x90SM\xf2\x00\x00\u07d4\xeb\xbbO,=\xa8\xbe>\xb6-
\x1f\xfb\x1f\x95\x02a\u03d8\xec\u0689lk\x93[\x8b\xbd@\x00\x00\u07d4\xeb\xbdM\xb9\x01\x99R\u
058b\x1b\x0fm\x8c\xf0h<\x008{\xb5\x89\x12\x04\x01V=}\x91\x00\x00\u07d4\xeb\xbe\xeb%\x91\x8
4\xa6\xe0\x1c\xcc\xfc\"\a\xbb\u0603xZ\xc9\n\x89!\x9b\xc1\xb0G\x83\xd3\x00\x00\u07d4\xeb\xd3V
\x15j81#4=H\x84;\xff\xeda\x03\xe8f\xb3\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xeb\xd3{%ec\xe3\f
o\x92\x89\xa8\xe2p/\bR\x88\b3\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xeb\xe4l\xc3\xc3L2\xf5\xad\xd
6\xc3\x19[\xb4\x86\xc4q>\xb9\x18\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xeb\xff\x84\xbb\xefB0
q\xe6\x04\xc3a\xbb\xa6w\xf5Y=\xefN\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xec\t'\xba\
xc7\xdc6f\x9c(5J\xb1\xbe\x83\xd7\xee\xc3\t4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xec\x0e\x18\xa
0\x1d\xc4\xdc]\xaa\xe5g\xc3\xfaL\u007f\x8f\x9bY\x02\x05\x89\x11\x1f\xfe@JA\xe6\x00\x00\xe0\x9
4\xec\x116,\xec\x81\t\x85\xd0\xeb\xbd{sE\x14D\x98[6\x9f\x8a\x06ZNIWpW1\x80\x00\u07d4\xec,\x
b8\xb97\x8d\xff1\xae\xc3\xc2.\x0em\xad\xff1J\xb5\u0749lk\x93[\x8b\xbd@\x00\x00\u07d4\xec0\xa
d\u0749[\x82\xee1\x9eT\xfb\x04\xcb+\xb09q\xf3k\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xec;\x8bX\
xa1'\x03\xe5\x81\xce_\xfd~!\xc5}\x1e\\f?\x89\\(=A\x03\x94\x10\x00\x00\u07d4\xecHg\xd2\x17Z\xb
5\xb9F\x93aYUFUF\x84\u0364`\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xecM\b\xaa.GIm\u0287\"]
\xe3?+@\xa8\xa5\xb3o\x89\b\x90\xb0\xc2\xe1O\xb8\x00\x00\u07d4\xecX\xbc\r\f
\xd8\xf4\x94efAS\xc5\xc1\x96\xfeY\u6f89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xec[\x19\x8a\x00
\u03f5Z\x97\xb5\xd56D\xcf\xfa\x8a\x04\u04abE\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xec]\xf2'\xbf\
xa8]z\xd7kBn\x1c\xee\x96;\xc7\xf5\x19\u074965\u026d\xc5\u07a0\x00\x00\xe0\x94\xec_\xea\xfe!\
f\x12\xbf\u0265\xd0Y%\xa1#\xf1\xe7?\xbe\xf8\x8a`\x8f\xcf=\x88t\x8d\x00\x00\x00\u07d4\xeci\x04\
xba\xe1\xf6\x97\x90Y\x17\t\xb0`\x97\x83s?%s\xe3\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\xe0\x94
\xecs\x11L^@o\u06fe\t\xb4\xfab\x1b\xd7\x0e\xd5N\xa1\xef\x8a\x050%\xcd!o\xceP\x00\x00\u07d4\
xecs\x83=\xe4\xb8\x10\xbb\x02x\x10\xfc\x8fi\xf5D\xe8<\x12\u044965\u026d\xc5\u07a0\x00\x00\u

07d4\xecu\xb4\xa4u\x13\x12\v\xa5\xf8`9\x81O\x19\x98\xe3\x81z\u00c9\t\xb0\xbc\xe2\xe8\xfd\xba\x00\x00\u07d4\xecv\xf1.W\xa6U\x04\x03?,\v\xceo\xc0;\xd7\xfa\n\u0109\xc2\x12z\xf8X\xdap\x00\x00\u0794\xec\x80\x14\xef\xc7\xcb\xe5\xb0\xceP\xf3V,\xf4\xe6\u007f\x85\x93\xcd2\x88\xf0\x15\xf2W6B\x00\x00\u07d4\xec\x82\xf5\r\x06G_hM\xf1\xb3\x92\xe0\r\xa3A\xaa\x14TD\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xec\x83\xe7\x98\u00d6\xb7\xa5^*\"$\xab\u0343K'\xeaE\x9c\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xec\x89\xf2\xb6x\xa1\xa1[\x914\xec^\xb7\fjb\a\x1f\xba\xf9\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xec\x8c\x1d{j\xac\xcdB\x9d\xb3\xa9\x1e\xe4\xc9\xeb\x1c\xa4\xf6\xf7<\x89\xe6d\x99\"\x88\xf2(\x00\x00\xe0\x94\xec\x98Q\xbd\x91rpa\x02g\xd6\x05\x18\xb5M<\xa2\xb3[\x17\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xec\x99\xe9]\xec\xe4o\xff\xfb\x17^\xb6@\x0f\xbe\xbb\b\ue6d5\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xec\xa5\xf5\x87\x92\xb8\xc6-*\xf5Vq~\xe3\xee0(\xbeM\u0389lk\x93[\x8b\xbd@\x00\x00\u07d4\xec\xabZ\xba[\x82\x8d\xe1pS\x81\xf3\x8b\xc7D\xb3+\xa1\xb47\x892\xf5\x1e\u06ea\xa30\x00\x00\u07d4\xec\xaf3P\xb7\xce\x14M\x06\x8b\x18`\x10\x85,\x84\xdd\f\xe0\xf0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xec\xb9LV\x8b\xfeY\xad\xe6Pd_O&0lsl\xac\xe4\x89\x0e~\xeb\xa3A\vt\x00\x00\xe0\x94\xec\xbeB^g\r9\tN
\xfbVC\xa9\xd8\x18\xee\xd26\u078a\x01\x0f\f\xf0d\xddY
\x00\x00\xe0\x94\xec\xbe^\x1c\x9a\u04b1\xdc\xcf\n0_\xc9R/Fi\xdd:\xe7\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xec\xcfz\x04W\xb5f\xb3F\xcag:\x18\x0fDA0!j\u00c9\x05k\xc7^-c\x10\x00\x00\u07d4\xec\u0466(\x025\x1aAV\x8d#\x030\x04\xac\xc6\xc0\x05\xa5\u04c9\x02\xb5\xe3\xaf\x16\xb1\x88\x00\x00\u07d4\xec\xd2v\xafd\u01dd\x1b\u0669+\x86\xb5\u835a\x95\xeb\x88\xf8\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xec\u0506\xfc\x19g\x91\xb9,\xf6\x12\xd3HaO\x91VH\x8b~\x8a\x02\x8a\x85t%Fo\x80\x00\x00\u07d4\xec\xda\xf92)\xb4^\xe6r\xf6]\xb5\x06\xfb^\xca\x00\xf7\xfc\xe6\x89W\x01\xf9m\xcc@\xee\x80\x00\u07d4\xec\xe1\x11g\vV<\u037e\xbc\xa5#\x84)\x0e\xcdh\xfe\\\x92\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xec\xe1\x15&\x82\xb7Y\x8f\xe2\xd1\xe2\x1e\xc1U3\x88T5\xac\x85\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xec\xe1)\bw\xb5\x83\xe3a\xa2\xd4\x1b\x00\x93F\xe6'N%8\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xec\xf0]\a\xea\x02n~\xbfIA\x00#5\xba\xf2\xfe\xd0\xf0\x02\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xec\xf2L\xdd|\"\x92\x8cD\x1eiM\xe4\xaa1\xb0\xfa\xb5\x97x\x89
\x86\xac5\x10R`\x00\x00\xe0\x94\xec\xfd\x00M\x02\xf3l\xd4\u0634\xa8\xc1\xa9S;j\xf8\\\xd7\x16\x8a\x01\x0fA\xac\xb4\xbb;\x9c\x00\x00\xe0\x94\xed\x02\x06\xcb#1Q(\xf8\xca\xff&\xf6\xa3\v\x98Tg\xd0\"\x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xed\x10e\xdb\u03dds\xc0O\xfcy\b\x87\r\x88\x14h\xc1\xe12\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xed\x12vQ;o\u0186(\xa7A\x85\xc2\xe2\f\xbb\xcax\x17\xbf\x89\nZ\xa8P\t\xe3\x9c\x00\x00\xe0\x94\xed\x12\xa1\xba\x1f\xb8\xad\xfc\xb2\r\xfa\x19X.RZ\xa3\xb7E$\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\xed\x16\xce9\xfe\xef;\xd7\xf5\xd1b\x04^\x0fg\xc0\xf0\x00F\xbb\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xed\x1a\\C\xc5t\xd4\xe94)\x9b$\xf1G,\u071f\xd6\xf0\x10\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xed\x1b$\xb6\x91-Q\xb34\xac\r\xe6\xe7q\xc7\xc0EF\x95\xea\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xed\x1f\x1e\x11Z\r`\xce\x02\xfb%\xdf\x01M(\x9e:\f\xbe}\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xed10\\1\x9f\x92s\u04d3m\x8f[/q\u9c72)c\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xed2z\x14\xd5\u03ed\u0641\x03\xfc\t\x99q\x8d~\xd7\x05(\xea\x89N\x10\x03\xb2\x8d\x92\x80\x00\x00\u07d4\xed<\xbc7\x82\u03bdg\x98\x9b0\\A3\xb2\xcd\xe3\"\x11\xeb\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xed@\x14S\x8c\xeefJ/\xbc\xb6\xdcf\x9fz\xb1m\v\xa5|\x89\n\

u05ce\xbcZ\xc6
\x00\x00\u07d4\xedA\u188f\\\xaa\x848\x80\xefN\x8b\b\xbdl3\x14\x1e\u07c9*\xd5\xdd\xfaz\x8d\x83
\x00\x00\xe0\x94\xedK\xe0J\x05-z\u0333\xdc\u03901\x9d\xba@
\xab,h\x8a\a\xf3zp\xea\xf3b\x17\x80\x00\xe0\x94\xedR\xa2\xcc\bi\u071e\x9f\x84+\u0415|G\xa8\xe
9\xb0\xc9\xff\x8a\x02\x05\xb4\u07e1\xeetx\x00\x00\u07d4\xed[LA\xe7b\xd9B@Cs\xca\xf2\x1e\xd
4a]%\xe6\xc1\x89m-
O=\x95%\xb4\x00\x00\u07d4\xed`\u012bnT\x02\x061~5\x94zc\xa9\xcak\x03\xe2\u02c9\x03\x1a\u
066d\vF\u007f\x80\x00\u07d4\xedd\x1e\x066\x8f\xb0\xef\xaa\x17\x03\xe0\x1f\xe4\x8fJhS\t\xeb\x8
9\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xedfC\xc0\xe8\x88K-
2\x11\x857\x85\xa0\x8b\xf8\xf3>\u049f\x89Hz\x9a0E9D\x00\x00\xe0\x94\xedp\xa3|\xdd\x1c\xbd\x
a9tm\x93\x96X\xae*a\x81(\x85x\x8a\x02\bj\xc3Q\x05&\x00\x00\x00\u07d4\xedsFvn\x1agm\r\x06\x
ec\x82\x18g\xa2v\xa0\x83\xbf1\x89\u064a\t1\xcc-
I\x00\x00\u07d4\xed\x86&\x16\xfc\xbf\xb3\xbe\xcbt\x06\xf7<\\\xbf\xf0\f\x94\aU\x89\\(=A\x03\x94\x1
0\x00\x00\u07d4\xed\x9e\x03\f\xa7\\\xb1\u049e\xa0\x1d\rL\xdf\xdc\xcd8D\xb6\xe4\x89\x01\xac\xc
1\x16\u03ef\xb1\x80\x00\xe0\x94\ud7bc\u02e4/\x98\x15\xe7\x823&m\xd6\xe85\xb6\xaf\xc3\x1b\x8
a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\ud7f1\xf5\xaf/\xbf\u007f\xfcP)\xce\xe4+p\xff\\'[\xf5\x89\x
0f-
\xc7\xd4\u007f\x15`\x00\x00\u07d4\xed\xa4\xb2\xfaY\u0584\xb2z\x81\r\xf8\x97\x8as\xdf0\x8ac\u0
089\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xed\xb4s59y\xa2\x06\x87\x9d\xe1D\xc1\n<Q\xd7\xd
7\b\x1a\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xed\xb7\x1e\xc4\x1b\xda}\u0386\xe7f\xe6\xe
8\xc3\xe9\x90w#\xa6\x9b\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xed\xba\xc9R{T\xd6\xdfz\xe
2\xe0\x00\u0323a;\xa0\x15\xca\xe3\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xed\xc2/\xb9,c\x8e\x1e!
\xff\\\xf09\u06a6\xe74\xda\xfb)\x89\x10'\x94\xad
\xdah\x00\x00\xe0\x94\xed\xda\u0354\uc262\uf58f\u03d7z\b\xf1\xfa\xe2uxi\x8a\x01\xb1\xaeMn.\xf
5\x00\x00\x00\u07d4\xed\u06ea\xfb\xc2\x1b\xe8\xf2Ub\xf1\xedm\x05\u05af\xb5\x8f\x02\u0089lk\x
93[\x8b\xbd@\x00\x00\u07d4\xed\xe0\x14~\xc02\xc3a\x83\x10\xc1\xff%i\v\xf1r\x19=\xac\x89lk\x9
3[\x8b\xbd@\x00\x00\u07d4\xed\xe5\xde|\u007f\xb7\xee\xe0\xf3ndS\nAD\x0e\u07fe\xfa\u03c9!u^\
xe1\xef+\x18\x00\x00\u07d4\xed\xe7\x9a\xe1\xffO\x16\x06\u0552p!o\xa4j\xb2\xdd\xd4\uca89\a\xe
a(2uw\b\x00\x00\xe0\x94\xed\xe8\xc2\u02c7o\xbe\x8aL\u0282\x906\x1a~\xa0\x1ai\xfd\xf8\x8a\x0
1\xa7\x8ckD\xf8A\x83\x80\x00\u07d4\xed\xebH\x94\xaa\xdd\x00\x81\xbb\xdd\xd3\xe8\x84h\x04\x
b5\x83\u045f'\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xed\xf6\x03\x89\x02(\xd7\xd5\u0793\t\x94+\\\
xadB\x19\xef\x9a\u05ca\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xed\xf8\xa3\xe1\xd4\x0f\x13\xb7\x9e\xc8\xe3\xe1\xec\xf2b\xfd\x92\x11bc\x89\b\x9
0\xb0\xc2\xe1O\xb8\x00\x00\u07d4\xed\xfd\xa2\xd5\u06d8\xf98\a\x14fMT\xb4\xee\x97\x1a\x1c\xa
e\x03\x89\x02+\xb8\xdd\xd6y\xbe\x00\x00\u07d4\xee\x00\a\xb0\x96\r\x00\x90\x8a\x94C*suW\x87j
\xac|1\x89\x02\xe0B\x1ei\xc4\u0300\x00\u07d4\xee\x04\x9a\xf0\x05\x97M\xd1\u01f3\xa9\u028d\x
9a\xa7qu\xbaS\xaa\x89\x12\x11\xec\xb5m\x13H\x80\x00\u07d4\xee%\xb9\xa7\x03&y\xb1\x13X\x
8e\xd5,\x13}\x1a\x05:\x1e\x94\x89\n\xd5\x0f?N\xea\x8e\x00\x00\u07d4\xee1\x16\u007f\x9c\xc9;<
de`\x9dy\xdb\f\u0790\xe8HL\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xee4\xc7\xe7\x99]\xb9\xf1\x87\x
cf\xf1V\x91\x8c\xfbo\x13\xf6\xe0\x03\x89j@v\xcfy\x95\xa0\x00\x00\u07d4\xee5d\xf5\xf1\xba\x0f\x
94\xec{\xac\x16K\u077f1\u0188\x8bU\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xeeX\xfb=\xb2\x90p\xd0\x13\x01\x88\xceG+\xe0\xa1r\xb8\x90U\x8a\x02\x1

fB\xdc\xdcX\xe3\x9c\x00\x00\u07d4\xeee[\xb4\xee\x0e\x8dTxRo\xb9\xf1^@d\xe0\x9f\xf3\u0749\n\
u05ce\xbcZ\xc6
\x00\x00\u07d4\xeeiY\xde+g\x96{q\x94\x8c\x89\x1a\xb0\r\x8c\x8f8\xc7\u0709\x06hZ\xc1\xbf\xe3,\
x00\x00\u07d4\xeel\x03B\x99i\xca\x12b\xcb?\nJT\xaf\xa7\xd3H\xd7\xf5\x89\r\xe2\x19\xf9\x1f\xc1\
x8a\x00\x00\u07d4\xeeqy>:\xcf\x12\xa7'OV9a\xf57R\x9d\x89\xc7\u0789lk\x93[\x8b\xbd@\x00\x00
\u07d4\xeer\x88\xd9\x10\x86\xd9\xe2\xeb\x91\x00\x14\u066b\x90\xa0-
x\u00a0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xee|=\xed|(\xf4Y\xc9/\xe1;M\x95\xba\xfb\xab\x026}\x
89%\xf2s\x93=\xb5p\x00\x00\xe0\x94\xee\x86}
\x91k\xd2\xe9\xc9\xec\xe0\x8a\xa0C\x85\xdbf|\x91.\x8a\n\x96\x81c\xf0\xa5{@\x00\x00\u07d4\ue2
5b\x02\xcb\xcb99\xcda\xde\x13B\xd5\x04\x82\xab\xb6\x852\x89_h\xe8\x13\x1e\u03c0\x00\x00\u0
7d4\xee\x90m}_\x17H%\x81t\xbeL\xbc8\x93\x03\x02\xab{B\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\ue5ea\x8a\u019e\xdfz\x98}mp\x97\x9f\x8e\xc1\xfb\xcaz\x94\x89\x14b\fW\xdd\xda\
xe0\x00\x00\u07d4\xee\xa1\xe9y\x88\xdeu\xd8!\xcd(\xadh\"\xb2,\u0398\x8b1\x89\x1c0s\x1c\xec\x
03
\x00\x00\xe0\x94\xee\u048c?\x06\x8e\tJ0K\x85<\x95\nh\t\xeb\xcb\x03\xe0\x8a\x03\xa9\u057a\xa4
\xab\xf1\xd0\x00\x00\u07d4\xee\u04c4\xef-
A\xd9\xd2\x03\x97NW\xc1#(\xeav\x0e\b\xea\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xee\xdflB\x8
0\xe6\xeb\x05\xb94\xac\xe4(\xe1\x1dB1\xb5\x90[\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xee\xe7a\x84~3\xfda\u0653\x87\xee\x14b\x86\x94\u047f\xd5%\x89lk\x93[\x8b\xbd
@\x00\x00\xe0\x94\xee\xe9\xd0Rn\xda\x01\xe41\x16\xa3\x952-
\u0689pW\x8f9\x8a\x02\x1e\x19\x99\xbb\xd5\u04be\x00\x00\u07d4\xee\xf1\xbb\xb1\xe5\xa8?\u07
82H\xf8\x8e\xe3\x01\x8a\xfa-\x132\xeb\x89\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xee\xfb\xa1-
\xfc\x99gB\xdby\x04d\xca}';\xe6\xe8\x1b>\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xee\xfd\x05\xb0
\xe3\xc4\x17\xd5[3C\x06\x04\x86\xcd\xd5\xe9*\xa7\xa6\x89M\x85<\x8f\x89\b\x98\x00\x00\u07d4\x
ef\r\xc7\xddzS\xd6\x12r\x8b\xcb\u04b2|\x19\xddM}fo\x89&A\x1c[5\xf0Z\x00\x00\u07d4\xef\x11RR\
xb1\xb8E\u0345\u007f\x00-
c\x0f\x1bo\xa3zNP\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xef\x1c\x04w\xf1\x18M`\xac\u02b3t\xd3t
Uz\n>\x10\xf3\x89\b=lz\xabc`\x00\x00\u07d4\xef,4\xbbH}7b\xc3\u0327\x82\xcc\xddz\x8f\xbb\n\x99
1\x89\t\xc2\x00vQ\xb2P\x00\x00\u07d4\xef5\xf6\u0531\a^j\xa19\x15\x1c\x97K/FX\xf7\x058\x89<;\
xc3?\x94\xe5\r\x80\x00\u07d4\xef9\u0291s\xdf\x15S\x1ds\xe6\xb7*hKQ\xba\x0f+\xb4\x89V\xa0\xb
4un\xe28\x00\x00\u07d4\xefF<&y\xfb'\x91d\xe2\f=&\x915\x87s\xa0\xad\x95\x89lk\x93[\x8b\xbd@\
x00\x00\u07d4\xefG\xcf\a>6\xf2q\xd5\"\xd7\xfaNq
\xadP\a\xa0\xbc\x89\x87\x86x2n\xac\x90\x00\x00\u07d4\xefa\x15[\xa0\t\xdc\u07be\xf1\v(\xd9\xda
=\x1b\xc6\xc9\xce\u0509\x034-
`\xdf\xf1\x96\x00\x00\u0794\xefix\x1f2\xff\xce34o,\x9a\xe3\xf0\x84\x93\xf3\xe8/\x89\x88\xfc\x93c\x
92\x80\x1c\x00\x00\u07d4\xefv\xa4\u034f\xeb\xcb\u0278\x18\xf1x(\xf8\xd94s\xf3\xf3\u02c9\xd8\xd
7&\xb7\x17z\x80\x00\x00\u07d4\uf4c1\x8fhM\xb0\xc3g^\xc8\x132\xb3\x18>\xcc(\xa4\x95\x89T\x0
6\x923\xbf\u007fx\x00\x00\xe0\x94\xef\x9fY\xae\xdaA\x8c\x14\x94h-
\x94\x1a\xabI$\xb5\xf4\x92\x9a\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\uf9b1\xf0\xdb`57\x82h\x91\xb8\xb4\xbc\x169\x84\xbb@\u03
495e\x9e\xf9?\x0f\xc4\x00\x00\u07d4\xef\xbdR\xf9}\xa5\xfd:g:F\xcb\xf30D{~\x8a\xad\\\x89\x05l<\x
9b\x80\xa0\xa6\x80\x00\xe0\x94\xef\xc8\xcf\x19c\u0269Rg\xb2(\xc0\x86#\x98\x89\xf4\xdf\xd4g\x8

a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xef\u02ae\x9f\xf6M,\xd9[RI\xdc\xff\xe7\xfa\xa0\xa0\xc0\xe4M\x89\x15\xbeat\xe1\x91.\x00\x00\u07d4\xef\xcc\xe0k\xd6\b\x9d\x0eE\x8e\xf5a\xf5\xa6\x89H\n\xfep\x00\x89
\x86\xac5\x10R`\x00\x00\u07d4\xef\xe0g]\xa9\x8a]\xdap\u0356\x19k\x87\xf4\xe7&\xb43H\x89?\x19\xbe\xb8\xdd\x1a\xb0\x00\x00\u07d4\xef\xe8\xff\x87\xfc&\x0e\agc\x8d\xd5\xd0/\xc4g.\x0e\xc0m\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xef\xeb\x19\x97\xaa\xd2w\xcc3C\x0ea\x11\xed\tCY@H\xb8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xef\xee\xa0\x10uo\x81\xdaK\xa2[r\x17\x87\xf0X\x17\v\uff49\x01\u009c\x9c\xf7p\xef\x00\x00\u07d4\xef\xf5\x1dr\xad\xfa\xe1C\xed\xf3\xa4+\x1a\xecU\xa2\xcc\xdd\v\x90\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xef\xf8kQ#\xbc\xdc\x17\xedL\xe8\xe0[~\x12\xe5\x13\x93\xa1\xf7\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xef\xfc\x15\u41f1\xbe\xda\n\x8d\x13%\xbd\xb4\x17\"@\xdcT\n\x89\x03\x8599\xee\xe1\xde\x00\x00\xe0\x94\xf0\x11\x95\xd6W\xef<\x94.l\xb89l\xe5\xa2\v\\\xfa\x8b\x1e\x8a\x05ts\xd0]\xab\xae\x80\x00\x00\u07d4\xf0'\x96)Q\x01gB\x88\xc1\xd94g\x05=\x04\"\x19\xb7\x94\x89(\x1d\x90\x1fO\xdd\x10\x00\x00\u07d4\xf09h={=\"[\xc7\xd8\u07ed\xefc\x164A\xbeA\xe2\x89\x01\xdd\x1eK\xd8\xd1\xee\x00\x00\u07d4\xf0Jj7\x97\b\xb9B\x8dr*\xa2\xb0kw\xe8\x895\u03c9\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xf0M,\x91\xef\xb6\xe9\xc4_\xfb\xe7KCL\x8c_+\x02\x8f\x1f\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xf0W\xaaf\xcav~\xde\x12J\x1c[\x9c\xc5\xfc\x94\xef\v\x017\x89p\xa2K\u02b6\xf4]\x00\x00\u07d4\xf0[\xa8\u05f6\x859\xd930\v\xc9(\x9c=\x94t\xd0A\x9e\x89\x06\xda'\x02M\xd9`\x00\x00\u07d4\xf0\\\xee\xabeA\x05dp\x99Qw<\x84E\xad\x9fN\u01d7\x89\x10C\x16'\xa0\x93;\x00\x00\xe0\x94\xf0_\xcdL\rs\xaa\x16~US\xc8\xc0\xd6\xd4\xf2\xfa\xa3\x97W\x8a\x02\xd2\xd6l1p\xb2\x98\x00\x00\u07d4\xf0g\xe1\xf1\u0583UjL\xc4\xfd\f\x03\x13#\x9f2\xc4\xcf\u060965\u026d\xc5\u07a0\x00\x00\u07d4\xf0g\xfb\x10\u07f2\x93\u962b\xe5d\xc0U\xe34\x8f\x9f\xbf\x1e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf0h\xdf\xe9]\x15\xcd:\u007f\x98\xff\xa6\x88\xb44hB\xbe&\x90\x89D\n\xd8\x19\xe0\x97L\x00\x00\xe0\x94\xf0j\x85J<]\xc3m\x1cI\xf4\xc8}m\xb33\xb5~J\u074a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xf0y\xe1\xb1&_P\xe8\u0229\x8e\xc0\u01c1^\xb3\xae\xac\x9e\xb4\x89\x01\x16\xdc:\x89\x94\xb3\x00\x00\xe0\x94\xf0{\xd0\xe5\xc2\xcei\xc7\u0127$\xbd&\xbb\xfa\x9d*\x17\xca\x03\x8a\x01@a\xb9\xd7z^\x98\x00\x00\xe0\x94\xf0\x83*k\xb2U\x03\xee\xcaC[\xe3\x1b\v\xf9\x05\xca\x1f\xcfW\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\xf0\x9b>\x87\xf9\x13\xdd\xfdW\xae\x80I\xc71\u06e9\xb66\xdf\u00c9
\xf5\xb1\uab4d\x80\x00\x00\u07d4\xf0\xb14\v\x99oo\v\xf0\xd9V\x1c\x84\x9c\xaf\u007fD0\xbe\xfa\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xf0\xb1\xf9\xe2x2\xc6\xdei\x14\xd7\n\xfc#\x8ct\x99\x95\xac\xe4\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xf0\xb4i\xea\xe8\x9d@\f\xe7\xd5\xd6j\x96\x95\x03p6\xb8\x89\x03\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xf0\xb9\u0583\u03a1+\xa6\x00\xba\xac\xe2\x19\xb0\xb3\xc9~\x8c\x00\xe4\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xf0\xbe\x0f\xafMy#\xfcDF\"\u0458\f\xf2\u0650\xaa\xb3\a\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf0\xc0\x81\xdaR\xa9\xae6d*\xdf^\b
_\x05\xc5Ah\xa6\x89\x06\x04o7\xe5\x94\\\x00\x00\u07d4\xf0\xc7\r\rm\xabvc\xaa\x9e\xd9\xce\xeaV~\xe2\u01b0'e\x89qC\x8a\u0167\x91\xa0\x80\x00\u07d4\xf0\xcb\xef\x84\xe1ic\x00\x98\xd4\xe3\x01\xb2\x02\b\xef\x05\x84j\u0249\x0e\v\x83EPkN\x00\x00\u07d4\xf0\xd2\x16c\u0630\x17n\x05\xfd\xe1\xb9\x0e\xf3\x1f\x850\xfd\xa9_\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94\xf0\xd5\xc3\x1c\xcbl\xbe0\xc7\xc9\xea\x19\xf2h\xd1Y\x85\x1f\x8c\x9c\x8a\x03\x89O\x0eo\x9b\x9fp\x00\x00\u07d4\xf0\xd6L\xf9\xdf\tt\x113\xd1pH_\xd2K\x00P\x11\xd5

\x89\x1b\b\x93A\xe1O\xcc\x00\x00\u07d4\xf0\xd8X\x10^\x1bd\x81\x01\xac?\x85\xa0\xf8\"+\xf4\xf8
\x1dj\x89 \x86\xac5\x10R`\x00\x00\u07d4\xf0\xdcC\xf2\x05a\x91'P{+\x1c\x1c\xfd\xf3-(1\t
\x89\x10^\xb7\x9b\x94\x17\b\x80\x00\u07d4\xf0\xe1\u07e4*\u07ac/\x17\xf6\xfd\xf5\x84\xc9Hb\xfd
V3\x93\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf0\xe2d\x9c~j?,]\xfe3\xbb\xfb\xd9'\xca<5\n
X\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf0\xe7\xfb\x9eB\nS@\xd56\xf4\x04\b4O\xea\xef\xc0j\xef\x
8965\u026d\xc5\u07a0\x00\x00\u07d4\xf1\x04b\xe5\x8f\xcc\a\U000d5121\x87c\x94Q\x16~\x85\x9
2\x01\x89\t4\xdd]3\xbc\x97\x00\x00\xe0\x94\xf1\x06a\xff\x94\x14\x0f
>zH%rCy8\xbe\xc9\xc3\xf7\x8a\x04<3\xc1\x93ud\x80\x00\x00\u0794\xf1\x14\xff\r\x0f$\xef\xf8\x96\
xed\xdeTq\u07a4\x84\x82J\x99\xb3\x88\xbe -
j\x0e\xda\x00\x00\u07d4\xf1\x16\xb0\xb4h\x0fS\xabr\xc9h\xba\x80.\x10\xaa\x1b\xe1\x1d\u0209\x0
1\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xf1\x1c\xf5\xd3cto\xeehd\xd3\xca3m\xd8\x06y\xbb\x87\xae\
x8a\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xf1\x1e\x01\u01e9\xd1$\x99\x00_M\xaew\x16\tZ4\x17b
w\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xf1;\b0\x93\xbaVN-
\xc61V\x8c\xf7T\r\x9a\x0e\xc7\x19\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xf1O\x0e\xb8m\xb0\xebhu
?\x16\x91\x8e]K\x80t7\xbd>\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf1Qx\xff\xc4:\xa8\a\x0e\xce2~\x93\x0f\x80\x9a\xb1\xa5O\x9d\x89\n\xb6@9\x12\x
010\x00\x00\u07d4\xf1V\xdc\v*\x98\x1e[U\xd3\xf2\xf0;\x814\xe31\u06ed\xb7\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xf1]\x9dZ!\xb1\x92\x9ey\x03q\xa1\u007f\x16\xd9_\fie\\\x89lk\x93[\x8b\xbd@\
x00\x00\u07d4\xf1^\x18,O\xbb\xady\xbd\x934\"B\xd4\xdc\xcf+\xe5\x89%\x89i*\xe8\x89p\x81\xd0\x
00\x00\u07d4\xf1bM\x98\ve3o\xea\u0166\xd5A%\x00\\\xfc\xf2\xaa\u02c9lk\x93[\x8b\xbd@\x00\x0
0\u07d4\xf1g\xf5\x86\x8d\xcfB3\xa7\x83\x06\th,\xaf-
\xf4\xb1\xb8\a\x89\x81\xe5B\xe1\xa78?\x00\x00\u07d4\xf1m\xe1\x89\x1d\x81\x96F\x13\x95\xf9\xb
16&[;\x95F\xf6\xef\x89\x01\xb2\x8e\x1f\x98\xbb\u0380\x00\u07d4\xf1z\x92\xe06\x1d\xba\xce\xcd\
xc5\xde\r\x18\x94\x95Z\xf6\xa9\xb6\x06\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf1z\xdbt\x0fE\u02fd
\xe3\tN~\x13qo\x81\x03\xf5c\xbd\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf1\x8b\x14\xcb\xf6iC6\xd0\
xfe\x12\xac\x1f%\xdf-
\xa0\xc0]\xbb\x89\xd8\xd4`,&\xbfl\x00\x00\u07d4\xf1\x9b98\x9dG\xb1\x1b\x8a,?\x1d\xa9\x12M\xe
c\xff\xbe\xfa\xf7\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf1\x9f\x195\b9>M*\x12{
\xb2\x03\x1f9\xc8%\x81\u0189\xbd\xbdz\x83\xbd/l\x00\x00\u07d4\xf1\xa1\xf3
@yd\xfd<\x8f.,\u0224X\r\xa9O\x01\xea\x89ll!wU|D\x00\x00\u07d4\xf1\xb4\xec\xc65%\xf7C,=\x83O
\xfe+\x97\x0f\xbe\xb8r\x12\x89\xa2\xa2@h\xfa\u0340\x00\x00\u07d4\U000753ef\xfa\x87\x94\xf5\n
\xf8\xe8\x83\t\u01e6&TU\xd5\x1a\x8963\x03\"\xd5#\x8c\x00\x00\u07d4\xf1\xc8\u0129A\xb4b\x8c\r
l0\xfd\xa5dR\u065c~\x1bd\x89N\x8c\xea\x1e\xdeu\x04\x00\x00\u07d4\xf1\xda@so\x99\xd5\xdf;\x0
6\x8a]t_\xaf\xc6F?\u0271\x89\x06\x96\xca#\x05\x8d\xa1\x00\x00\u07d4\xf1\u070a\xc8\x10B\xc6z\
x9c<g\x92\xb20\xc4j\xc0\x16\xca\x10\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf1\xdfU\xdc\xc3J\x05\x10\x12\xb5u\u02d6\x8b\xc9\xc4X\xea\t\u0249\xd8\xd7&\x
b7\x17z\x80\x00\x00\xe0\x94\xf1\xe9\x80\xc5Y\xa1\xa8\xe5\xe5\nG\xf8\xff\xfd\xc7s\xb7\xe0jT\x8a
\x06_\xfb\xcd\xea\x04\xb7H\x00\x00\xe0\x94\xf1\xf3\x91\u0292\x80\x88\x17\xb7U\xa8\xb8\xf4\xe2
\xca\b\xd1\xfd\x11\b\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xf1\xf7f\xb0\xe4ms\xfc\xd4\xd5
.zr\xe1\xb9\x19\f\xc62\xb3\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xf2\x04\x952\xfdE\x8a\x83\
xca\x1b\xff.\ubb36\xd5\xcac\xf4\xa4\x89\u010c\x99\x1d\xc1T\\\x80\x00\u07d4\xf2\x06\xd3(\xe4q\x
d0\x11{$m*F\x19\x82w\t\xe9m\U000c98af=\xc0\x05CD\x00\x00\u07d4\xf2\f\x9a\x99\xb7GY\u05c2

\xf2\\\x1c\xec\xa8\x02\xa2~\vCl\x89Z\x87\xe7\xd7\xf5\xf6X\x00\x00\xe0\x94\xf2\x12}T\x18\x8f\xed\xef\x0f3\x8a_8\xc7\xffs\xad\x9foB\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xf2\x1341\xd1\u0663{\xa2\xf0v+\xc4\fZ\u030a\xa6\x97\x8e\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xf2\x15l\xbd\xd1Hy\x12\xf9\x00\xa7R=\xb5\xf7ba!\xbb\xa3\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xf2\x18\xbd\x84\x8e\xe7\xf9\u04cb\xfd\xd1\xc4\xeb.\xd2Ij\xe40_\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf2$\xeb\x90\v7\xb4I\x0e\xeej\vd
\xd8\\\x94}\x873\x894\x95tD\xb8@\xe8\x00\x00\u07d4\xf2)J\u06f6\xf0\xdc\xc7nc.\xbe\U0010ad1f\x12M\xbb\xa4\x89NC96\x00\xa7\xb1\x00\x00\u07d4\xf2/@x\xfe\xbb\xba\xa8\xb0\xe7\x8ed,\x8aB\xf3]C9\x05\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xf27\xef\x05&\x1c4\u05dc\xc2+\x86\r\xe0\xf1\u007fy<8`\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf2<{\f\xb8\xcdY\xb8+\u0610dJW\xda\xf4\f\x85\xe2x\x89\x02\xb6j\xaf\xe3&\xff\x00\x00\u07d4\xf2=\x01X\x9e\xb1-
C\x9ftH\xffT0u)\U0005114d\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf2>\\c2!\xa8\xf76>e\x87\f\x9f(t$\u04a9`\x89J\xcfX\xe0rW\x10\x00\x00\u07d4\xf2B\u0684]B\u053fw\x9a\x00\xf2\x95\xb4\aP\xfeI\xea\x13\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xf2RY\xa5\xc99\xcd%\x96l\x9bc\x03\xd3s\x1cS\u077cL\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf2^Lp\xbcFV2\u021eV%\xa82\xa7r/k\xff\xab\x89\xf3K\x82\xfd\x8e\x91
\x00\x00\u07d4\xf2k\xce\xdc\xe3\xfe\xad\u03a3\xbc>\x96\xeb\x10@\xdf\xd8\xff\u1809*\x03I\x19\u07ff\xbc\x00\x00\u07d4\xf2py%v\xf0]QD\x93\xff\xd1\xf5\xe8K\xecK-
\xf8\x10\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xf2s,\xf2\xc1;\x8b\xb8\xe7I*\x98\x8f_\x89\xe3\x82s\xdd\u0209
\x86\xac5\x10R`\x00\x00\xe0\x94\xf2t.hY\xc5i\xd5\xf2\x10\x83Q\xe0\xbfM\xca5*H\xa8\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xf2\x81:d\xc5&]\x02\x025\u02dc1\x9bl\x96\xf9\x06\xc4\x1e\x89\x12\xf99\u025e\u06b8\x00\x00\u07d4\xf2\x87\xffR\xf4a\x11z\xdb>\x1d\xaaq\x93-
\x14\x93\xc6_.\x89\xc5S%\xcat\x15\xe0\x00\x00\u07d4\xf2\xab\x11au\x02D\xd0\xec\xd0H\xee\r>Q\xab\xb1C\xa2\xfd\x89B\xfe+\x90ss\xbc\x00\x00\u07d4\xf2\xb4\xab,\x94'\xa9\x01^\xf6\xee\xff\xf5\xed\xb6\x019\xb7\x19\u0449&\u06d9*;\x18\x00\x00\x00\u07d4\xf2\xc0>*8\x99\x8c!d\x87`\xf1\xe5\xae~\xa3\a}\x85\"\x89\x8f?q\x93\xab\a\x9c\x00\x00\u0794\xf2\u0090N\x9f\xa6d\xa1\x1e\xe2VV\xd8\xfd,\xc0\u0665\"\xa0\x88\xb9\x8b\xc8)\xa6\xf9\x00\x00\u07d4\xf2\xc3b\xb0\xef\x99\x1b\xc8/\xb3nf\xffu\x93*\xe8\u0742%\x89\x04\x02\xf4\xcf\xeeb\xe8\x00\x00\u07d4\xf2\xd0\xe9\x86\xd8\x14\xea\x13\xc8\xf4f\xa0S\x8cS\u0712&Q\xf0\x89J\xcfX\xe0rW\x10\x00\x00\xe0\x94\xf2\u04775w$\xecL\x03\x18[\x87\x9bc\xf5~&X\x91S\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xf2\xd5v<\xe0s\x12~,\xed\xdeo\xab\xa7\x86\xc7<\xa9AA\x8a\x01\xacB\x86\x10\x01\x91\xf0\x00\x00\xe0\x94\xf2\u055c\x89#u\x90s\xd6\xf4\x15\xaa\xf8\xeb\x06_\xf2\U000f614a\x01\xab,\xf7\xc9\xf8~
\x00\x00\u07d4\xf2\xe9\x9f\\\xbb\x83kz\xd3bGW\x1a0,\xbeKH\x1ci\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf2\xed>w%J\u02c3#\x1d\xc0\x86\x0e\x1a\x11$+\xa6'\u06c9kV\x05\x15\x82\xa9p\x00\x00\xe0\x94\xf2\xed\xde7\xf9\xa8\u00dd\u07a2My\xf4\x01WW\xd0k\xf7\x86\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xf2\xef\xe9e`\xc9\xd9{r\xbd6DxC\x88\\\x1d\x90\xc21\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf2\xfb\xb6\u0607\xf8\xb8\xcc:\x86\x9a\xba\x84\u007f=\x1fd<S\u0589\xd8\xc9F\x00c\xd3\x1c\x00\x00\u07d4\xf3\x03Cg\xf8}$\xd3\a\u007f\xa9\xa2\u328b\f\xcb\x11\x04\xef\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xf3\x03\u0568\x16\xaf\xfd\x97\xe8=\x9eM\xac/y\a+\xb0\t\x8f\x894\n\xad!\xb3\xb7\x00\x00\x00\u07d4\xf3\x15\x98f\u00bc\x86\xbb\xa4\x0f\x9ds\xbb\x9

9\xf1\xee\xe5{\xb9\u05c965\u026d\xc5\u07a0\x00\x00\u07d4\xf3\x16\xef\x1d\xf2\xffMl\x18\b\u06e6c\uc013iyh\xe0\x89aFMl\u0700\xf0\x00\x00\xe0\x94\xf3-
%\xeb\x0e\xa2\xb8\xb3\x02\x8aLz\x15]\xc1\xaa\xe8exM\x8a\x015\x93\xa9)\u007f\xda\xd6\x00\x00\xe0\x94\xf32\xc0\xf3\xe0Z'\xd9\x12o\u0436A\xa8\xc2\xd4\x06\x06\b\xfd\x8a\x01\x0f\x1bb\xc4\xd9dN\x80\x00\xe0\x94\xf38E\x9f2\xa1Y\xb2=\xb3\n\xc35v\x9a\xb25\x1a\xa6<\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4\xf3>\xfcc\x97\xaae\xfbS\xa8\xf0z\x0f\x89:\xae0\xe8\xbc\xee\x89|\xf28\x1fa\x9f\x15\x00\x00\u07d4\xf3@\x83\xec\xea8P\x17\xaa@\xbd\xd3^\xf7\xef\xfbL\xe7v-
\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xf3F\xd7\u0792t\x1c\b\xfcX\xa6M\xb5[\x06-\xde\x01-
\x14\x89\x0f\xffk\x1fv\x1em\x00\x00\xe0\x94\xf3U\xd3\xec\f\xfb\x90}\x8d\xbb\x1b\xf3FNE\x81(\x19\v\xac\x8a\x01\v\x04n\u007f\r\x80\x10\x00\x00\u07d4\xf3m\xf0/\xbd\x89`sG\xaf\xce)i\xb9\xc4#jX\xa5\x06\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf3s\xe9\u06ac\f\x86u\xf5;yz\x16\x0fo\xc04\xaek#\x89\x05k\xc7^-c\x10\x00\x00\u07d4\xf3{BeG\xa1d-
\x8032H\x14\xf0\xed\xe3\x11O\xc2\x12\x89\x15\xbeat\xe1\x91.\x00\x00\u07d4\xf3{\xf7\x8cXu\x15G\x11\xcbd\r7\xeam(\xcf\xcb\x12Y\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xf3\x82\xdfX1U\xd8T\x8f?\x93D\f\xd5\xf6\x8c\xb7\x9d`&\x8a8u}\x02\u007f\xc1\xfd\\\x00\x00\xe0\x94\xf3\x82\xe4\xc2\x04\x10\xb9Q\b\x9e\x19\xba\x96\xa2\xfe\xe3\xd9\x1c\xce~\x8a\x01\x11\xfaV\xee\u00a88\x00\x00\xe0\x94\xf3\x8al\xa8\x01hS~\x97M\x14\xe1\xc3\xd19\x90\xa4L,\x1b\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\u07d4\xf3\x9a\x9dz\xa3X\x1d\xf0~\xe4'\x9a\xe6\xc3\x12\xef!\x036X\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xf3\xb6h\xb3\xf1M\x92\x0e\xbc7\x90\x92\u06d8\x03\x1bg\xb2\x19\xb3\x89\n\xd6\xee\xdd\x17\xcf;\x80\x00\u07d4\U000fe679\x10<\xe7U\n\xa7O\xf1\xdb\x18\xe0\x9d\xfe2\xe0\x05\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf3\xc1\xab\u049d\xc5{A\xdc\x19-
\x0e8M\x02\x1d\xf0\xb4\xf6\u0509\x97\xae\f\u07cf\x86\xf8\x00\x00\u07d4\xf3\xc4qm\x1e\xe5'\x9a\x86\xd0\x16:\x14a\x81\x81\xe1a6\u01c965\u026d\xc5\u07a0\x00\x00\xe0\x94\xf3\u030b\xcbU\x94e\xf8\x1b\xfeX;\u05eb\n#\x06E;\x9e\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xf3\u0588\xf0k\xbd\xbfP\xf9\x93,AE\xcb\xe4\x8e\xcd\xf6\x89\x04\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xf3\xdb\xcf\x13Z\u02dd\xee\x1aH\x9cY<\x02O\x03\u00bb\xae\u0389lk\x93[\x8b\xbd@\x00\x00\u07d4\xf3\xde_&\xefj\xde\xd6\xf0m;\x91\x13F\xeep@\x1d\xa4\xa0\x89\x13:\xb3}\x9f\x9d\x03\x00\x00\u07d4\xf3\xdfc\xa9q\x99\x93308;>\xd7W\v\x96\u0101#4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf3\xe7OG\f}:?\x003x\x0fv\xa8\x9f>\xf6\x91\xe6\u02c9\xa3\xcf\xe61\xd1Cd\x00\x00\u07d4\xf3\xeb\x19H\xb9Q\xe2-
\xf1ax)\xbf;\x8d\x86\x80\xeckh\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xf3\xf1\xfa9\x18\xca4\xe2\xcf~\x84g\v\x1fM\x8e\xca\x16\r\xb3\x89$\xdc\xe5M4\xa1\xa0\x00\x00\u07d4\xf3\xf2O\u009e@?\xc0\xe8\xf5\xeb\xbbU4&\xf7\x82p\xa2\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xf3\xfar5R\xa5\xd0Q.+b\xf4\x8d\xca{+\x81\x050[\x89\amA\xc6$\x94\x84\x00\x00\u07d4\xf3\xfeQ\xfd\xe3D\x13\xc73\x18\xb9\xc8T7\xfe~\x82\x0fV\x1a\x896b2\\\u044f\xe0\x00\x00\u07d4\xf4\x00\xf9=_\\~?\xc3\x03\x12\x9a\xc8\xfb\f/xd\a\xfa\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf4\v\x13O\xea\"\u01b2\x9c\x84W\xf4\x9f\x00\x0f\x9c\xdax\x9a\u06c9
\x86\xac5\x10R`\x00\x00\u07d4\xf4\x15W\xdf\u07f1\xa1\xbd\xce\xfe\xfe.\xba\x1e!\xfe\nJ\x99B\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf4\x17z\r\x85\u050b\x0e&B\x11\xce*\xa2\xef\xd3\xf1\xb4\u007f\b\x89\xc2\xcc\xca&\xb7\xe8\x0e\x80\x00\u07d4\xf4/\x90R1\xc7p\xf0\xa4\x06\xf2\xb7h\x87\u007f\xb4\x9e\xee\x0f!\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\u07d4\xf42\xb9\u06ef\x11\xbd\xbds\xb6Q\x9f

\xc0\xa9\x04\x19\x87q\xaa\u0189\b=lz\xabc`\x00\x00\u07d4\xf4=\xa3\xa4\xe3\xf5\xfa\xb1\x04\u029b\xc1\xa0\xf7\xf3\xbbJV\xf3Q\x89lj\xccg\u05f1\xd4\x00\x00\xe0\x94\xf4G\x10\x8b\x98\xdfd\xb5~\x87\x103\x88\\\x1a\xd7\x1d\xb1\xa3\xf9\x8a\x01v\xf4\x9e\xad4\x83P\x80\x00\u07d4\xf4O\x85Q\xac\xe93r\a\x12\xc5\u0111\u0376\xf2\xf9Qsl\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u0794\xf4V\x05Z\x11\xab\x91\xfff\x8e.\xc9\"\x96\x1f*#\xe3\xdb%\x88\xfc\x93c\x92\x80\x1c\x00\x00\u07d4\xf4V\xa7[\xb9\x96U\xa7A,\xe9}\xa0\x81\x81m\xfd\xb2\xb1\xf2\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf4[\x1d\xcb.A\xdc'\xff\xa0$\u06ad\xf6\x19\xc1\x11u\xc0\x87\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4\xf4c\xa9\f\xb3\xf1>\x1f\x06CB66\xbe\xab\x84\xc1#\xb0m\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xf4h\x90n~\xdffJ\xb0\u063e=\x83\xebz\xb3\xf7\xff\xdcx\x89\\(=A\x03\x94\x10\x00\x00\u07d4\xf4i\x80\u3929\u049ajn\x90`E7\xa3\x11K\xcb(\x97\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf4kk\x9c|\xb5R\x82\x9c\x1d=\xfd\x8f\xfb\x11\xaa\xba\xe7\x82\xf6\x89\x01#n\xfc\xbc\xbb4\x00\x00\u07d4\xf4v\xe1&\u007f\x86$|\xc9\b\x81o.z\xd58\x8c\x95-
\xb0\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xf4v\xf2\xcbr\b\xa3.\x05\x1f\xd9N\xa8f)\x92c\x82\x87\xa2\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xf4{\xb14\xda0\xa8\x12\xd0\x03\xaf\x8d\u0338\x88\xf4K\xbfW$\x8a\x01\x19Y\xb7\xfe3\x95X\x00\x00\u07d4\xf4\x83\xf6\a\xa2\x1f\xcc(\x10\n\x01\x8cV\x8f\xfb\xe1@8\x04\x10\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xf4\x8e\x1f\x13\xf6\xafM\x84\xb3q\xd7\xdeK'=\x03\xa2c'\x8e\x89
\x86\xac5\x10R`\x00\x00\xe0\x94\xf4\x9cG\xb3\xef\xd8knj[\xc9A\x8d\x1f\x9f\xec\x81Ki\xef\x8a\x04<3\xc1\x93ud\x80\x00\x00\xe0\x94\xf4\x9fo\x9b\xaa\xbc\x01\x8c\x8f\x8e\x11\x9e\x01\x15\xf4\x91\xfc\x92\xa8\xa4\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xf4\xa3g\xb1f\u0499\x1a+\xfd\xa9\xf5dc\xa0\x9f%,\x1b\x1d\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf4\xa5\x1f\xceJ\x1d[\x94\xb0q\x83\x89\xbaNx\x14\x13\x9c\xa78\x89\x10CV\x1a\x88)0\x00\x00\u07d4\xf4\xa9\xd0\f\xef\xa9{zX\xef\x94\x17\xfcbg\xa5\x06\x909\xee\x89\x01.\x89(\u007f\xa7\x84\x00\x00\u07d4\xf4\xaa\xa3\xa6\x16>7\x06W{I\xc0v~\x94\x8ah\x1e\x16\xee\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf4\xb1bn$\xf3\v\xca\xd9'<R\u007f\xccqK]\x00{\x8f\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xf4\xb4\x91\x00uwr\xf3<\x17{\x9av\xba\x95\"l\x8f=\u060a\x01k5-
\xa5\xe0\xed0\x00\x00\xe0\x94\xf4\xb6\xcd\xcf\xcb$#\v3}w\r\xf6\x03M\xfb\xd4\xe1P?\x8a\x04\x05\xfd\xf7\u5bc5\xe0\x00\x00\u07d4\xf4\xb7Y\u030a\x1cu\xf8\bI\xeb\xbc\u0687\x8d\xc8\xf0\xd6m\xe4\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xf4\xbajF\xd5Q@\xc49\xcb\xcf\al\xc6W\x13bb\xf4\xf8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf4\xd6z\x90D\xb45\xb6n\x89w\xff9\xa2\x8d\u013dSr\x9a\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf4\xd9vd\xccN\xec\x9e\xdb\xe7\xfa\t\xf4u\nf;P}y\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xf4\xdc{\xa8T\x80\xbb\xb3\xf55\xc0\x95h\xaa\xa3\xafo7!\u018a\x01\x87\x1f\xb60~5\xe5\x00\x00\xe0\x94\xf4\xeb\xf5\v\xc7\xe5O\x82\u9e7d$\xba\xef)C\x8e%\x9c\xe6\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xf4\uc397\xa2\n\xa5\xf8\xdd oU
~\x06\xb8\x13\xdf,\xc0\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf4\ud10e\xc9as\x9c,~5/C[\xa7\n|\xd5\xdb8\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf4\xfcM9\xbc\f,@h\xa3m\xe5\x0eJ\xb4\xd4\xdb~4\n\x89\x01`7\u07c7\xefj\x00\x00\xe0\x94\xf5\x04\x94:\xaf\x16yn\v4\x1b\xbc\xdf!\xd1\x1c\u0146\xcd\u044a\x01\xe7\xe4\x17\x1b\xf4\u04e0\x00\x00\u07d4\xf5\x06\x1e\xe2\xe5\xee&\xb8\x15P6w\x13\x0e\x1d\xe0zR\xdb\a\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xf5\tU~\x90\x18?\xbf\x0f\x06Q\xa7\x86H{\xccB\x8b\xa1u\x89\n\x84Jt$\xd9\x

c8\x00\x00\xe0\x94\xf5\n\xbb\u052aE\xd3\xeb\x88QTe\xa8\xba\v1\x0f\u0675!\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\xf5\n\xe7\xfa\xb4\u03f5\xa6F\xee\x04\u03ad\xf9\xbf\x9d\u0568\xe5@\x89\xd8\xd6|/X\x95H\x00\x00\u07d4\xf5\f\xba\xfd9~\xddUl\x06x\x98\x8c\xb2\xaf\\&\x17\u0889&\xd0~\xfex+\xb0\x00\x00\xe0\x94\xf5\x1f\xde\xd8\n\xcbP(\x90\xe8sit\x1f7\"QL\xef\xff\x8a\x04<4V\xca<m\x11\x00\x00\xe0\x94\xf5*X\x82\xe8\x92}\x94K5\x9b&6k\xa2\xb9\xca\u03fa\xe8\x8a\x05KA\xce/\xe6;\xa8\x00\x00\xe0\x94\xf5,\nxw4_\xe0\xc23\xbb\x0f\x04\xfdj\xb1\x8bo\x14\xba\x8aT\xcb\xe5Y\x89\xf3\x8d\xe0\x00\x00\u07d4\xf5C~\x15\x80\x90\xb2\xa2\u058f\x82\xb5JXd\xb9]\xd6\xdb\xea\x89\xd9l\x16p;+\xfe\x00\x00\xe0\x94\xf5L\x19\xd9\xef8s\xbf\xd1\xf7\xa6\"\xd0-
\x86$\x9a2\x8f\x06\x8a\t`\xae\x12z\xf3/\xb2\x80\x00\u07d4\xf5P\x01x\u02d9\x8f\x12d\x17\x83\x1a\b\xc2\u05eb\xff\xf6\xab_\x89F\xf4\xf4\xa5\x87Z\x9f\x80\x00\u07d4\xf5SH\x15\xdcc^\xfa\\\xc8K*\xc74r>!\xb2\x93r\x89U\xa6\xe7\x9c\xcd\x1d0\x00\x00\u07d4\xf5U\xa2{\xb1\xe2\xfdN,\u01c4\xca\ue493\x9f\xc0n/\u0249lk\x93[\x8b\xbd@\x00\x00\u07d4\xf5X\xa2\xb2\xdd&\u0755\x93\xaa\xe0E1\xfd<<\u00c5Kg\x89\n\xbb\xcdN\xf3wX\x00\x00\u07d4\xf5`H\xdd!\x81\u0523od\xfc\xec\xc6!T\x81\xe4*\xbc\x15\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf5dB\xf6\x0e!i\x13\x95\u043f\xfa\xa9\x19M\xca\xff\x12\u2dc9\x0e\x189\x8ev\x01\x90\x00\x00\u07d4\xf5yqJE\xeb\x8fR\xc3\xd5{\xbd\xef\xd2\xc1[./\x11\u07c9T\x91YV\xc4\t`\x00\x00\u07d4\xf5\x93\xc6R\x85\xeek\xbdf7\U000fe3c9\xad@\u0509\xf6U\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xf5\x98\xdb.\t\xa8\xa5\xee}r\r+\\C\xbb\x12m\x11\xec\u0089\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xf5\x9d\xab\x1b\xf8\xdf\x112~a\xf9\xb7\xa1KV:\x96\xec5T\x8a\x01EB\xba\x12\xa37\xc0\x00\x00\xe0\x94\xf5\x9f\x9f\x02\xbb\u024e\xfe\t~\xab\xb7\x82\x10\x97\x90!\x89\x8b\xfd\x8a\x02\x1e\x17\x1a>\xc9\xf7,\x00\x00\u07d4\xf5\xa5E\x9f\xcd\xd5\xe5\xb2s\x83\r\xf8\x8e\xeaL\xb7}\xda\u07f9\x89\x04\t\xe5+H6\x9a\x00\x00\u07d4\xf5\xa7gj\xd1H\xae\x9c\x1e\xf8\xb6\xf5\xe5\xa0\xc2\xc4s\xbe\x85\v\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf5\xb0h\x98\x9d\xf2\x9c%5w\xd0@Z\xden\x0eu(\xf8\x9e\x89WG=\x05\u06ba\xe8\x00\x00\u07d4\xf5\xb6\xe9\x06\x1aN\xb0\x96\x16\aw\xe2gb\xcfH\xbd\u0635]\x89\r\xc5_\xdb\x17d{\x00\x00\u07d4\xf5\xcf\xfb\xbabN~\xb3!\xbc\x83\xc6\f\xa6\x81\x99\xb4\xe3fq\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf5\xd1ER\xb1\xdc\xe0\xd6\xdc\x1f2\r\xa6\xff\u02231\xcdo\f\x89Hz\x9a0E9D\x00\x00\xe0\x94\xf5\xd6\x1a\xc4\u0295G^[{\xff\xd5\xf2\xf6\x90\xb3\x16u\x96\x15\x8a\x06\x92\xae\x88\x97\b\x1d\x00\x00\x00\u07d4\xf5\xd9\xcf\x00\xd6X\xddEQzH\xa9\xd3\xf5\xf63T\x1aS=\x89\x06O_\xdfIOx\x00\x00\u07d4\xf5\xea\xdc\xd2\u0478ez\x12\x1f3\xc4X\xa8\xb1>v\xb6U&\x89\r\x8b\x0fZZ\xc2J\x00\x00\u07d4\xf6\a\xc2\x15\r>\x1b\x99\xf2O\xa1\xc7\xd5@\xad\xd3\\N\xbe\x1e\x89\xa7\xf1\xaa\a\xfc\x8f\xaa\x00\x00\u07d4\xf6\v\xd75T>k\xfd.\xa6\xf1\x1b\xffbs@\xbc\x03Z#\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf6\f\x1bE\xf1d\xb9X\x0e
'Z\\9\xe1\xd7\x1e5\xf8\x91\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xf6\x0fb\xd797\x95?\xef5\x16\x9e\x11\xd8r\xd2\xea1~\xec\x8a\x01!\xeah\xc1\x14\xe5\x10\x00\x00\u07d4\xf6\x12\x83\xb4\xbd\x85\x04\x05\x8c\xa3`\u94d9\x9bb\xcb\xc8\xcdg\x89\r\xd2\xd5\xfc\xf3\xbc\x9c\x00\x00\u07d4\xf6\x17\xb9g\xb9\xbdH_v\x95\xd2\xefQ\xfbw\x92\u0618\xf5\x00\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf6\x18\u0671\x04A\x14\x80\xa8c\xe6#\xfcU#-
\x1aOH\xaa\x89\x0eh\x9emD\xb1f\x80\x00\u07d4\xf6\"\u5126b>\xaa\xf9\x9f+\xe4\x9eS\x80\xc5\xcb\xcf\\\u0609\n\u05ce\xbcZ\xc6 \x00\x00\u07d4\xf62\xad\xffI\r\xa4\xb7-
\x126\xd0KQ\x0ft\xd2\xfa\xa3\u0349K\xe4\xe7&{j\xe0\x00\x00\u07d4\xf69\xac1\u069fg'\x1b\xd1\x04\x02\xb7eN\\\xe7c\xbdG\x89\x15\xaf\x0fB\xba\xf9&\x00\x00\u07d4\xf6:W\x9b\xc3\xea\u00a9I\x0

4\x10\x12\x8d\xbc\xeb\xe6\xd9\u0782C\x89P\xc5\xe7a\xa4D\b\x00\x00\u07d4\xf6E\xdd|\x89\x00\x93\xe8\xe4\u022a\x92\xa6\xbb55\"\xd3\u0718\x89\aC\x9f\xa2\t\x9eX\x00\x00\xe0\x94\xf6H\xea\x89\xc2u%q\x01r\x94Ny\xed\xff\x84x\x03\xb7u\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xf6JJ\xc8\xd5@\xa9(\x9ch\xd9`\xd5\xfb|\xc4Zw\x83\x1c\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf6N\xcf!\x17\x93\x1cmSZ1\x1eO\xfe\xae\xf9\u0514\x05\xb8\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\xf6O\xe0\x93\x9a\x8d\x1e\xea*\x0e\u035a\x970\xfdyX\xe31\t\x89\x01\x1d\xe1\xe6\xdbE\f\x00\x00\u07d4\xf6V\x16\xbe\x9c\x8by~t\x15\"|\x918\xfa\xa0\x89\x17B\u05c9*\xd3s\xcef\x8e\x98\x00\x00\u07d4\xf6W\xfc\xbeh.\xb4\xe8\xdb\x15.\u03c9$V\x00\vQ=\x15\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xf6X\x19\xacL\xc1L\x13\u007f\x05\xddyw\xc7\xda\xe0\x8d\x1aJ\xb5\x89\x05\x87\x88\u02d4\xb1\xd8\x00\x00\u07d4\xf6{\xb8\xe2\x11\x8b\xbc\u0550'fn\xed\xf6\x94>\xc9\xf8\x80\xa5\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xf6\x84d\xbfd\xf2A\x13V\xe4\xd3%\x0e\xfe\xfe\\P\xa5\xf6[\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xf6\x86x[\x89r\va\x14_\ua017\x8dj\u030e\v\xc1\x96\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xf6\x8c^3\xfa\x97\x13\x9d\xf5\xb2\xe68\x86\xce4\xeb\xf3\u45dc\x89\xb3\xfaAi\xe2\xd8\xe0\x00\x00\u07d4\xf6\xa8cWW\xc5\xe8\xc14\xd2\r\x02\x8c\xf7x\u03c6\t\xe4j\x89O\x1dw/\xae\xc1|\x00\x00\u07d4\xf6\xb7\x82\xf4\xdc\xd7E\xa6\xc0\xe2\xe00`\x0e\x04\xa2K%\xe5B\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\xe0\x94\xf6\xbc7\xb1\u04a3x\x8dX\x9bm\xe2\x12\xdc\x17\x13\xb2\xf6\u738a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xf6\xc3\u010a\x1a\xc0\xa3G\x99\xf0M\xb8n\u01e9u\xfewh\xf3\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf6\xd2]?=\x84m#\x9fR_\xa8\xca\xc9{\xc45x\u06ec\x890\x92\u007ft\xc9\xde\x00\x00\x00\u07d4\xf6\xea\xacp2\u0512\xef\x17\xfd`\x95\xaf\xc1\x1dcOV\xb3\x82\x89\x1b\x1bk\u05efd\xc7\x00\x00\xe0\x94\xf6\xea\xd6}\xbf[~\xb13X\xe1\x0f6\x18\x9dS\xe6C\xcf\u03ca\bxg\x83&\xea\xc9\x00\x00\x00\u07d4\xf6\xf1\xa4C\t\x05\x1ck%\xe4}\xff\x90\x9b\x17\x9b\xb9\xabY\x1c\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xf7\x03(\xef\x97b_\xe7E\xfa\xa4\x9e\xe0\xf9\u052a;\r\xfbi\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xf7\n\x99\x8aq{3\x8d\x1d\u0658T@\x9b\x1a3\x8d\ue930\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf7\rcz\x84\\\x06\xdbl\u0711\xe67\x1c\xe7\xc48\x8ab\x8e\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xf7\x15R\x13D\x98\x92tK\xc6\x0f.\x04@\a\x88\xbd\x04\x1f\u0749\x03\x9f\xba\xe8\xd0B\xdd\x00\x00\xe0\x94\xf7\x1bE4\xf2\x86\xe40\x93\xb1\xe1^\xfe\xa7I\xe7Y{\x8bW\x8a\x16\x1c\x13\xd34\x1c\x87(\x00\x00\u07d4\xf74\xec\x03rM\xde\xe5\xbbRy\xaa\x1a\xfc\xf6\x1b\f\xb4H\xa1\x89\xe5\xbf,\u0270\x97\x80\x00\x00\u07d4\xf76\u0716v\x00\x128\x8f\xe8\x8bf\xc0n\xfeW\xe0\xd7\xcf\n\x89q\xd7Z\xb9\xb9 P\x00\x00\u07d4\xf7:\xc4l
;\xe1S\x81\x11\xb1Q\xec\x82
\u01c6\xd8AD\x89\x0f\xf77x\x17\xb8+\x80\x00\u07d4\xf7=\xd9\xc1B\xb7\x1b\xce\x11\xd0n0\xe7\xe7\xd02\xf2\uc71e\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf7A\x8a\xa0\xe7\x13\xd2H\"\x87v\xb2\xe7CB\"\xaeu\u3949lk\x93[\x8b\xbd@\x00\x00\u07d4\xf7Nn\x14S\x82\xb4\u06c2\x1f\xe0\xf2\u0643\x88\xf4V\t\u019f\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xf7P\f\x16o\x8b\xea/\x824v\x06\xe5\x02K\xe9\xe4\xf4\u0399\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xf7W\xfc\x87
\xd3\xc4\xfaRw\a^`\xbd\\A\x1a\xeb\xd9w\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf7[\xb3\x9cy\x97y\xeb\xc0J3m&\r\xa61F\xed\x98\u0409\x01Z\xf1\u05cbX\xc4\x00\x00\xe0\x94\xf7h\xf3!\xfdd3\xd9kO5M<\xc1e,\x172\xf5\u007f\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xf7oi\xce\xe4\xfa\xa0\xa6;0\xae\x1ex\x81\xf4\xf7\x15ep\x10\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf7w6\x1a=\u062bb\xe5\xf1\xb9\xb0GV\x8c\xc0\xb5UpL\x8965\u026d\xc5\u07a0\

x00\x00\u07d4\xf7|{\x84QI\xef\xba\x19\xe2a\xbc|u\x15y\b\xaf\xa9\x90\x89lk\x93[\x8b\xbd@\x00\x
00\u07d4\xf7\u007f\x95\x87\xffz-
r\x95\xf1\xf5q\u0206\xbd3\x92jR|\x89lh\xcc\u041b\x02,\x00\x00\u07d4\xf7\x82X\xc1$\x81\xbc\xdd\
u06f7*\x8c\xa0\xc0C\tra\xc6\u0149\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xf7\x98\xd1m\xa4\xe4`\
xc4`\xcdH_\xae\x0f\xa0Y\x97\b\ub08965\u026d\xc5\u07a0\x00\x00\u07d4\xf7\xa1\xad\xe2\xd0\xf5
)\x12=\x10U\xf1\x9b\x17\x91\x9fV!Ng\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf7\xac\xff\x9
3K\x84\xda\ti\xdc7\xa8\xfc\xf6C\xb7\xd7\xfb\xedA\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xf7\xb1Q\x
cc^W\x1c\x17\xc7e9\xdb\xe9\x96L\xbbo\xe5\xdey\x89tq|\xfbh\x83\x10\x00\x00\u07d4\xf7\xb2\x9b\
x82\x19\\\x88-
\xabx\x97\u00ae\x95\xe7w\x10\xf5xu\x89w5Aa2\xdb\xfc\x00\x00\u07d4\xf7\xbcLD\x91\rZ\xed\xd6
n\xd25U8\xa6\xb1\x93\xc3a\xec\x89\x05A\xde,-
\x8db\x00\x00\u07d4\xf7\xc0\f\xdb\x1f\x02\x03\x10\u056c\xab{Ij\xaaD\xb7y\b^\x89Z\x87\xe7\xd7\x
f5\xf6X\x00\x00\u07d4\xf7\xc1\xb4C\x96\x8b\x11{]\u0677UW/\xcd9\xca^\xc0K\x89\x18\xb9h\u0092\xf1\xb5\x00\x00\xe0\x94\xf7\xc5\x0f\x92*\xd1ka\xc6\u047a\xa0E\xed\x81h\x15\xba\u010f\x8a\x02\xa99j\x97\x84\xad}\x00\x00\u07d4\xf7\xc7\b\x01Pq\xd4\xfb\n:*\t\xa4]\x15c\x96\xe34\x9e\x89\xa2\xa1]\tQ\x9b\xe0\x00\x00\u07d4\xf7\xcb\u06e6\xbel\xfeh\xdb\xc2<+\x0f\xf50\xee\x05\"o\x84\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xf7\xd0\xd3\x10\xac\xea\x18@a8\xba\xaa\xbb\xfe\x05q\xe8\r\xe8_\x89Hz\x9a0E9D\x00\x00\u07d4\xf7\u05ef
LV\xf3\x1f\xd9C\x98\xe4\r\xf1\x96K\u063f\x12<\x89\b!\xd2!\xb5)\x1f\x80\x00\u07d4\xf7\xdc%\x11\x96\xfb\u02f7|\x94}|\x19F\xb0\xffe\x02\x1c\xea\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xf7\xe4Z\x12\xaaq\x1cp\x9a\xce\xfe\x95\xf3;xa-
\xd2\x8a\x0e\x06U\xe2\xf2k\xc9\x18\x00\x00\u07d4\xf7\xf4\x89\x8cLRm\x95_!\xf0U\xcbnG\xb9\x15\xe5\x19d\x89|\b`\xe5\xa8\r\xc0\x00\x00\u07d4\xf7\xf9\x1ez\xcb[\x81)\xa3\x06\x87|\xe3\x16\x8eoC\x8bf\xa1\x89\t\x8a}\x9b\x83\x14\xc0\x00\x00\u07d4\xf7\xfcE\xab\xf7oP\x88\xe2\u5d68\xd12\xf2\x8aMN\xc1\xc0\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf8\x06:\xf4\xcc\x1d\xd9a\x9a\xb5\u063f\xf3\xfc\xd1\xfa\xa8H\x82!\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf8\bnBf\x1e\xa9)\xd2\u0761\xablt\x8c\xe3\x05]\x11\x1e\x8965\u026d\xc5\u07a0\x00\x00\xe0\x94\xf8\bw\x86\xb4-
\xa0N\xd6\xd1\xe0\xfe&\xf6\xc0\xee\xfe\x1e\x9fZ\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xf8\r6\x19p/\xa5\x83\x8cH9\x18Y\xa89\xfb\x9c\xe7\x16\x0f\x89l\a\xa7\u0471np\x00\x00\u07d4\xf8\x14y\x9fm\xdfM\xcb)\xc7\xee\x87\x0eu\xf9\xcc-
52m\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xf8\x15\xc1\n\x03-
\x13\xc3K\x89v\xfan;\xd2\xc9\x13\x1a\x8b\xa9\x89Hz\x9a0E9D\x00\x00\u07d4\xf8\x16\"\xe5WW\xda\xeafu\x97]\xd958\xda}\x16\x99\x1e\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf8$\xee3\x1eJ\xc3\xccXv\x939[W\xec\xf6%\xa6\xc0\u0089V\xc9]\xe8\xe8\xca\x1d\x00\x00\u07d4\xf8'\xd5n\xd2\xd3'
\u052b\xf1\x03\xd6\xd0\xefM;\xcdU\x9b\x89\x01l\x80\x06W\x91\xa2\x80\x00\u07d4\xf8)\x85\x91R>P\xb1\x03\xf0\xb7\x01\xd6#\xcb\xf0\xf7EV\xf6\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf8H\xfc\xe9\xaba\x1c}\x99
n#\xfa\u019a\u0508\xb9O\xe1\x89\x02\xa1\x12\x9d\t6r\x00\x00\u07d4\xf8O\t\n\xdf?\x8d\xb7\u1533P\xfb\xb7u\x00i\x9ff\xfd\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf8Q\xb0\x10\xf63\xc4\n\xf1\xa8\xf0js\ubeabe\az\xb5\x89\xee\x86D/\xcd\x06\xc0\x00\x00\u07d4\xf8X\x17\x1a\x04\xd3W\xa1;IA\xc1n~U\xdd\u0514\x13)\x89\x02F\xa5!\x8f*\x00\x00\x00\u07d4\xf8[\xab\x1c\xb3q\x0f\xc0_\xa1\x9f\xfa\xc2.gR\x1a\v\xa2\x1d\x89l\x955\u007f\xa6\xb3l\x00\x00\u07d4\xf8j>\xa8\a\x1fp\x95\xc7\u06ca\x0

5\xaePz\x89)\u06f8v\x89\x126\xef\xcb\u02f3@\x00\x00\u07d4\xf8pL\x16\xd2\xfd[\xa3\xa2\xc0\x1d
\x0e\xb2\x04\x84\xe6\xec\xfa1\t\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xf8p\x99_\xe1\xe5\"2\x1duC7\xa4\\\f\x9d{8\x95\x1c\x89\x01\x15\x8eF\t\x13\xd0\x0
0\x00\u07d4\xf8s\xe5ze\xc9;n\x18\xcbu\xf0\xdc\a}[\x893\xdc\\\x89\n\xad\xec\x98?\xcf\xf4\x00\x00\
u07d4\xf8ua\x9d\x8a#\xe4]\x89\x98\u0444\u0500\xc0t\x89p\x82*\x89\xd8\xd7&\xb7\x17z\x80\x00\
x00\u07d4\xf8{\xb0{(\x9d\xf70\x1eT\xc0\xef\xdaj,\xf2\x91\xe8\x92\x00\x89K\xe4\xe7&{j\xe0\x00\x0
0\u0794\xf8\x89\x00\xdbsyU\xb1Q\x9b\x1a}\x17\n\x18\x86L\xe5\x90\xeb\x88\xfc\x93c\x92\x80\x1c
\x00\x00\u07d4\xf8\x8bX\xdb7B\vFL\v\xe8\x8bE\xee+\x95)\x0f\x8c\xfa\x89\x02+\x1c\x8c\x12'\xa0\
x00\x00\u07d4\xf8\x96+u\xdb]$\xc7\xe8\xb7\xce\xf1\x06\x8c>g\u03bb0\xa5\x89\x0f-
\xc7\xd4\u007f\x15`\x00\x00\u07d4\xf8\xa0e\xf2\x87\xd9\x1dw\xcdbj\xf3\x8f\xfa\"\r\x9bU*+\x89g\x
8a\x93
b\xe4\x18\x00\x00\u07d4\xf8\xa4\x9c\xa29\f\x1fm\\\x0ebQ;\a\x95qt?|\u0189\xa2\xa1]\tQ\x9b\xe0\x
00\x00\u07d4\xf8\xa5\f\xee.h\x8c\xee\u3b24\u0522\x97%\xd4\a,\u0103\x89lk\x93[\x8b\xbd@\x00\
x00\u07d4\xf8\xacJ9\xb5<\x110x
\x97;D\x13e\xcf\xfeYof\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf8\xae\x85{g\xa4\xa2\x89:?\xbe|z\x8
7\xff\x1c\x01\u01a6\xe7\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xf8\xbf\x9c\x04\x87NZw\xf3\x
8fL8R~\x80\xc6v\xf7\xb8\x87\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf8\xc7\xf3J8\xb3\x18\x01\xda
C\x064w\xb1+'\xd0\xf2\x03\xff\x89\x1a\u04ba\xbao\xefH\x00\x00\u07d4\xf8\xca3l\x8e\x91\xbd
\xe3\x14\xc2\v-\xd4`\x8b\x9c\x8b\x94Y\x89-
\u071b\u0173,x\x00\x00\u07d4\xf8\xd1t$\xc7g\xbe\xa3\x12\x05s\x9a+W\xa7'r\x14\uef89\x02F\xdd
\xf9yvh\x00\x00\u07d4\xf8\xd5-
\xcc_\x96\xcc(\x00{>\u02f4\t\xf7\xe2*dl\xaa\x89\b\x16\x90\xe1\x81(H\x00\x00\u07d4\xf8\xdc\xe8g\
xf0\xa3\x9c[\xef\x9e\xeb\xa6\t\"\x9e\xfa\x02g\x8bl\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf8\xf2&\x1
4*B\x844\xab\x17\xa1\x86J%\x97\xf6J\xab/\x06\x89\tY\x8b/\xb2\xe9\xf2\x80\x00\u07d4\xf8\xf6d^\r
\xeedK=\xad\x81\xd5q\uf6ef\x84\x00!\xad\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf9\x01\xc0\x0f\xc1
\u06c8\xb6\x9cK\xc3%+\\\xa7\x0e\xa6\xee\\\xf6\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xf9=[\
xcb\x06D\xb0\xcc\xe5\xfc\u0763C\xf5\x16\x8f\xfa\xb2\x87}\x89\vb\a\xb6}&\xf9\x00\x00\u07d4\xf9
W\x0e\x92L\x95\u07bbpa6\x97\x92\xcf.\xfe\u00a8-
^\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xf9d
\x86\xb1\xfb\xaea\xa6\x80M\xbe_\xb1^\xc2\u04b57\xf4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf9d\
x88i\x85\x90\xdc;,UVB\xb8q4\x8d\xfa\x06z\u0549\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf9d\u
064d(\x170\xba5\xb2\xe3\xa3\x14yn{B\xfe\xdfg\x89S\xb0\x87`\x98\xd8\f\x00\x00\u07d4\xf9e\ri\x89
\xf1\x99\xab\x1c\xc4ycm\xed0\xf2A\x02\x1fe\x89.\x14\x1e\xa0\x81\xca\b\x00\x00\xe0\x94\xf9h\x8
3X$Y\x90\x8c\x82v'\xe8o(\xe6F\xf9\xc7\xfcz\x8a\x01\u0127\x877\xcd\u03f8\x00\x00\u07d4\xf9kL\
x00voSsj\x85t\xf8\"\xe6GL/!\xda-
\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xf9r\x9dH(,\x9e\x87\x16m^\xef-
\x01\xed\xa9\xdb\xf7\x88!\x89\x05k\x83\xdd\xc7(T\x80\x00\u07d4\xf9v~N\xcbJY\x80Ru\b\u05fe\xc
3\xd4^Ld\x9c\x13\x89g\x8a\x93
b\xe4\x18\x00\x00\xe0\x94\xf9x\xb0%\xb6B3U\\\xc3\xc1\x9a\xda\u007fA\x99\xc94\x8b\xf7\x8aT\x
b4\v\x1f\x85+\xda\x00\x00\x00\u07d4\xf9{V\xeb\u0577z\xbc\x9f\xba\u02eb\u0514\xb9\xd2\xc2!\xc
d\x03\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xf9\x81\x1f\xa1\x9d\xad\xbf\x02\x9f\x8b\xfeV\x9a\xdb
\x18\"\x8c\x80H\x1a\x89\n\u05ce\xbcZ\xc6

\x00\x00\u07d4\xf9\x82Ps\fLa\xc5\u007f\x12\x985\xf2h\b\x94yEB\xf3\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xf9\x894gr\x99^\xc1\x90o\xaf\xfe\xba*\u007f\xe7\u079ck\xab\x8a\x01je\x02\xf1Z\x1eT\x00\x00\u07d4\xf9\x98\xca4\x11s\nl\xd1\x0etU\xb0A\x0f\xb0\xf6\xd3\xff\x80\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf9\x9a\xeeDKW\x83\xc0\x93\xcf\xff\xd1\xc4c,\xf9<o\x05\f\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xf9\x9e\xee\xce9\xfa~\xf5\am\x85Pa8@\ty,\xf2\xe0\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf9\xa5\x9c<\xc5\xff\xac\xbc\xb6{\xe0\xfcrV\xf6L\x9b\x12|\xb4\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xf9\xa9K\xd5a\x98\xda$^\xd0\x1d\x1ed0\xb2K'\b\xdc\xc0\x89(\xa7z\xfd\xa8~\xe5\x00\x00\u07d4\xf9\xb3x%\xf00s\xd3\x1e$\x93x\xc3\fy\\3\xf8:\xf2\x89\n\u066a\xbf\x8c\x9b\xfc\x00\x00\u07d4\xf9\xb6\x17\xf7R\xed\xec\xae>\x90\x9f\xbb\x91\x1d/\x81\x92\xf8B\t\x89\x90\xf54`\x8ar\x88\x00\x00\u07d4\xf9\xbf\xb5\x9dS\x8a\xfcHt\xd4\xf5\x94\x1b\b\xc9s\x0e8\xe2K\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xf9\xdd#\x90\b\x18/\xb5\x19\xfb0\xee\xdd
\x93\xfe\xd1c\x9b\xe8\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xf9\u07ba\xec\xb5\xf39\xbe\xeaH\x94\xe5
K\xfa4\r\x06\u007f%\x89ZB\x84Fs\xb1d\x00\x00\xe0\x94\xf9\xe3tG@IA!\x97\xb2\u2bbc\x00\x1dn0\u024c`\x8a\x01\xc4y\xbbCI\xc0\xee\x00\x00\u07d4\xf9\xe7\"/\xaa\xf0\xf4\xda@\xc1\u0124\x0607:\t\xbe\u05f6\x89\x9bO\u0730\x94V$\x00\x00\u07d4\xf9\xec\xe0\"\xbc\xcd,\x924i\x11\xe7\x9d\xd5\x03\x03\xc0\x1e\x01\x88\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xfa\x00\xc3v\xe8\x9c\x05\u81c1z\x9d\xd0t\x8d\x96\xf3A\xaa\x89\x89\x10M\r\x00\u04b7\xf6\x00\x00\u07d4\xfa\f\x1a\x98\x8c\x8a\x17\xad5(\xeb(\xb3@\x9d\xaaX\"_&\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xfa\x10_\x1a\x11\xb6\xe4\xb1\xf5`\x12\xa2y\"\xe2\xac-
\xa4\x81/\x8a\x02\x05\xb4\u07e1\xeetx\x00\x00\u07d4\xfa\x14/\xe4~\u0697\xe6P;8k\x18\xa2\xbe\xdds\u0335\xb1\x89.\x15:\xd8\x15H\x10\x00\x00\u07d4\xfa\x14\xb5f#J\xbe\xe70B\xc3\x1d!qq\x82\u02e1J\xa1\x89\x11\xc7\xea\x16.x
\x00\x00\u07d4\xfa\x19\xd6\xf7\xa5\x0fO\a\x98\x93\xd1g\xbf\x14\xe2\x1d\x00s\u0456\x89\x1c\xbb:?\xf0\x8d\b\x00\x00\u07d4\xfa\x1f\x19q\xa7u\xc3PO\xefPy\xf6@\xc2\u013c\xe7\xac\x05\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfa'\x9b\xfd\x87g\xf9V\xbf\u007f\xa0\xbdV`\x16\x8d\xa7V\x86\xbd\x89\x90\xf54`\x8ar\x88\x00\x00\xe0\x94\xfa'\xccI\xd0\vl\x98s6\xa8u\xae9\xdaX\xfb\x04\x1b.\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\xfa(2\x99`=\x87X\xe8\u02b0\x82\x12],\x8f}DT)\x8a\x01[\xca\xcb\x1e\x05\x01\xae\x80\x00\u07d4\xfa+\xbc\xa1]?\u37ca2\x8e\x91\xf9\r\xa1Oz\xc6%=\x89\n\u05ce\xbcZ\xc6
\x00\x00\xe0\x94\xfa/\u049d\x03\xfe\xe9\xa0x\x93\xdf:&\x9fV\xb7/.\x1ed\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xfa3U2\x85\xa9sq\x9a\r_\x95o\xf8a\xb2\u061e\xd3\x04\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xfa:\fK\x90?n\xa5.\xa7\xab{\x88c\xb6\xa6\x16\xadfP\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xfa:\x1a\xa4H\x8b5\x1a\xa7V\f\xf5\xeec\n/\xd4\\2\"\x89/\xa4~j\xa74\r\x00\x00\u07d4\xfaA\tq\xad\"\x9c06\xf4\x1a\u03c5/*\u0259(\x19P\x89\u0633\x11\xa8\xdd\xfa|\x00\x00\u07d4\xfaD\xa8U\xe4\x04\xc8m\f\xa8\xef3$%\x1d\xfb4\x9cS\x9e\x89T\"S\xa1&\xce@\x00\x00\xe0\x94\xfaR\x01\xfe\x13B\xaf\x110{\x91B\xa0A$<\xa9./\t\x8a
8\x11j:\xc0C\x98\x00\x00\xe0\x94\xfa`\x86\x8a\xaf\xd4\xffL\\W\x91K\x8e\u054bBWs\u07e9\x8a\x01\xcf\xe5\xc8\b\xf3\x9f\xbc\x00\x00\u07d4\xfag\xb6{O7\xa0\x15\t\x15\x11\x0e\xde\a;\x05\xb8S\xbd\xa2\x89#\x19\xba\x94sq\xad\x00\x00\u07d4\xfah\xe0\xcb>\xdfQ\xf0\xa6\xf2\x11\u0272\xcb^\a<\x9b\xff\xe6\x89\x0f\xc969(\x01\xc0\x00\x00\xe0\x94\xfaj7\xf0\x18\xe9yg\x93\u007f\xc5\xe8a{\xa1\u05c6\xdd_w\x8a\x04<0\xfb\b\x84\xa9l\x00\x00\u07d4\xfav\x06C[5l\xee%{\xd2\xfc\xd3\xd9\xea\xc

b<\xd1\xc4\xe1\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xfaz\xdff\v\x8d\x99\xce\x15\x93=|_\a/<\xbe\xb9\x9d3\x8a\x01@a\xb9\xd7z^
\x98\x00\x00\u07d4\xfa\x86\xca'\xbf(T\u0648p\x83\u007f\xb6\xf6\xdf\xe4\xbfdS\xfc\x89\x11u~\x85
%\xcf\x14\x80\x00\u07d4\xfa\x8c\xf4\xe6'i\x8c]W\x88\xab\xb7\x88\x04\x17\xe7P#\x13\x99\x89\xe
6\x1a6\x96\xee\xf6\x10\x00\x00\u07d4\xfa\x8e;\x1f\x13C9\x00s}\xaa\xf1\xf6)\x9cH\x87\xf8[_\x89&\
u009eG\u0104L\x00\x00\u07d4\xfa\x9e\xc8\xef\xe0\x86\x86\xfaX\xc1\x813Xr\xbai\x85`\ucac9lj\xc
cg\u05f1\xd4\x00\x00\u07d4\xfa\xad\x90]\x84|{#A\x8a\xee\xcb\xe3\xad\u06cd\xd3\xf8\x92J\x89j\x
cb=\xf2~\x1f\x88\x00\x00\u07d4\xfa\xae\xba\x8f\xc0\xbb\xdaU<\xa7.0\xef=s.&\xe8
A\x89H\x8d(*\xaf\xc9\xf6\x80\x00\u07d4\xfa\xb4\x87P\r\xf2\x0f\xb8>\xbe\xd9\x16y\x1dV\x17r\xad\
xbe\xbf\x89lkLM\xa6\u077e\x00\x00\u07d4\xfa\xc5\u0294u\x80x\xfb\xfc\xcd\x19\xdb5X\xda~\u8827
h\x897(\xa6+\r\xcf\xf6\x00\x00\u07d4\xfa\xd9j\xb6\xacv\x8a\xd5\t\x94R\xacGw\xbd\x1aG\xed\u0
10f\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xfa\xe7g\x19\xd9~\xacA\x87\x04(\xe9@'\x9d\x97\xddW\xb2\xf6\x8a\x14\u0
6f2\x19\\\xa2(\x90\x00\x00\u07d4\xfa\u8053pG\x89Zf\f\xf2)v\x0f'\xe6h(\xd6C\x89\t\xdd\xc1\xe3\xb
9\x01\x18\x00\x00\u07d4\xfa\xe9,\x13p\xe9\u115a]\xf8;V\xd0\xf5\x86\xaa;@L\x89\x05\u0174\xf3\x
d8C\x98\x00\x00\xe0\x94\xfa\xf5\xf0\xb7\xb6\xd5X\xf5\t\r\x9e\xa1\xfb-
B%\x9cX`x\x8a\x01Z\xff\xb8B\fkd\x00\x00\xe0\x94\xfb\x12o\x0e\xc7i\xf4\x9d\xce\xfc\xa2\xf2\x00(d
QX0\x84\xb8\x8a\x01\x0f\xcb\xc25\x03\x96\xbf\x00\x00\xe0\x94\xfb\x13^\xb1Z\x8b\xacr\xb6\x99\x
154*`\xbb\xc0k~\a|\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xfb\"<\x1e\"\xea\xc1&\x9b2\xee\x15j
S\x85\x92.\xd3o\xb8\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfb7\xcfkO\x81\xa9\xe2\"\xfb\xa2.\x9b\x
d2KP\x98\xb73\u03c9\x02\x1auJm\xc5(\x00\x00\u07d4\xfb8`\xf4\x12\x1cC.\xbd\xc8\xecj\x031\xb1
\xb7\ty.\x90\x89
\x8c9J\xf1\u0208\x00\x00\u07d4\xfb9\x18\x9a\xf8v\xe7b\xc7\x1dl>t\x18\x93\xdf\"l\xed\u0589\xd8\x
d7&\xb7\x17z\x80\x00\x00\u07d4\xfb:\v\rkjq\x8fo\xc0)*\x82]\xc9$z\x90\xa5\u0409\n\xd6\xdd\x19\x
9e\x97[\x00\x00\xe0\x94\xfb?\xa1\xac\b\xab\xa9\xcc;\xf0\xfe\x9dH8
h\x8fe\xb4\x10\x8a\x06ZM\xa2]0\x16\xc0\x00\x00\u07d4\xfb?\xe0\x9b\xb86\x86\x15)\xd7Q\x8d\xa
2v5\xf58PV\x15\x89K\xe3\x92\x16\xfd\xa0p\x00\x00\xe0\x94\xfbQ%\xbf\x0f^\xb0\xb6\xf0
\xe5k\xfc/\xdf=@,\t~\x8a\x01@a\xb9\xd7z^\x98\x00\x00\u07d4\xfbU\x18qL\xef\xc3m\x04\x86]\xe5\
x91^\xf0\xffG\xdf\xe7C\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfb_\xfa\xa0\xf7aW&5x\x91GX\x18\x9
3\x9d
7\u03d6\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xfbh\\\x15\xe49\x96^\xf6&\xbf\r\x83L\u0468\x
9f+V\x95\x89\u0556{\xe4\xfc?\x10\x00\x00\u07d4\xfbtK\x95\x1d\tK1\x02b\xc8\xf9\x86\xc8`\u07da\
xb1\xdee\x89\x02\xd1\xc5\x15\xf1\xcbJ\x80\x00\u07d4\xfby\xab\u06d2\\U\xb9\xf9\x8e\xfe\xefd\xcf\
xc9\xeba\xf5\x1b\xb1\x89a@\xc0V\xfb\n\xc8\x00\x00\u07d4\xfb\x81\x13\xf9M\x91s\xee\xfdZ0s\xf5
\x16\x80:\x10\xb2\x86\xae\x89\x04V9\x18$O@\x00\x00\u07d4\xfb\x84,\xa2\xc5\xef\x139\x17\xa2
6\xa0\u052c@i\x01\x10\xb08\x89\x10\x96\x9ab\xbe\x15\x88\x00\x00\u07d4\xfb\x91\xfb\x1aiUS\xf
0\u018e!'m\xec\xf0\xb89\t\xb8m\x89\x05l\x006\x17\xafx\x00\x00\u07d4\xfb\x94s\xcfw\x125\n\x1f\x
a09Rs\xfc\x80V\aR\xe4\xfb\x89\x06\xaf!\x98\xba\x85\xaa\x00\x00\xe0\x94\xfb\x94\x9cd\u007f\xdc
\xfd%\x14\xc7\u054e1\xf2\x8aS-
\x8cX3\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xfb\xa5HmS\xc6\xe2@IBA\xab\xf8~C\xc7`\rA:\x8
9k\xbfaIIH4\x00\x00\u07d4\xfb\xb1a\xfe\x87_\t)\nK&+\xc6\x01\x10\x84\x8f\r\"&\x89lk\x93[\x8b\xbd
@\x00\x00\u07d4\xfb\xbb\xeb\u03fe#^W\xdd#\x06\xad\x1a\x9e\u0141\xc7\xf9\xf4\x8f\x89\x02+\x1

c\x8c\x12'\xa0\x00\x00\xe0\x94\xfb\xc0\x1d\xb5NG\xcd\xc3\xc48iJ\xb7\x17\xa8V\xc2?\xe6\xe9\x8a\x01\xcaqP\xab\x17OG\x00\x00\xe0\x94\xfb\xcf\xccJ{\x0f&\xcf&\xe9\xf33!2\xe2\xfcj#\af\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xfb\xe7\x16\"\xbc\xbd1\xc1\xa3iv\xe7\xe5\xf6p\xc0\u007f\xfe\x16\u0789\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xfb\xed\xe3,4\x9f3\x00\xefL\xd3;M\xe7\xdc\x18\xe4C\xd3&\x89\xabM\xcf9\x9a:`\x00\x00\u07d4\xfb\xf2\x04\xc8\x13\xf86\xd89b\u01c7\fx\b\xca4\u007f\xd3>\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xfb\xf7Y3\xe0\x1bu\xb1T\xef\x06i\ak\xe8\u007fb\xdf\xfa\xe1\x8a\x10\x84cr\xf2I\xd4\xc0\x00\x00\u07d4\xfc\x00\x96\xb2\x1e\x95\xac\xb8\xd6\x19\xd1v\xa4\xa1\xd8\xd5)\xba\xdb\xef\x89\x14\xd9i;\xcb\xec\x02\x80\x00\xe0\x94\xfc\x00\xa4
\xa3a\a\xdf\xd5\xf4\x95\x12\x8a_\u5af2\xdb\x0f4\x8a\x01C\x17\x9d\x86\x91\x10
\x00\x00\xe0\x94\xfc\x01\x8ai\n\xd6tm\xbe:\u03d7\x12\xdd\xcaR\xb6%\x009\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xfc\x02s@3\xe5\u007fpQ~\n\xfc~\xe6$a\xf0o\xad\x8e\x89\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\xfc\x0e\xe6\xf7\u00b3qJ\xe9\x91lEVf\x05\xb6V\xf3$A\x89_h\xe8\x13\x1e\u03c0\x00\x00\u07d4\xfc\x10\xb7\xa6{2h\xd53\x1b\xfbj\x14\xde\xf5\xeaJ\x16,\xa3\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xfc\x15\u02d9\xa8\xd1\x03\v\x12w\n\xdd\x03:y\xee\r\f\x90\x8c\x89\x12\xfa\x00\xbdR\xe6$\x00\x00\u07d4\xfc)R\xb4\u011f\xed\u043c\x05(\xa3\bI^mj\x1cq\u0589lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xfc,\x1f\x88\x96\x1d\x01\x9c>\x9e\xa30\t\x15.\x06\x93\xfb\xf8\x8a\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\xe0\x94\xfc6\x11\x05\u0750\xf9\xed\xe5fI\x9di\xe9\x13\x03\x95\xf1*\u020aS\xa4\xfe/
N\x80\xe0\x00\x00\u07d4\xfc7/\xf6\x92|\xb3\x96\xd9\xcf)\x805\x00\x11\r\xa62\xbcR\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfc9\xbeA\tK\x19\x97\xd2\x16\x9e\x82d\xc2\u00fa\xa6\u025b\u0109lk\x93[\x8b\xbd@\x00\x00\u07d4\xfc=\"k\xb3jX\xf5&V\x88W\xb0\xbb\x12\xd1\t\xec\x93\x01\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfcC\x82\x9a\u01c7\xff\x88\xaa\xf1\x83\xba5*\xad\xbfZ\x15\xb1\x93\x89\u05ac\n+\x05R\xe0\x00\x00\u07d4\xfcI\xc1C\x9aA\u05b3\xcf&\xbbg\xe06R$\xe5\xe3\x8f_\x8966\u05ef^\u024e\x00\x00\u07d4\xfcU\x00\x82Q\x05\xcfq*1\x8a^\x9c;\xfci\u021d\f\x12\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\xe0\x94\xfcf\xfa\xba'\u007fK]\xe6J\xd4^\xb1\x9c1\xe0\f\xed>\u054a\x011\xbe\xb9%\xff\xd3 \x00\x00\xe0\x94\xfc~\"\xa5\x03\xecZ\xbe\x9b\b\xc5\v\xd1I\x99\xf5
\xfaH\x84\x8a\x01ZG}\xfb\xe1\xea\x14\x80\x00\u07d4\xfc\x82\x15\xa0\xa6\x99\x13\xf6*C\xbf\x1c\x85\x90\xb9\xdd\xcd\r\x8d\u06c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xfc\x98\x9c\xb4\x87\xbf\x1a}\x17\xe4\xc1\xb7\u0137\xaa\xfd\xdak\n\x8d\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\xe0\x94\xfc\x9b4td\xb2\xf9\x92\x9d\x80~\x03\x9d\xaeH\xd3\u064d\xe3y\x8a\x02\xf6\xf1\a\x80\xd2,\xc0\x00\x00\u07d4\xfc\xa4;\xbc#\xa0\xd3!\xba\x9eF\xb9)s\\\xe7\xd8\xef\f\x18\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xfc\xa7>\xff\x87q\xc0\x10;\xa3\xcc\x1a\x9c%\x94H\xc7*\xbf\v\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xfc\xad\xa3\x00(?k\xcc\x13J\x91Eg`\xb0\xd7}\xe4\x10\xe0\x89lk\x93[\x8b\xbd@\x00\x00\xe0\x94\xfc\xbc\\q\xac\xe7\x97AE\v\x01,\xf6\xb8\xd3\xf1}\xb6\x8ap\x8a\x02\x05\xb4\u07e1\xeetx\x00\x00\u07d4\xfc\xbd\x85\xfe\xeajuO\xcf4ID\x9e7\xff\x97\x84\xf7w<\x89\xa7J\xdai\xab\xd7x\x00\x00\xe0\x94\xfc\xc9\u0524&.z\x02z\xb7Q\x91\x10\xd8\x02\u0115\xce\xea9\x8a\x01YQ\x82\"K&H\x00\x00\xe0\x94\xfc\xcd\r\x1e\xce\xe2z\xdd\xea\x95\xf6\x85z\xee\xc8\u01e0K(\xee\x8a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\xe0\x94\xfc\u0434\x82|\xd2\b\xff\xbf^u\x9d\xba\x8c<\xc6\x1d\x8c,<\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xfc\xe0\x89c\\\xe9z\xba\xc0kD\x81\x9b\xe5\xbb\n>.\v7\x89\x05\x03\x92\nv0\xa7\x80\x00\u07d4\xfc\xf1\x99\xf8\xb8T\"/\x18.N\x1d\t\x9dN2>*\xae\x01\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xfc\xfc:P\x04\xd6xa?\v6\xa6B&\x9a\u007f7\x1c?j\x8965\u026

d\xc5\u07a0\x00\x00\u07d4\xfd\x19\x1a5\x15}x\x13s\xfbA\x1b\xf9\xf2R\x90\x04|^\xef\x8965\u026
d\xc5\u07a0\x00\x00\u07d4\xfd\x1f\xaa4{\x0f\u0300L-
\xa8l6\xd5\xf1\u044bp\x87\xbb\x89\x02\xd6\xeb$z\x96\xf6\x00\x00\u07d4\xfd\x1f\xb5\xa8\x9a\x89\
xa7!\xb8yph\xfb\xc4\u007f>\x9dR\xe1I\x89\f\u0435\x83\u007f\xc6X\x00\x00\u07d4\xfd
OOJ\xba%%\xbar\x8a\xfd\xf7\x87\x92\xcb\u07b75\xae\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfd'W
\xcc5Q\xa0\x95\x87\x8d\x97\x87V\x15\xfe\fj2\xaa\x8a\x89
m\xb1R\x99\xbe\xac\x00\x00\u07d4\xfd(r\u045eW\x85<\xfa\x16\xef\xfe\x93\u0431\xd4{O\x93\xfb\
x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xfd))'\x1e\x9d
\x95\xa2dv~{\r\xf5.\xa0\xd1\xd4\x00\x89\xa2\xa1\xeb%\x1bZ\xe4\x00\x00\u07d4\xfd7z8Rr\x90\f\xb
46\xa3\xbbyb\xcd\xff\xe9?]\xad\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfd@$+\xb3Jp\x85^\xf0\xfd\x
90\xf3\x80-\xec!6\xb3'\x89h\xa8u\a>)$\x00\x00\xe0\x94\xfdE,9i\xec\xe3\x80\x1cT
\xf1\xcd\u02a1\xc7\x1e\xd2=\x8a\x15-
\x02\xc7\xe1J\xf6\x80\x00\x00\u07d4\xfdKU\x1fo\xdb\u0366\xc5\x11\xb5\xbb7\"P\xa6\xb7\x83\xe5
4\x89\x01\x1d\xe1\xe6\xdbE\f\x00\x00\u07d4\xfdK\x98\x95X\xae\x11\xbe\f;6\xe2\xd6\xf2\xa5J\x93
C\xca.\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfdM\xe8\xe3t\x8a(\x9c\xf7\xd0`Q}\x9d88\x8d\xb0\x1f\
xb8\x89\r\x8drkqw\xa8\x00\x00\u07d4\xfdZc\x15\u007f\x91O\u04d8\uac5c\x13}\xd9U\v\xb7q\\\x89
\x05k\xc7^-
c\x10\x00\x00\u07d4\xfd`\u04b5\xaf=5\xf7\xaa\xf0\u00d3\x05.y\xc4\xd8#\u0645\x89\x03\x0e\xb5\r
.\x14\b\x00\x00\u07d4\xfdhm\xe5?\xa9\u007f\x99c\x9e%hT\x97 \xbcX\x8c\x9e\xfc\x89j\xc5\xc6-
\x94\x86\a\x00\x00\u07d4\xfd~\u078fR@\xa0eA\xebi\x9dx,/\x9a\xfb!p\xf6\x89Hz\x9a0E9D\x00\x00
\u07d4\xfd\x81+\u019f\xb1p\xefW\xe22~\x80\xaf\xfd\x14\xf8\xe4\xb6\u0489lk\x93[\x8b\xbd@\x00\
x00\u07d4\xfd\x88\xd1\x14\"\x0f\b\x1c\xb3\xd5\xe1[\xe8\x15*\xb0sfWj\x89\x10CV\x1a\x88)0\x00\x
00\u07d4\xfd\x91\x856\xa8\xef\xa6\xf6\xce\xfe\x1f\xa1\x159\x95\xfe\xf5\xe3=;\x89\x1b\x1a\xe4\xd
6\xe2\xefP\x00\x00\u07d4\xfd\x92\x0fr&\x82\xaf\xb5\xafE\x1b\x05D\xd4\xf4\x1b;\x9dWB\x89~R\x0
5j\x12?<\x00\x00\u07d4\xfd\x95y\xf1\x19\xbb\xc8\x19\xa0+a\u3348\x03\xc9B\xf2M2\x89\x05\xb9~
\x90\x81\xd9@\x00\x00\u07d4\xfd\xa0\xce\x153\a\a\xf1\v\xce2\x01\x17-
\x18\xb9\xdd\xeat\x89\x02\xd0A\xd7\x05\xa2\xc6\x00\x00\xe0\x94\xfd\xa3\x04(\x19\xaf>f)\x00\xe1
\xb9+CX\xed\xa6\xe9%\x90\x8a\x19\a\xa2\x84\u054fc\xe0\x00\x00\u07d4\xfd\xa6\x81\x0e\xa5\xa
c\x98]o\xfb\xf1\xc5\x11\xf1\xc1B\xed\xcf\xdd\xf7\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xfd\xb
39D\xf26\x06\x15\xe5\xbe#\x95w\u0221\x9b\xa5-\x98\x87\x89
\x9d\x92/RY\xc5\x00\x00\u07d4\xfd\xbaSY\xf7\xec;\xc7p\xacI\x97]\x84N\xc9qbV\xf1\x8965\u026d\
xc5\u07a0\x00\x00\xe0\x94\xfd\xc4\xd4vZ\x94/[\xf9i1\xa9\xe8\xccz\xb8\xb7W\xffL\x8a\x12lG\x8a\x
0e>\xa8`\x00\x00\xe0\x94\xfd\xcd]\x80\xb1\x05\x89zW\xab\xc4xev\x8b)\x00RB\x95\x8a\x01Z\xf1\
u05cbX\xc4\x00\x00\x00\u0794\xfd\xd1\x19_y}O5q}\x15\xe6\xf9\x81\n\x9a?\xf5T`\x88\xfc\x93c\x9
2\x80\x1c\x00\x00\u07d4\xfd\xd5\x02\xa7N\x81;\u03e3U\xce\xda<\x17ojhq\xaf\u007f\x89\x15\xaf\
x1dx\xb5\x8c@\x00\x00\u07d4\xfd\u357c\vm\\\xbbL\x1d\x8f\xea>\vK\xffc^\x9d\xb7\x89lk\x93[\x8b\
xbd@\x00\x00\u07d4\xfd\xea\xac*\xcf\x1d\x13\x8e\x19\xf2\xfc?\x9f\xb7E\x92\xe3\ud04a\x89$=M\
x18\"\x9c\xa2\x00\x00\u07d4\xfd\xec\xc8-
\xdf\xc5a\x92\xe2oV<=h\xcbTJ\x96\xbf\xed\x89\x17\xda:\x04\u01f3\xe0\x00\x00\u07d4\xfd\xf4#C\
x01\x9b\v\fk\xf2`\xb1s\xaf\xab~E\xb9\xd6!\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xfd\xf4I\xf1\b\xc6\xf
bOZ+\b\x1e\xed~E\u645eM%\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfd\xfda4\xc0J\x8a\xb7\xeb\x1
6\xf0\x06C\xf8\xfe\xd7\u06aa\ucc89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xfe\x00\xbfC\x99\x11\

xa5S\x98-
\xb68\x03\x92E\xbc\xf02\xdb\u0709\x15[\xd90\u007f\x9f\xe8\x00\x00\u07d4\xfe\x01n\xc1~\xc5\xf1
\x0e;\xb9\x8f\xf4\xa1\xed\xa0E\x15v\x82\xab\x89\x14_T\x02\xe7\xb2\xe6\x00\x00\u07d4\xfe\x0e0\
xe2\x14)\rt=\xd3\x0e\xb0\x82\xf1\xf0\xa5\"Z\xdea\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xfe!\v\x8f\x04\xdcmOv!j\xcf\xcb\u055b\xa8;\xe9\xb60\x89\x01\x15\x8eF\t\x13\xd0\
x00\x00\u07d4\xfe\"\xa0\xb3\x88f\x8d\x1a\xe2d>w\x1d\xac\xf3\x8aCB#\u0309\xd8\xdb^\xbd{&8\x0
0\x00\u07d4\xfe6&\x88\x84_\xa2D\u0300~K\x110\xeb7A\xa8\x05\x1e\x8965\u026d\xc5\u07a0\x0
0\x00\u07d4\xfe8'\xd5v0\u03c7a\xd5\x12y{\v\x85\x8eG\x8b\xbd\x12\x89\x01\x15\x8eF\t\x13\xd0\x
00\x00\u07d4\xfeA\x8bB\x1a\x9cm76\x02y\x04u\xd20>\x11\xa7Y0\x897\b\xba\xed=h\x90\x00\x00\
u07d4\xfeBI\x12yP\xe2\xf8\x96\xec\x0e~.=\x05Z\xab\x10U\x0f\x89$=M\x18\"\x9c\xa2\x00\x00\xe0
\x94\xfeM\x84\x03!o\xd5qW+\xf1\xbd\xb0\x1d\x00W\x89x\u0588\x8a\x02\x15\xf85\xbcv\x9d\xa8\x
00\x00\u07d4\xfeS\xb9I\x89\u0619d\xda
aS\x95&\xbb\xe9y\xdd.\xa9\x89h\xa8u\a>)$\x00\x00\u07d4\xfeT\x9b\xbf\xe6G@\x18\x98\x92\x93
%8\u06afF\u04b6\x1dO\x89\x02+\x1c\x8c\x12'\xa0\x00\x00\xe0\x94\xfea]\x97\\\b\x87\xe0\xc9\x11
>\xc7)\x84
\xa7\x93\xaf\x8b\x96\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xfee\xc4\x18\x8dy\"Wi\td
D\xfd\xc5#\x95V\x01e\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xfei\u007f\xf2,\xa5G\xbf\xc9^3\x
d9`\xda`\\gc\xf3[\x89G\xd4\x11\x9f\xd9`\x94\x00\x00\u07d4\xfej\x89[y\\\xb4\xbf\x85\x90=<\xe0\x9c
Z\xa49S\u04ff\x89\xb8Pz\x82\a(
\x00\x00\u07d4\xfeo_B\xb6\x19;\x1a\xd1b\x06\u4bf5#\x9dM}\xb4^\x89]\u0212\xaa\x111\xc8\x00\x
00\u07d4\xfep\x11\xb6\x98\xbf3q\x13-
tE\xb1\x9e\xb5\xb0\x945j\xee\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfe\x80\xe9#-
\xea\xff\x19\xba\xf9\x98i\x88:K\xdf\x00\x04\xe5<\x89.b\xf2\ni\xbe@\x00\x00\u07d4\xfe\x8en6eW\r\
xffz\x1b\xdaiz\xa5\x89\xc0\xb4\xe9\x02J\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfe\x8f\x1f\u072b\u0
07f\xbe\u0266\xa3\xfc\xc5\aa\x96\x00P\\6\xa3\x89\x01\x11du\x9f\xfb2\x00\x00\u07d4\xfe\x91\xec\
xcf+\xd5f\xaf\xa1\x16\x96\xc5\x04\x9f\xa8Lic\nR\x89i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xfe\x96\x
c4\xcd8\x15b@\x1a\xa3*\x86\xe6[\x9dR\xfa\x8a\xee'\x89\x8f\x1d\\\x1c\xae7@\x00\x00\u07d4\xfe\
x98\xc6d\xc3\xe4G\xa9^i\xbdX!q\xb7\x17n\xa2\xa6\x85\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d
4\xfe\x9a\xd1.\xf0]m\x90&\x1f\x96\xc84\n\x03\x81\x97M\xf4w\x89lk\x93[\x8b\xbd@\x00\x00\u07d4
\xfe\x9c\x0f\xff\xef\xb8\x03\b\x12V\xc0\xcfMfY\xe6\xd3>\xb4\xfb\x89R\xd5B\x80O\x1c\xe0\x00\x00
\u07d4\xfe\x9c\xfc;\xb2\x93\u0772\x85\xe6%\xf3X/t\xa6\xb0\xa5\xa6\u0349j\xcb=\xf2~\x1f\x88\x00
\x00\xe0\x94\xfe\x9e\x11\x97\u05d7JvH\xdc\u01e01\x12\xa8\x8e\xdb\xc9\x04]\x8a\x01\n\xfc\x1a\x
de;N\xd4\x00\x00\xe0\x94\xfe\xac\xa2\xactbK\xf3H\xda\u0258QC\xcf\xd6R\xa4\xbeU\x8a\x05\x89
\u007f\u02f0)\x14\b\x80\x00\u07d4\xfe\xad\x18\x03\xe5\xe77\xa6\x8e\x18G-
\x9a\xc7\x15\xf0\x99L\u00be\x89\x1b\x1a\xe4\xd6\xe2\xefP\x00\x00\u07d4\xfe\xb8\xb8\xe2\xafqj\
xe4\x1f\xc7\xc0K\xcf)T\x01VF\x1ek\x89TQt\xa5(\xa7z\x00\x00\u07d4\xfe\xb9-
0\xbf\x01\xff\x9a\x19\x01flUsS+\xfa\a\xee\xec\x8965\u026d\xc5\u07a0\x00\x00\u07d4\xfe\xbc1s\x
bc\x90r\x13cT\x00+{O\xb3\xbf\xc5?\"\xf1\x89\x14\x0e\xc8\x0f\xa7\xee\x88\x00\x00\u07d4\xfe\xbd
H\xd0\xff\xdb\xd5el\xd5\xe6\x866:a\x14R(\xf2y\x89\x97\xc9\xceL\xf6\xd5\xc0\x00\x00\u07d4\xfe\x
bd\x9f\x81\xcfx\xbd_\xb6\u0139\xa2K\xd4\x14\xbb\x9b\xfaLN\x89k\xe1\x0f\xb8\xedn\x13\x80\x00\
u07d4\xfe\xc0o\xe2{D\u01c4\xb29n\xc9/{\x92:\xd1~\x90w\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xfe
\xc1NT\x85\xde+>\xef^t\xc4aF\u06ceEN\x035\x89\t\xb4\x1f\xbf\x9e\n\xec\x00\x00\u07d4\xfe\xd8G

m\x10\u0544\xb3\x8b\xfag7`\x0e\xf1\x9d5\xc4\x1e\u0609b\xa9\x92\xe5:\n\xf0\x00\x00\u07d4\xfe\x
ef;n\xab\xc9J\xff\xd31\f\x1cM\x0ee7^\x13\x11\x19\x89\x01\x15\x8eF\t\x13\xd0\x00\x00\u07d4\xfe\
xf0\x9dp$?9\xed\x8c\xd8\x00\xbf\x96QG\x9e\x8fJ\xca<\x89\n\u05ce\xbcZ\xc6
\x00\x00\u07d4\xfe\xf3\xb3\u07ad\x1ai&\u051a\xa3+\x12\xc2*\xf5M\x9f\xf9\x85\x8965\u026d\xc5\
u07a0\x00\x00\u07d4\xff\v|\xb7\x1d\xa9\xd4\xc1\xean\xcc(\xeb\xdaPLc\xf8/\u04498\x8a\x88]\xf2\x
fcl\x00\x00\u07d4\xff\f<w\x98\xe8s=\xd2f\x81R\x89\x1b\xab\x80\xa8\xbe\x95\\\x89\x04YF\xb0\xf9\
xe9\xd6\x00\x00\u07d4\xff\f\xb0lB\xe3\u0609H\xe4[\u05f0\xd4\u246e\xfe\xeaQ\x89g\x8a\x93
b\xe4\x18\x00\x00\u07d4\xff\f\xc8\xda\xc8$\xfa$\xfc<\xaa!i\xe6\xe0W\xcfc\x8a\u0589\xd8\xd7&\xb
7\x17z\x80\x00\x00\u07d4\xff\x0e/\xec0B\aF~\x1e3\a\xf6L\xbf0\xaf\x8f\xd9\u0349lk\x93[\x8b\xbd
@\x00\x00\u07d4\xff\x12\x8fK5[\xe1\xdcJo\x94\xfaQ\r\u007f\x15\xd5<*\xff\x89\x93s\x954\u0486\x
80\x00\x00\u07d4\xff\x12\u474e\x06\xaa
\xf8\x86)<\v\x98\xed~\xffx\x88\x05\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xff
s\b\xce\xd28\xa6\xc0\x1a\xd0!<\xa9\xebDe\xd4%\x90\x89lj\xccg\u05f1\xd4\x00\x00\u07d4\xff&\x1
3\x830'M\xf4\xe0\xa3\b\x1em\xf7\u0758>\xc6\u73c9lk\x93[\x8b\xbd@\x00\x00\u07d4\xff'&)AH\xb8
lx\xa97$\x97\xe4Y\x89\x8e\xd3\xfe\xe3\x89j\xcb=\xf2~\x1f\x88\x00\x00\u07d4\xff=\xedz@\u04ef\xf
0\u05e8\xc4_\xa6\x13j\xa0C=\xb4W\x89lh\xcc\u041b\x02,\x00\x00\u07d4\xff>\xeeW\xc3Mm\xae\x
97\r\x8b1\x11\x17\xc55\x86\xcd5\x02\x89\\(=A\x03\x94\x10\x00\x00\u07d4\xff>\xf6\xba\x15\x1c!\x
b5\x99\x86\xaed\xf6\xe8\"\x8b\u0262\xc73\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xffA\xd9\xe1\xb4\
xef\xfe\x18\u0630\xd1\xf6?\xc4%_\xb4\xe0l=\x89Hz\x9a0E9D\x00\x00\u07d4\xffE\xcb4\xc9(6M\x9
c\xc9\u063b\x0074ta\x8f\x06\xf3\x89\x05k\xc7^-
c\x10\x00\x00\xe0\x94\xffI\xa7u\x81N\xc0\x00Q\xa7\x95\xa8u\xde$Y.\xa4\x00\u050a*Z\x05\x8f\u0
095\xed\x00\x00\x00\u07d4\xffJ@\x8fP\xe9\xe7!F\xa2\x8c\xe4\xfc\x8d\x90'\x1f\x11n\x84\x89j\xcb
=\xf2~\x1f\x88\x00\x00\u07d4\xffM\x9c\x84\x84\xc4<B\xff,Z\xb7Y\x99d\x98\xd3#\x99M\x89\xd8\xd
7&\xb7\x17z\x80\x00\x00\u07d4\xffO\xc6`i\x04lRVX\xc37\xa9\x17\xf2\u05382\xb4\t\x89lk\x93[\x8b
\xbd@\x00\x00\u07d4\xffQb\xf25M\u0112\xc7_\xd6\xe3\xa1\a&\x86`\xee\xcbG\x89\\(=A\x03\x94\x
10\x00\x00\u07d4\xffT[\xbbf\xfb\xd0\x0e\xb5\xe67?\xf4\xe3&\xf5\xfe\xb5\xfe\x12\x89\x01\x15\x8eF
\t\x13\xd0\x00\x00\u07d4\xff^~\xe7\xd5\x11H!\xe1Y\u0725\xe8\x1f\x18\xf1\xbf\xff\xbf\xf9\x89lk\x93
[\x8b\xbd@\x00\x00\u07d4\xffa\xc9\xc1\xb7\xa3\u0635;\xba
\xb3DfTK{!fD\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xffeQ\x1c\xad\xa2Y&\f\x1d\xdcA\x97N\xca\xec\
xd3-
cW\x89_h\xe8\x13\x1e\u03c0\x00\x00\xe0\x94\xffxC\xc7\x01\n\xa7\xe6\x15\x19\xb7b\xdf\xe4\x91
$\xa7k\x0eN\x8a\u0171y$A+\x9b\xb0\x00\x00\xe0\x94\xffxT\x17V\xab+pn\rp\xb1\x8a\xdbp\x0f\xc4
\xf1d=\x8a\t(\x96R\x9b\xad\u0708\x00\x00\u07d4\xff\x83\x85PQ\xee\x8f\xfbp\xb4\x81}\xba2\x11\x
ed#U\x86\x9d\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xff\x85\x0e;\xe1\xebjMrl\b\xfas\xaa\xd3
X\xf3\x97\x06\u0689i*\xe8\x89p\x81\xd0\x00\x00\u07d4\xff\x86\xe5\xe8\xe1[S\x90\x96\x00\xe4\x1
3\b\u06b7_\x0e$\xe4k\x890\xebP\xd2\xe1@\x80\x00\x00\u07d4\xff\x88\xeb\xac\xc4\x1b6\x87\xf3\
x9eKY\xe1YY\x9b\x80\u02e3?\x89\x15\xaf\x1dx\xb5\x8c@\x00\x00\u07d4\xff\x8a,\xa5\xa8\x133\x
f1\x99\x98%_
2V\xe1\xa8\x19\xc0\xaa\x89\f$\x9f\xdd2w\x80\x00\x00\u07d4\xff\x8e\xb0}\xe3\u051d\x9dR\xbb\xe
8\xe5\xb2m\xbe\x1d\x16\x0f\xa84\x89\xd8\x14\u0739DS\b\x00\x00\xe0\x94\xff\xa4\xaf\xf1\xa3\u0
07f\x98K\ng'!I':\xe9\xbdA\u3f0a\x02\x1e\x19\xe0\u027a\xb2@\x00\x00\u07d4\xff\xa6\x96\xec\xbdx
~f\xab\xaeO\xe8{c_\a\xcaW\xd8H\x89Hz\x9a0E9D\x00\x00\u07d4\xff\xac=\xb8y\xa6\xc7\x15\x8e\

x8d\xec`;@tc\xba\r1\u03c9j\xcb=\xf2~\x1f\x88\x00\x00\xe0\x94\xff\xad=\xd7N,\x1fyj\xc6@\xdeV\u0719\xb4\u01d2\xa4\x02\x8a\x01\x0f\f\xf0d\xddY
\x00\x00\u07d4\xff\xb0G&\u07e4\x1a\xfd\xc8\x19\x16\x84\x18a\x04r\x97\r{\xfc\x89\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xff\xb3\xbc\xc3\x19j\x8c<\xb84\xce\xc9L4\xfe\xd3[>\x10T\x89H\xa4<T`/p\x00\x00\u07d4\xff\xb9tg3g\xf5\xc0{\xe5\xfd'\r\u0137\x13\x8b\aMW\x89\x85\xeb\u023d\xb9\x06m\x80\x00\u07d4\xff\xb9\xc7!~ft01\xeb7z\xf6\\w\xdbsY\xdc\u0689\x02+\x1c\x8c\x12'\xa0\x00\x00\u07d4\xff\xbc=\xa08\x1e\xc39\xc1\xc0I\xeb\x1e\xd9\xee4\xfd\u03a6\u0289\xd8\xd7&\xb7\x17z\x80\x00\x00\u07d4\xff\xc5\xfcK~\x8a\x02\x93\xff9\xa3\xa0\xf7\xd6\r&F\xd3zt\x89lk\x93[\x8b\xbd@\x00\x00\u07d4\xff\xc9\xcc0\x94\xb0A\xad\x0e\ao\x96\x8a\r\xe3\xb1g%Xf\x89\x17p\xc1e\v\xee\xe8\x00\x00\u07d4\xff\xd5\x17\x0f\u0468\x11\x8dU\x8eu\x11\xe3d\xb2I\x06\xc4\xf6\xb3\x89\x03A\xd8\xcd'\xf1X\x80\x00\xe0\x94\xff\xd6\u0695\x8e\xec\xbc\x01k\xab\x91\x05\x84@\u04dbA\u01fe\x83\x8a\x04<3\xc1\x93ud\x80\x00\x00\u07d4\xff\xe0\xe9\x97\xf1\x97za_Z1Z\xf4\x13\xfdHi4;\xa0\x89\x05l\xd5_\xc6M\xfe\x00\x00\u07d4\xff\u2375<\x90D\xb4\xec\xd4\x05?\u0474\xb1\rpV\u0188\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xff\xe2\xe2\x8c?\xb7GI\xd7\xe7\x80\u070a]B%8\xe6\xe4Q\x89\r\xbb\x81\xe0[\xc1-
\x80\x00\u07d4\xff\xe8\xcb\xc1h\x1e^\x9d\xb7J\x0f\x93\xf8\xed%\x89u\x19\x12\x0f\x89Q\xb1\u04c3\x92a\xac\x00\x00\u07d4\xff\xea\xc00^\xde:\x91R\x95\xec\x8ea\xc7\xf8\x81\x00oDt\x89\x05V\xf6L\x1f\xe7\xfa\x00\x00\u07d4\xff\xec\t\x13\xc65\xba\xca/^W\xa3z\xa9\xfb{l\x9bn&\x89+\xa3\x9e\x82\xed]t\x00\x00\xe0\x94\xff\xf3:;\xd3j\xbd\xbdA'\a\xb8\xe3\x10\xd6\x01\x14T\xa7\xae\x8a\x01\xb1\xaeMn.\xf5\x00\x00\x00\u07d4\xff\xf4\xba\u0556c4y\xa2\xa2\x9f\x9a\x8b?x\xee\xfd\a\xe6\xee\x89\x05k\xc7^-
c\x10\x00\x00\u07d4\xff\xf7\xac\x99\xc8\xe4\xfe\xb6\f\x97P\x05K\xdc\x14\xce\x18W\xf1\x81\x8965\u026d\xc5\u07a0\x00\x00"
const testnetAllocData =
"\xf9\x03\xa4\u0080\x01\xc2\x01\x01\xc2\x02\x01\xc2\x03\x01\xc2\x04\x01\xc2\x05\x01\xc2\x06\x01\xc2\a\x01\xc2\b\x01\xc2\t\x01\xc2\n\x80\xc2\v\x80\xc2\f\x80\xc2\r\x80\xc2\x0e\x80\xc2\x0f\x80\xc2\x10\x80\xc2\x11\x80\xc2\x12\x80\xc2\x13\x80\xc2\x14\x80\xc2\x15\x80\xc2\x16\x80\xc2\x17\x80\xc2\x18\x80\xc2\x19\x80\xc2\x1a\x80\xc2\x1b\x80\xc2\x1c\x80\xc2\x1d\x80\xc2\x1e\x80\xc2\x1f\x80\xc2
\x80\xc2!\x80\xc2\"\x80\xc2#\x80\xc2$\x80\xc2%\x80\xc2&\x80\xc2'\x80\xc2(\x80\xc2)\x80\xc2*\x80\xc2+\x80\xc2,\x80\xc2-
\x80\xc2.\x80\xc2/\x80\xc20\x80\xc21\x80\xc22\x80\xc23\x80\xc24\x80\xc25\x80\xc26\x80\xc27\x80\xc28\x80\xc29\x80\xc2:\x80\xc2;\x80\xc2<\x80\xc2=\x80\xc2>\x80\xc2?\x80\xc2@\x80\xc2A\x80\xc2B\x80\xc2C\x80\xc2D\x80\xc2E\x80\xc2F\x80\xc2G\x80\xc2H\x80\xc2I\x80\xc2J\x80\xc2K\x80\xc2L\x80\xc2M\x80\xc2N\x80\xc2O\x80\xc2P\x80\xc2Q\x80\xc2R\x80\xc2S\x80\xc2T\x80\xc2U\x80\xc2V\x80\xc2W\x80\xc2X\x80\xc2Y\x80\xc2Z\x80\xc2[\x80\xc2\\\x80\xc2]\x80\xc2^\x80\xc2_\x80\xc2`\x80\xc2a\x80\xc2b\x80\xc2c\x80\xc2d\x80\xc2e\x80\xc2f\x80\xc2g\x80\xc2h\x80\xc2i\x80\xc2j\x80\xc2k\x80\xc2l\x80\xc2m\x80\xc2n\x80\xc2o\x80\xc2p\x80\xc2q\x80\xc2r\x80\xc2s\x80\xc2t\x80\xc2u\x80\xc2v\x80\xc2w\x80\xc2x\x80\xc2y\x80\xc2z\x80\xc2{\x80\xc2|\x80\xc2}\x80\xc2~\x80\xc2\u007f\x80\u00c1\x80\x80\u00c1\x81\x80\u00c1\x82\x80\u00c1\x83\x80\u00c1\x84\x80\u00c1\x85\x80\u00c1\x86\x80\u00c1\x87\x80\u00c1\x88\x80\u00c1\x89\x80\u00c1\x8a\x80\u00c1\x8b\

x80\u00c1\x8c\x80\u00c1\x8d\x80\u00c1\x8e\x80\u00c1\x8f\x80\u00c1\x90\x80\u00c1\x91\x80\u0
0c1\x92\x80\u00c1\x93\x80\u00c1\x94\x80\u00c1\x95\x80\u00c1\x96\x80\u00c1\x97\x80\u00c1\x
98\x80\u00c1\x99\x80\u00c1\x9a\x80\u00c1\x9b\x80\u00c1\x9c\x80\u00c1\x9d\x80\u00c1\x9e\x8
0\u00c1\x9f\x80\u00c1\xa0\x80\u00c1\xa1\x80\u00c1\xa2\x80\u00c1\xa3\x80\u00c1\xa4\x80\u00c
1\xa5\x80\u00c1\xa6\x80\u00c1\xa7\x80\u00c1\xa8\x80\u00c1\xa9\x80\u00c1\xaa\x80\u00c1\xab\
x80\u00c1\xac\x80\u00c1\xad\x80\u00c1\xae\x80\u00c1\xaf\x80\u00c1\xb0\x80\u00c1\xb1\x80\u0
0c1\xb2\x80\u00c1\xb3\x80\u00c1\xb4\x80\u00c1\xb5\x80\u00c1\xb6\x80\u00c1\xb7\x80\u00c1\x
b8\x80\u00c1\xb9\x80\u00c1\xba\x80\u00c1\xbb\x80\u00c1\xbc\x80\u00c1\xbd\x80\u00c1\xbe\x8
0\u00c1\xbf\x80\u00c1\xc0\x80\u00c1\xc1\x80\u00c1\u0080\u00c1\u00c0\u00c1\u0100\u00c1\u0
140\u00c1\u0180\u00c1\u01c0\u00c1\u0200\u00c1\u0240\u00c1\u0280\u00c1\u02c0\u00c1\u030
0\u00c1\u0340\u00c1\u0380\u00c1\u03c0\u00c1\u0400\u00c1\u0440\u00c1\u0480\u00c1\u04c0\
u00c1\u0500\u00c1\u0540\u00c1\u0580\u00c1\u05c0\u00c1\u0600\u00c1\u0640\u00c1\u0680\u0
0c1\u06c0\u00c1\u0700\u00c1\u0740\u00c1\u0780\u00c1\u07c0\u00c1\xe0\x80\u00c1\xe1\x80\u
00c1\xe2\x80\u00c1\xe3\x80\u00c1\xe4\x80\u00c1\xe5\x80\u00c1\xe6\x80\u00c1\xe7\x80\u00c1\
xe8\x80\u00c1\xe9\x80\u00c1\xea\x80\u00c1\xeb\x80\u00c1\xec\x80\u00c1\xed\x80\u00c1\xee\x8
0\u00c1\xef\x80\u00c1\xf0\x80\u00c1\xf1\x80\u00c1\xf2\x80\u00c1\xf3\x80\u00c1\xf4\x80\u00c1\x
f5\x80\u00c1\xf6\x80\u00c1\xf7\x80\u00c1\xf8\x80\u00c1\xf9\x80\u00c1\xfa\x80\u00c1\xfb\x80\u0
0c1\xfc\x80\u00c1\xfd\x80\u00c1\xfe\x80\u00c1\xff\x80\u3507KT\xa8\xbd\x15)f\xd6?pk\xae\x1f\xfe
\xb0A\x19!\xe5\x8d\f\x9f,\x9c\xd0Ft\xed\xea@\x00\x00\x00"
const rinkebyAllocData =
"\xf9\x03\xb7\u0080\x01\xc2\x01\x01\xc2\x02\x01\xc2\x03\x01\xc2\x04\x01\xc2\x05\x01\xc2\x06\x
01\xc2\a\x01\xc2\b\x01\xc2\t\x01\xc2\n\x01\xc2\v\x01\xc2\f\x01\xc2\r\x01\xc2\x0e\x01\xc2\x0f\x01\
xc2\x10\x01\xc2\x11\x01\xc2\x12\x01\xc2\x13\x01\xc2\x14\x01\xc2\x15\x01\xc2\x16\x01\xc2\x17\x
01\xc2\x18\x01\xc2\x19\x01\xc2\x1a\x01\xc2\x1b\x01\xc2\x1c\x01\xc2\x1d\x01\xc2\x1e\x01\xc2\x1
f\x01\xc2
\x01\xc2!\x01\xc2\"\x01\xc2#\x01\xc2$\x01\xc2%\x01\xc2&\x01\xc2'\x01\xc2(\x01\xc2)\x01\xc2*\x0
1\xc2+\x01\xc2,\x01\xc2-
\x01\xc2.\x01\xc2/\x01\xc20\x01\xc21\x01\xc22\x01\xc23\x01\xc24\x01\xc25\x01\xc26\x01\xc27\x0
1\xc28\x01\xc29\x01\xc2:\x01\xc2;\x01\xc2<\x01\xc2=\x01\xc2>\x01\xc2?\x01\xc2@\x01\xc2A\x01
\xc2B\x01\xc2C\x01\xc2D\x01\xc2E\x01\xc2F\x01\xc2G\x01\xc2H\x01\xc2I\x01\xc2J\x01\xc2K\x01
\xc2L\x01\xc2M\x01\xc2N\x01\xc2O\x01\xc2P\x01\xc2Q\x01\xc2R\x01\xc2S\x01\xc2T\x01\xc2U\x
01\xc2V\x01\xc2W\x01\xc2X\x01\xc2Y\x01\xc2Z\x01\xc2[\x01\xc2\\\x01\xc2]\x01\xc2^\x01\xc2_\x0
1\xc2`\x01\xc2a\x01\xc2b\x01\xc2c\x01\xc2d\x01\xc2e\x01\xc2f\x01\xc2g\x01\xc2h\x01\xc2i\x01\x
c2j\x01\xc2k\x01\xc2l\x01\xc2m\x01\xc2n\x01\xc2o\x01\xc2p\x01\xc2q\x01\xc2r\x01\xc2s\x01\xc2t
\x01\xc2u\x01\xc2v\x01\xc2w\x01\xc2x\x01\xc2y\x01\xc2z\x01\xc2{\x01\xc2|\x01\xc2}\x01\xc2~\x0
1\xc2\u007f\x01\u00c1\x80\x01\u00c1\x81\x01\u00c1\x82\x01\u00c1\x83\x01\u00c1\x84\x01\u00c
1\x85\x01\u00c1\x86\x01\u00c1\x87\x01\u00c1\x88\x01\u00c1\x89\x01\u00c1\x8a\x01\u00c1\x8b\
x01\u00c1\x8c\x01\u00c1\x8d\x01\u00c1\x8e\x01\u00c1\x8f\x01\u00c1\x90\x01\u00c1\x91\x01\u0
0c1\x92\x01\u00c1\x93\x01\u00c1\x94\x01\u00c1\x95\x01\u00c1\x96\x01\u00c1\x97\x01\u00c1\x
98\x01\u00c1\x99\x01\u00c1\x9a\x01\u00c1\x9b\x01\u00c1\x9c\x01\u00c1\x9d\x01\u00c1\x9e\x0
1\u00c1\x9f\x01\u00c1\xa0\x01\u00c1\xa1\x01\u00c1\xa2\x01\u00c1\xa3\x01\u00c1\xa4\x01\u00c
1\xa5\x01\u00c1\xa6\x01\u00c1\xa7\x01\u00c1\xa8\x01\u00c1\xa9\x01\u00c1\xaa\x01\u00c1\xab\

x01\u00c1\xac\x01\u00c1\xad\x01\u00c1\xae\x01\u00c1\xaf\x01\u00c1\xb0\x01\u00c1\xb1\x01\u0
0c1\xb2\x01\u00c1\xb3\x01\u00c1\xb4\x01\u00c1\xb5\x01\u00c1\xb6\x01\u00c1\xb7\x01\u00c1\x
b8\x01\u00c1\xb9\x01\u00c1\xba\x01\u00c1\xbb\x01\u00c1\xbc\x01\u00c1\xbd\x01\u00c1\xbe\x0
1\u00c1\xbf\x01\u00c1\xc0\x01\u00c1\xc1\x01\u00c1\xc2\x01\u00c1\xc3\x01\u00c1\xc4\x01\u00c
1\xc5\x01\u00c1\xc6\x01\u00c1\xc7\x01\u00c1\xc8\x01\u00c1\xc9\x01\u00c1\xca\x01\u00c1\xcb\x
01\u00c1\xcc\x01\u00c1\xcd\x01\u00c1\xce\x01\u00c1\xcf\x01\u00c1\xd0\x01\u00c1\xd1\x01\u00
c1\xd2\x01\u00c1\xd3\x01\u00c1\xd4\x01\u00c1\xd5\x01\u00c1\xd6\x01\u00c1\xd7\x01\u00c1\xd
8\x01\u00c1\xd9\x01\u00c1\xda\x01\u00c1\xdb\x01\u00c1\xdc\x01\u00c1\xdd\x01\u00c1\xde\x01\
u00c1\xdf\x01\u00c1\xe0\x01\u00c1\xe1\x01\u00c1\xe2\x01\u00c1\xe3\x01\u00c1\xe4\x01\u00c1\
xe5\x01\u00c1\xe6\x01\u00c1\xe7\x01\u00c1\xe8\x01\u00c1\xe9\x01\u00c1\xea\x01\u00c1\xeb\x
01\u00c1\xec\x01\u00c1\xed\x01\u00c1\xee\x01\u00c1\xef\x01\u00c1\xf0\x01\u00c1\xf1\x01\u00c
1\xf2\x01\u00c1\xf3\x01\u00c1\xf4\x01\u00c1\xf5\x01\u00c1\xf6\x01\u00c1\xf7\x01\u00c1\xf8\x01\
u00c1\xf9\x01\u00c1\xfa\x01\u00c1\xfb\x01\u00c1\xfc\x01\u00c1\xfd\x01\u00c1\xfe\x01\u00c1\xff\
x01\xf6\x941\xb9\x8d\x14\x00{\xde\xe67)\x80\x86\x98\x8a\v\xbd1\x18E#\xa0\x02\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00"
const devAllocData =
"\xf9\x01\x94\xc2\x01\x01\xc2\x02\x01\xc2\x03\x01\xc2\x04\x01\xf0\x94\x1a&3\x8f\r\x90^)_\u0337
\x1f\xa9\ua11f\xfa\x12\xaa\xf4\x9a\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf0\x94.\xf4q\x00\xe0x{\x91Q\x05\xfd^?O\xf6u
y\xd5\u02da\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\xf0\x94l8jK&\xf7<\x80/4g?rH\xbb\x11\x8f\x97BJ\x9a\x01\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf0\x9
4\xb9\xc0\x15\x91\x8b\u06ba$\xb4\xff\x05z\x92\xa3\x87=n\xb2\x01\xbe\x9a\x01\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf0\x94\
xcd*=\x9f\x93\x8e\x13\u0354~\xc0Z\xbc\u007f\xe74\u07cd\xd8&\x9a\x01\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf0\x94\xdb\xd
b\xdb,\xbd#\xb7\x83t\x1e\x8d\u007f\xcfQ\xe4Y\xb4\x97\u499a\x01\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf0\x94\xe4\x15{4\xea\
x96\x15\u03fd\xe6\xb4\xfd\xa4\x19\x82\x81$\xb7\fx\x9a\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf0\x94\xe6qo\x95D\xa5lS\r\x
86\x8eK\xfb\xac\xb1r1[\u07ad\x9a\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"

81:F:\git\coin\ethereum\go-ethereum\core\genesis_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"math/big"
"reflect"

```
"testing"

"github.com/davecgh/go-spew/spew"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/params"
)

func TestDefaultGenesisBlock(t *testing.T) {
block, _ := DefaultGenesisBlock().ToBlock()
if block.Hash() != params.MainnetGenesisHash {
t.Errorf("wrong mainnet genesis hash, got %v, want %v", block.Hash(),
params.MainnetGenesisHash)
}
block, _ = DefaultTestnetGenesisBlock().ToBlock()
if block.Hash() != params.TestnetGenesisHash {
t.Errorf("wrong testnet genesis hash, got %v, want %v", block.Hash(),
params.TestnetGenesisHash)
}
}

func TestSetupGenesis(t *testing.T) {
var (
customghash =
common.HexToHash("0x89c99d90b79719238d2645c7642f2c9295246e80775b38cfd162b696817f
bd50")
customg     = Genesis{
Config: &params.ChainConfig{HomesteadBlock: big.NewInt(3)},
Alloc: GenesisAlloc{
{1}: {Balance: big.NewInt(1), Storage: map[common.Hash]common.Hash{{1}: {1}}},
},
}
oldcustomg = customg
)
oldcustomg.Config = &params.ChainConfig{HomesteadBlock: big.NewInt(2)}
tests := []struct {
name       string
fn         func(ethdb.Database) (*params.ChainConfig, common.Hash, error)
wantConfig *params.ChainConfig
```

```
wantHash   common.Hash
wantErr    error
}{
{
name: "genesis without ChainConfig",
fn: func(db ethdb.Database) (*params.ChainConfig, common.Hash, error) {
return SetupGenesisBlock(db, new(Genesis))
},
wantErr:    errGenesisNoConfig,
wantConfig: params.AllProtocolChanges,
},
{
name: "no block in DB, genesis == nil",
fn: func(db ethdb.Database) (*params.ChainConfig, common.Hash, error) {
return SetupGenesisBlock(db, nil)
},
wantHash:   params.MainnetGenesisHash,
wantConfig: params.MainnetChainConfig,
},
{
name: "mainnet block in DB, genesis == nil",
fn: func(db ethdb.Database) (*params.ChainConfig, common.Hash, error) {
DefaultGenesisBlock().MustCommit(db)
return SetupGenesisBlock(db, nil)
},
wantHash:   params.MainnetGenesisHash,
wantConfig: params.MainnetChainConfig,
},
{
name: "custom block in DB, genesis == nil",
fn: func(db ethdb.Database) (*params.ChainConfig, common.Hash, error) {
customg.MustCommit(db)
return SetupGenesisBlock(db, nil)
},
wantHash:   customghash,
wantConfig: customg.Config,
},
{
name: "custom block in DB, genesis == testnet",
fn: func(db ethdb.Database) (*params.ChainConfig, common.Hash, error) {
customg.MustCommit(db)
return SetupGenesisBlock(db, DefaultTestnetGenesisBlock())
```

```
},
wantErr:    &GenesisMismatchError{Stored: customghash, New: params.TestnetGenesisHash},
wantHash:   params.TestnetGenesisHash,
wantConfig: params.TestnetChainConfig,
},
{
name: "compatible config in DB",
fn: func(db ethdb.Database) (*params.ChainConfig, common.Hash, error) {
oldcustomg.MustCommit(db)
return SetupGenesisBlock(db, &customg)
},
wantHash:   customghash,
wantConfig: customg.Config,
},
{
name: "incompatible config in DB",
fn: func(db ethdb.Database) (*params.ChainConfig, common.Hash, error) {
// Commit the 'old' genesis block with Homestead transition at #2.
// Advance to block #4, past the homestead transition block of customg.
genesis := oldcustomg.MustCommit(db)
bc, _ := NewBlockChain(db, oldcustomg.Config, ethash.NewFullFaker(), new(event.TypeMux),
vm.Config{})
bc.SetValidator(bproc{})
bc.InsertChain(makeBlockChainWithDiff(genesis, []int{2, 3, 4, 5}, 0))
bc.CurrentBlock()
// This should return a compatibility error.
return SetupGenesisBlock(db, &customg)
},
wantHash:   customghash,
wantConfig: customg.Config,
wantErr: &params.ConfigCompatError{
What:         "Homestead fork block",
StoredConfig: big.NewInt(2),
NewConfig:    big.NewInt(3),
RewindTo:     1,
},
},
}

for _, test := range tests {
db, _ := ethdb.NewMemDatabase()
config, hash, err := test.fn(db)
```

```
// Check the return values.
if !reflect.DeepEqual(err, test.wantErr) {
spew := spew.ConfigState{DisablePointerAddresses: true, DisableCapacities: true}
t.Errorf("%s: returned error %#v, want %#v", test.name, spew.NewFormatter(err),
spew.NewFormatter(test.wantErr))
}
if !reflect.DeepEqual(config, test.wantConfig) {
t.Errorf("%s:\nreturned %v\nwant     %v", test.name, config, test.wantConfig)
}
if hash != test.wantHash {
t.Errorf("%s: returned hash %s, want %s", test.name, hash.Hex(), test.wantHash.Hex())
} else if err == nil {
// Check database content.
stored := GetBlock(db, test.wantHash, 0)
if stored.Hash() != test.wantHash {
t.Errorf("%s: block in DB has hash %s, want %s", test.name, stored.Hash(), test.wantHash)
}
}
}
}
```

82:F:\git\coin\ethereum\go-ethereum\core\gen_genesis.go

```
func (g Genesis) MarshalJSON() ([]byte, error) {
type Genesis struct {
Config     *params.ChainConfig                  `json:"config"`
Nonce      math.HexOrDecimal64                  `json:"nonce"`
Timestamp  math.HexOrDecimal64                  `json:"timestamp"`
ParentHash common.Hash                          `json:"parentHash"`
ExtraData  hexutil.Bytes                        `json:"extraData"`
GasLimit   math.HexOrDecimal64                  `json:"gasLimit"   gencodec:"required"`
Difficulty *math.HexOrDecimal256                `json:"difficulty" gencodec:"required"`
Mixhash    common.Hash                          `json:"mixHash"`
Coinbase   common.Address                       `json:"coinbase"`
Alloc      map[common.UnprefixedAddress]GenesisAccount `json:"alloc"      gencodec:"required"`
}
var enc Genesis
enc.Config = g.Config
enc.Nonce = math.HexOrDecimal64(g.Nonce)
enc.Timestamp = math.HexOrDecimal64(g.Timestamp)
enc.ParentHash = g.ParentHash
enc.ExtraData = g.ExtraData
```

```
enc.GasLimit = math.HexOrDecimal64(g.GasLimit)
enc.Difficulty = (*math.HexOrDecimal256)(g.Difficulty)
enc.Mixhash = g.Mixhash
enc.Coinbase = g.Coinbase
if g.Alloc != nil {
enc.Alloc = make(map[common.UnprefixedAddress]GenesisAccount, len(g.Alloc))
for k, v := range g.Alloc {
enc.Alloc[common.UnprefixedAddress(k)] = v
}
}
return json.Marshal(&enc)
}

func (g *Genesis) UnmarshalJSON(input []byte) error {
type Genesis struct {
Config     *params.ChainConfig                `json:"config"`
Nonce      *math.HexOrDecimal64               `json:"nonce"`
Timestamp  *math.HexOrDecimal64               `json:"timestamp"`
ParentHash *common.Hash                       `json:"parentHash"`
ExtraData  hexutil.Bytes                      `json:"extraData"`
GasLimit   *math.HexOrDecimal64               `json:"gasLimit"   gencodec:"required"`
Difficulty *math.HexOrDecimal256              `json:"difficulty" gencodec:"required"`
Mixhash    *common.Hash                       `json:"mixHash"`
Coinbase   *common.Address                    `json:"coinbase"`
Alloc      map[common.UnprefixedAddress]GenesisAccount `json:"alloc"      gencodec:"required"`
}
var dec Genesis
if err := json.Unmarshal(input, &dec); err != nil {
return err
}
if dec.Config != nil {
g.Config = dec.Config
}
if dec.Nonce != nil {
g.Nonce = uint64(*dec.Nonce)
}
if dec.Timestamp != nil {
g.Timestamp = uint64(*dec.Timestamp)
}
if dec.ParentHash != nil {
g.ParentHash = *dec.ParentHash
}
```

```
if dec.ExtraData != nil {
g.ExtraData = dec.ExtraData
}
if dec.GasLimit == nil {
return errors.New("missing required field 'gasLimit' for Genesis")
}
g.GasLimit = uint64(*dec.GasLimit)
if dec.Difficulty == nil {
return errors.New("missing required field 'difficulty' for Genesis")
}
g.Difficulty = (*big.Int)(dec.Difficulty)
if dec.Mixhash != nil {
g.Mixhash = *dec.Mixhash
}
if dec.Coinbase != nil {
g.Coinbase = *dec.Coinbase
}
if dec.Alloc == nil {
return errors.New("missing required field 'alloc' for Genesis")
}
g.Alloc = make(GenesisAlloc, len(dec.Alloc))
for k, v := range dec.Alloc {
g.Alloc[common.Address(k)] = v
}
return nil
}

83:F:\git\coin\ethereum\go-ethereum\core\gen_genesis_account.go
func (g GenesisAccount) MarshalJSON() ([]byte, error) {
type GenesisAccount struct {
Code    hexutil.Bytes               `json:"code,omitempty"`
Storage map[common.Hash]common.Hash `json:"storage,omitempty"`
Balance *math.HexOrDecimal256       `json:"balance" gencodec:"required"`
Nonce   math.HexOrDecimal64         `json:"nonce,omitempty"`
}
var enc GenesisAccount
enc.Code = g.Code
enc.Storage = g.Storage
enc.Balance = (*math.HexOrDecimal256)(g.Balance)
enc.Nonce = math.HexOrDecimal64(g.Nonce)
return json.Marshal(&enc)
}
```

```
func (g *GenesisAccount) UnmarshalJSON(input []byte) error {
type GenesisAccount struct {
Code    hexutil.Bytes               `json:"code,omitempty"`
Storage map[common.Hash]common.Hash `json:"storage,omitempty"`
Balance *math.HexOrDecimal256       `json:"balance" gencodec:"required"`
Nonce   *math.HexOrDecimal64        `json:"nonce,omitempty"`
}
var dec GenesisAccount
if err := json.Unmarshal(input, &dec); err != nil {
return err
}
if dec.Code != nil {
g.Code = dec.Code
}
if dec.Storage != nil {
g.Storage = dec.Storage
}
if dec.Balance == nil {
return errors.New("missing required field 'balance' for GenesisAccount")
}
g.Balance = (*big.Int)(dec.Balance)
if dec.Nonce != nil {
g.Nonce = uint64(*dec.Nonce)
}
return nil
}

84:F:\git\coin\ethereum\go-ethereum\core\headerchain.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
crand "crypto/rand"
"errors"
"fmt"
"math"
"math/big"
mrand "math/rand"
"time"
```

```
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/params"
"github.com/hashicorp/golang-lru"
)

const (
headerCacheLimit = 512
tdCacheLimit     = 1024
numberCacheLimit = 2048
)

// HeaderChain implements the basic block header chain logic that is shared by
// core.BlockChain and light.LightChain. It is not usable in itself, only as
// a part of either structure.
// It is not thread safe either, the encapsulating chain structures should do
// the necessary mutex locking/unlocking.
type HeaderChain struct {
config *params.ChainConfig

chainDb       ethdb.Database
genesisHeader *types.Header

currentHeader     *types.Header // Current head of the header chain (may be above the block chain!)
currentHeaderHash common.Hash   // Hash of the current head of the header chain (prevent recomputing all the time)

headerCache *lru.Cache // Cache for the most recent block headers
tdCache     *lru.Cache // Cache for the most recent block total difficulties
numberCache *lru.Cache // Cache for the most recent block numbers

procInterrupt func() bool

rand   *mrand.Rand
engine consensus.Engine
}

// NewHeaderChain creates a new HeaderChain structure.
```

```
//  getValidator should return the parent's validator
//  procInterrupt points to the parent's interrupt semaphore
//  wg points to the parent's shutdown wait group
func NewHeaderChain(chainDb ethdb.Database, config *params.ChainConfig, engine
consensus.Engine, procInterrupt func() bool) (*HeaderChain, error) {
headerCache, _ := lru.New(headerCacheLimit)
tdCache, _ := lru.New(tdCacheLimit)
numberCache, _ := lru.New(numberCacheLimit)

// Seed a fast but crypto originating random generator
seed, err := crand.Int(crand.Reader, big.NewInt(math.MaxInt64))
if err != nil {
return nil, err
}

hc := &HeaderChain{
config:        config,
chainDb:       chainDb,
headerCache:   headerCache,
tdCache:       tdCache,
numberCache:   numberCache,
procInterrupt: procInterrupt,
rand:          mrand.New(mrand.NewSource(seed.Int64())),
engine:        engine,
}

hc.genesisHeader = hc.GetHeaderByNumber(0)
if hc.genesisHeader == nil {
return nil, ErrNoGenesis
}

hc.currentHeader = hc.genesisHeader
if head := GetHeadBlockHash(chainDb); head != (common.Hash{}) {
if chead := hc.GetHeaderByHash(head); chead != nil {
hc.currentHeader = chead
}
}
hc.currentHeaderHash = hc.currentHeader.Hash()

return hc, nil
}
```

```
// GetBlockNumber retrieves the block number belonging to the given hash
// from the cache or database
func (hc *HeaderChain) GetBlockNumber(hash common.Hash) uint64 {
if cached, ok := hc.numberCache.Get(hash); ok {
return cached.(uint64)
}
number := GetBlockNumber(hc.chainDb, hash)
if number != missingNumber {
hc.numberCache.Add(hash, number)
}
return number
}

// WriteHeader writes a header into the local chain, given that its parent is
// already known. If the total difficulty of the newly inserted header becomes
// greater than the current known TD, the canonical chain is re-routed.
//
// Note: This method is not concurrent-safe with inserting blocks simultaneously
// into the chain, as side effects caused by reorganisations cannot be emulated
// without the real blocks. Hence, writing headers directly should only be done
// in two scenarios: pure-header mode of operation (light clients), or properly
// separated header/block phases (non-archive clients).
func (hc *HeaderChain) WriteHeader(header *types.Header) (status WriteStatus, err error) {
// Cache some values to prevent constant recalculation
var (
hash   = header.Hash()
number = header.Number.Uint64()
)
// Calculate the total difficulty of the header
ptd := hc.GetTd(header.ParentHash, number-1)
if ptd == nil {
return NonStatTy, consensus.ErrUnknownAncestor
}
localTd := hc.GetTd(hc.currentHeaderHash, hc.currentHeader.Number.Uint64())
externTd := new(big.Int).Add(header.Difficulty, ptd)

// Irrelevant of the canonical status, write the td and header to the database
if err := hc.WriteTd(hash, number, externTd); err != nil {
log.Crit("Failed to write header total difficulty", "err", err)
}
if err := WriteHeader(hc.chainDb, header); err != nil {
log.Crit("Failed to write header content", "err", err)
```

```
}
// If the total difficulty is higher than our known, add it to the canonical chain
// Second clause in the if statement reduces the vulnerability to selfish mining.
// Please refer to http://www.cs.cornell.edu/~ie53/publications/btcProcFC.pdf
if externTd.Cmp(localTd) > 0 || (externTd.Cmp(localTd) == 0 && mrand.Float64() < 0.5) {
// Delete any canonical number assignments above the new head
for i := number + 1; ; i++ {
hash := GetCanonicalHash(hc.chainDb, i)
if hash == (common.Hash{}) {
break
}
DeleteCanonicalHash(hc.chainDb, i)
}
// Overwrite any stale canonical number assignments
var (
headHash   = header.ParentHash
headNumber = header.Number.Uint64() - 1
headHeader = hc.GetHeader(headHash, headNumber)
)
for GetCanonicalHash(hc.chainDb, headNumber) != headHash {
WriteCanonicalHash(hc.chainDb, headHash, headNumber)

headHash = headHeader.ParentHash
headNumber = headHeader.Number.Uint64() - 1
headHeader = hc.GetHeader(headHash, headNumber)
}
// Extend the canonical chain with the new header
if err := WriteCanonicalHash(hc.chainDb, hash, number); err != nil {
log.Crit("Failed to insert header number", "err", err)
}
if err := WriteHeadHeaderHash(hc.chainDb, hash); err != nil {
log.Crit("Failed to insert head header hash", "err", err)
}
hc.currentHeaderHash, hc.currentHeader = hash, types.CopyHeader(header)

status = CanonStatTy
} else {
status = SideStatTy
}

hc.headerCache.Add(hash, header)
hc.numberCache.Add(hash, number)
```

```
return
}

// WhCallback is a callback function for inserting individual headers.
// A callback is used for two reasons: first, in a LightChain, status should be
// processed and light chain events sent, while in a BlockChain this is not
// necessary since chain events are sent after inserting blocks. Second, the
// header writes should be protected by the parent chain mutex individually.
type WhCallback func(*types.Header) error

func (hc *HeaderChain) ValidateHeaderChain(chain []*types.Header, checkFreq int) (int, error) {
// Do a sanity check that the provided chain is actually ordered and linked
for i := 1; i < len(chain); i++ {
if chain[i].Number.Uint64() != chain[i-1].Number.Uint64()+1 || chain[i].ParentHash != chain[i-
1].Hash() {
// Chain broke ancestry, log a messge (programming error) and skip insertion
log.Error("Non contiguous header insert", "number", chain[i].Number, "hash", chain[i].Hash(),
"parent", chain[i].ParentHash, "prevnumber", chain[i-1].Number, "prevhash", chain[i-1].Hash())

return 0, fmt.Errorf("non contiguous insert: item %d is #%d [%x…], item %d is #%d [%x…] (parent
[%x…])", i-1, chain[i-1].Number,
chain[i-1].Hash().Bytes()[:4], i, chain[i].Number, chain[i].Hash().Bytes()[:4], chain[i].ParentHash[:4])
}
}

// Generate the list of seal verification requests, and start the parallel verifier
seals := make([]bool, len(chain))
for i := 0; i < len(seals)/checkFreq; i++ {
index := i*checkFreq + hc.rand.Intn(checkFreq)
if index >= len(seals) {
index = len(seals) - 1
}
seals[index] = true
}
seals[len(seals)-1] = true // Last should always be verified to avoid junk

abort, results := hc.engine.VerifyHeaders(hc, chain, seals)
defer close(abort)

// Iterate over the headers and ensure they all check out
for i, header := range chain {
```

```
// If the chain is terminating, stop processing blocks
if hc.procInterrupt() {
log.Debug("Premature abort during headers verification")
return 0, errors.New("aborted")
}
// If the header is a banned one, straight out abort
if BadHashes[header.Hash()] {
return i, ErrBlacklistedHash
}
// Otherwise wait for headers checks and ensure they pass
if err := <-results; err != nil {
return i, err
}
}

return 0, nil
}

// InsertHeaderChain attempts to insert the given header chain in to the local
// chain, possibly creating a reorg. If an error is returned, it will return the
// index number of the failing header as well an error describing what went wrong.
//
// The verify parameter can be used to fine tune whether nonce verification
// should be done or not. The reason behind the optional check is because some
// of the header retrieval mechanisms already need to verfy nonces, as well as
// because nonces can be verified sparsely, not needing to check each.
func (hc *HeaderChain) InsertHeaderChain(chain []*types.Header, writeHeader WhCallback, start
time.Time) (int, error) {
// Collect some import statistics to report on
stats := struct{ processed, ignored int }{}
// All headers passed verification, import them into the database
for i, header := range chain {
// Short circuit insertion if shutting down
if hc.procInterrupt() {
log.Debug("Premature abort during headers import")
return i, errors.New("aborted")
}
// If the header's already known, skip it, otherwise store
if hc.GetHeader(header.Hash(), header.Number.Uint64()) != nil {
stats.ignored++
continue
}
```

```
if err := writeHeader(header); err != nil {
return i, err
}
stats.processed++
}
// Report some public statistics so the user has a clue what's going on
last := chain[len(chain)-1]
log.Info("Imported new block headers", "count", stats.processed, "elapsed",
common.PrettyDuration(time.Since(start)),
"number", last.Number, "hash", last.Hash(), "ignored", stats.ignored)

return 0, nil
}

// GetBlockHashesFromHash retrieves a number of block hashes starting at a given
// hash, fetching towards the genesis block.
func (hc *HeaderChain) GetBlockHashesFromHash(hash common.Hash, max uint64)
[]common.Hash {
// Get the origin header from which to fetch
header := hc.GetHeaderByHash(hash)
if header == nil {
return nil
}
// Iterate the headers until enough is collected or the genesis reached
chain := make([]common.Hash, 0, max)
for i := uint64(0); i < max; i++ {
next := header.ParentHash
if header = hc.GetHeader(next, header.Number.Uint64()-1); header == nil {
break
}
chain = append(chain, next)
if header.Number.Sign() == 0 {
break
}
}
return chain
}

// GetTd retrieves a block's total difficulty in the canonical chain from the
// database by hash and number, caching it if found.
func (hc *HeaderChain) GetTd(hash common.Hash, number uint64) *big.Int {
// Short circuit if the td's already in the cache, retrieve otherwise
```

```
if cached, ok := hc.tdCache.Get(hash); ok {
return cached.(*big.Int)
}
td := GetTd(hc.chainDb, hash, number)
if td == nil {
return nil
}
// Cache the found body for next time and return
hc.tdCache.Add(hash, td)
return td
}

// GetTdByHash retrieves a block's total difficulty in the canonical chain from the
// database by hash, caching it if found.
func (hc *HeaderChain) GetTdByHash(hash common.Hash) *big.Int {
return hc.GetTd(hash, hc.GetBlockNumber(hash))
}

// WriteTd stores a block's total difficulty into the database, also caching it
// along the way.
func (hc *HeaderChain) WriteTd(hash common.Hash, number uint64, td *big.Int) error {
if err := WriteTd(hc.chainDb, hash, number, td); err != nil {
return err
}
hc.tdCache.Add(hash, new(big.Int).Set(td))
return nil
}

// GetHeader retrieves a block header from the database by hash and number,
// caching it if found.
func (hc *HeaderChain) GetHeader(hash common.Hash, number uint64) *types.Header {
// Short circuit if the header's already in the cache, retrieve otherwise
if header, ok := hc.headerCache.Get(hash); ok {
return header.(*types.Header)
}
header := GetHeader(hc.chainDb, hash, number)
if header == nil {
return nil
}
// Cache the found header for next time and return
hc.headerCache.Add(hash, header)
return header
```

```
}

// GetHeaderByHash retrieves a block header from the database by hash, caching it if
// found.
func (hc *HeaderChain) GetHeaderByHash(hash common.Hash) *types.Header {
return hc.GetHeader(hash, hc.GetBlockNumber(hash))
}

// HasHeader checks if a block header is present in the database or not, caching
// it if present.
func (hc *HeaderChain) HasHeader(hash common.Hash) bool {
return hc.GetHeaderByHash(hash) != nil
}

// GetHeaderByNumber retrieves a block header from the database by number,
// caching it (associated with its hash) if found.
func (hc *HeaderChain) GetHeaderByNumber(number uint64) *types.Header {
hash := GetCanonicalHash(hc.chainDb, number)
if hash == (common.Hash{}) {
return nil
}
return hc.GetHeader(hash, number)
}

// CurrentHeader retrieves the current head header of the canonical chain. The
// header is retrieved from the HeaderChain's internal cache.
func (hc *HeaderChain) CurrentHeader() *types.Header {
return hc.currentHeader
}

// SetCurrentHeader sets the current head header of the canonical chain.
func (hc *HeaderChain) SetCurrentHeader(head *types.Header) {
if err := WriteHeadHeaderHash(hc.chainDb, head.Hash()); err != nil {
log.Crit("Failed to insert head header hash", "err", err)
}
hc.currentHeader = head
hc.currentHeaderHash = head.Hash()
}

// DeleteCallback is a callback function that is called by SetHead before
// each header is deleted.
type DeleteCallback func(common.Hash, uint64)
```

```
// SetHead rewinds the local chain to a new head. Everything above the new head
// will be deleted and the new one set.
func (hc *HeaderChain) SetHead(head uint64, delFn DeleteCallback) {
height := uint64(0)
if hc.currentHeader != nil {
height = hc.currentHeader.Number.Uint64()
}

for hc.currentHeader != nil && hc.currentHeader.Number.Uint64() > head {
hash := hc.currentHeader.Hash()
num := hc.currentHeader.Number.Uint64()
if delFn != nil {
delFn(hash, num)
}
DeleteHeader(hc.chainDb, hash, num)
DeleteTd(hc.chainDb, hash, num)
hc.currentHeader = hc.GetHeader(hc.currentHeader.ParentHash,
hc.currentHeader.Number.Uint64()-1)
}
// Roll back the canonical chain numbering
for i := height; i > head; i-- {
DeleteCanonicalHash(hc.chainDb, i)
}
// Clear out any stale content from the caches
hc.headerCache.Purge()
hc.tdCache.Purge()
hc.numberCache.Purge()

if hc.currentHeader == nil {
hc.currentHeader = hc.genesisHeader
}
hc.currentHeaderHash = hc.currentHeader.Hash()

if err := WriteHeadHeaderHash(hc.chainDb, hc.currentHeaderHash); err != nil {
log.Crit("Failed to reset head header hash", "err", err)
}
}

// SetGenesis sets a new genesis block header for the chain
func (hc *HeaderChain) SetGenesis(head *types.Header) {
hc.genesisHeader = head
```

```
}

// Config retrieves the header chain's chain configuration.
func (hc *HeaderChain) Config() *params.ChainConfig { return hc.config }

// Engine retrieves the header chain's consensus engine.
func (hc *HeaderChain) Engine() consensus.Engine { return hc.engine }

// GetBlock implements consensus.ChainReader, and returns nil for every input as
// a header chain does not have blocks available for retrieval.
func (hc *HeaderChain) GetBlock(hash common.Hash, number uint64) *types.Block {
return nil
}

85:F:\git\coin\ethereum\go-ethereum\core\helper_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package core

import (
"container/list"
"fmt"

"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/event"
)

// Implement our EthTest Manager
type TestManager struct {
// stateManager *StateManager
eventMux *event.TypeMux

db         ethdb.Database
txPool     *TxPool
blockChain *BlockChain
Blocks     []*types.Block
}

func (tm *TestManager) IsListening() bool {
return false
}
```

```
func (tm *TestManager) IsMining() bool {
return false
}

func (tm *TestManager) PeerCount() int {
return 0
}

func (tm *TestManager) Peers() *list.List {
return list.New()
}

func (tm *TestManager) BlockChain() *BlockChain {
return tm.blockChain
}

func (tm *TestManager) TxPool() *TxPool {
return tm.txPool
}

// func (tm *TestManager) StateManager() *StateManager {
// return tm.stateManager
// }

func (tm *TestManager) EventMux() *event.TypeMux {
return tm.eventMux
}

// func (tm *TestManager) KeyManager() *crypto.KeyManager {
// return nil
// }

func (tm *TestManager) Db() ethdb.Database {
return tm.db
}

func NewTestManager() *TestManager {
db, err := ethdb.NewMemDatabase()
if err != nil {
fmt.Println("Could not create mem-db, failing")
return nil
```

```
}

testManager := &TestManager{}
testManager.eventMux = new(event.TypeMux)
testManager.db = db
// testManager.txPool = NewTxPool(testManager)
// testManager.blockChain = NewBlockChain(testManager)
// testManager.stateManager = NewStateManager(testManager)

return testManager
}

86:F:\git\coin\ethereum\go-ethereum\core\mkalloc.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// +build none

/*

   The mkalloc tool creates the genesis allocation constants in genesis_alloc.go
   It outputs a const declaration that contains an RLP-encoded list of (address, balance) tuples.

       go run mkalloc.go genesis.json

*/
package main

import (
"encoding/json"
"fmt"
"math/big"
"os"
"sort"
"strconv"

"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/rlp"
)

type allocItem struct{ Addr, Balance *big.Int }

type allocList []allocItem
```

```
func (a allocList) Len() int           { return len(a) }
func (a allocList) Less(i, j int) bool { return a[i].Addr.Cmp(a[j].Addr) < 0 }
func (a allocList) Swap(i, j int)      { a[i], a[j] = a[j], a[i] }

func makelist(g *core.Genesis) allocList {
a := make(allocList, 0, len(g.Alloc))
for addr, account := range g.Alloc {
if len(account.Storage) > 0 || len(account.Code) > 0 || account.Nonce != 0 {
panic(fmt.Sprintf("can't encode account %x", addr))
}
a = append(a, allocItem{addr.Big(), account.Balance})
}
sort.Sort(a)
return a
}

func makealloc(g *core.Genesis) string {
a := makelist(g)
data, err := rlp.EncodeToBytes(a)
if err != nil {
panic(err)
}
return strconv.QuoteToASCII(string(data))
}

func main() {
if len(os.Args) != 2 {
fmt.Fprintln(os.Stderr, "Usage: mkalloc genesis.json")
os.Exit(1)
}

g := new(core.Genesis)
file, err := os.Open(os.Args[1])
if err != nil {
panic(err)
}
if err := json.NewDecoder(file).Decode(g); err != nil {
panic(err)
}
fmt.Println("const allocData =", makealloc(g))
}
```

87:F:\git\coin\ethereum\go-ethereum\core\state\database.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"fmt"
"sync"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/trie"
lru "github.com/hashicorp/golang-lru"
)

// Trie cache generation limit after which to evic trie nodes from memory.
var MaxTrieCacheGen = uint16(120)

const (
// Number of past tries to keep. This value is chosen such that
// reasonable chain reorg depths will hit an existing trie.
maxPastTries = 12

// Number of codehash->size associations to keep.
codeSizeCacheSize = 100000
)

// Database wraps access to tries and contract code.
type Database interface {
// Accessing tries:
// OpenTrie opens the main account trie.
// OpenStorageTrie opens the storage trie of an account.
OpenTrie(root common.Hash) (Trie, error)
OpenStorageTrie(addrHash, root common.Hash) (Trie, error)
// Accessing contract code:
ContractCode(addrHash, codeHash common.Hash) ([]byte, error)
ContractCodeSize(addrHash, codeHash common.Hash) (int, error)
// CopyTrie returns an independent copy of the given trie.
CopyTrie(Trie) Trie
}

```
// Trie is a Ethereum Merkle Trie.
type Trie interface {
TryGet(key []byte) ([]byte, error)
TryUpdate(key, value []byte) error
TryDelete(key []byte) error
CommitTo(trie.DatabaseWriter) (common.Hash, error)
Hash() common.Hash
NodeIterator(startKey []byte) trie.NodeIterator
GetKey([]byte) []byte // TODO(fjl): remove this when SecureTrie is removed
}

// NewDatabase creates a backing store for state. The returned database is safe for
// concurrent use and retains cached trie nodes in memory.
func NewDatabase(db ethdb.Database) Database {
csc, _ := lru.New(codeSizeCacheSize)
return &cachingDB{db: db, codeSizeCache: csc}
}

type cachingDB struct {
db            ethdb.Database
mu            sync.Mutex
pastTries     []*trie.SecureTrie
codeSizeCache *lru.Cache
}

func (db *cachingDB) OpenTrie(root common.Hash) (Trie, error) {
db.mu.Lock()
defer db.mu.Unlock()

for i := len(db.pastTries) - 1; i >= 0; i-- {
if db.pastTries[i].Hash() == root {
return cachedTrie{db.pastTries[i].Copy(), db}, nil
}
}
tr, err := trie.NewSecure(root, db.db, MaxTrieCacheGen)
if err != nil {
return nil, err
}
return cachedTrie{tr, db}, nil
}

func (db *cachingDB) pushTrie(t *trie.SecureTrie) {
```

```
db.mu.Lock()
defer db.mu.Unlock()

if len(db.pastTries) >= maxPastTries {
copy(db.pastTries, db.pastTries[1:])
db.pastTries[len(db.pastTries)-1] = t
} else {
db.pastTries = append(db.pastTries, t)
}
}

func (db *cachingDB) OpenStorageTrie(addrHash, root common.Hash) (Trie, error) {
return trie.NewSecure(root, db.db, 0)
}

func (db *cachingDB) CopyTrie(t Trie) Trie {
switch t := t.(type) {
case cachedTrie:
return cachedTrie{t.SecureTrie.Copy(), db}
case *trie.SecureTrie:
return t.Copy()
default:
panic(fmt.Errorf("unknown trie type %T", t))
}
}

func (db *cachingDB) ContractCode(addrHash, codeHash common.Hash) ([]byte, error) {
code, err := db.db.Get(codeHash[:])
if err == nil {
db.codeSizeCache.Add(codeHash, len(code))
}
return code, err
}

func (db *cachingDB) ContractCodeSize(addrHash, codeHash common.Hash) (int, error) {
if cached, ok := db.codeSizeCache.Get(codeHash); ok {
return cached.(int), nil
}
code, err := db.ContractCode(addrHash, codeHash)
if err == nil {
db.codeSizeCache.Add(codeHash, len(code))
}
```

```
return len(code), err
}

// cachedTrie inserts its trie into a cachingDB on commit.
type cachedTrie struct {
*trie.SecureTrie
db *cachingDB
}

func (m cachedTrie) CommitTo(dbw trie.DatabaseWriter) (common.Hash, error) {
root, err := m.SecureTrie.CommitTo(dbw)
if err == nil {
m.db.pushTrie(m.SecureTrie)
}
return root, err
}

88:F:\git\coin\ethereum\go-ethereum\core\state\dump.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"encoding/json"
"fmt"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/rlp"
"github.com/ethereum/go-ethereum/trie"
)

type DumpAccount struct {
Balance  string            `json:"balance"`
Nonce    uint64            `json:"nonce"`
Root     string            `json:"root"`
CodeHash string            `json:"codeHash"`
Code     string            `json:"code"`
Storage  map[string]string `json:"storage"`
}

type Dump struct {
Root     string                `json:"root"`
```

```
Accounts map[string]DumpAccount `json:"accounts"`
}

func (self *StateDB) RawDump() Dump {
dump := Dump{
Root:     fmt.Sprintf("%x", self.trie.Hash()),
Accounts: make(map[string]DumpAccount),
}

it := trie.NewIterator(self.trie.NodeIterator(nil))
for it.Next() {
addr := self.trie.GetKey(it.Key)
var data Account
if err := rlp.DecodeBytes(it.Value, &data); err != nil {
panic(err)
}

obj := newObject(nil, common.BytesToAddress(addr), data, nil)
account := DumpAccount{
Balance:  data.Balance.String(),
Nonce:    data.Nonce,
Root:     common.Bytes2Hex(data.Root[:]),
CodeHash: common.Bytes2Hex(data.CodeHash),
Code:     common.Bytes2Hex(obj.Code(self.db)),
Storage:  make(map[string]string),
}
storageIt := trie.NewIterator(obj.getTrie(self.db).NodeIterator(nil))
for storageIt.Next() {
account.Storage[common.Bytes2Hex(self.trie.GetKey(storageIt.Key))] =
common.Bytes2Hex(storageIt.Value)
}
dump.Accounts[common.Bytes2Hex(addr)] = account
}
return dump
}

func (self *StateDB) Dump() []byte {
json, err := json.MarshalIndent(self.RawDump(), "", "    ")
if err != nil {
fmt.Println("dump err", err)
}
```

```
return json
}

89:F:\git\coin\ethereum\go-ethereum\core\state\iterator.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"bytes"
"fmt"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/rlp"
"github.com/ethereum/go-ethereum/trie"
)

// NodeIterator is an iterator to traverse the entire state trie post-order,
// including all of the contract code and contract state tries.
type NodeIterator struct {
state *StateDB // State being iterated

stateIt trie.NodeIterator // Primary iterator for the global state trie
dataIt  trie.NodeIterator // Secondary iterator for the data trie of a contract

accountHash common.Hash // Hash of the node containing the account
codeHash    common.Hash // Hash of the contract source code
code        []byte      // Source code associated with a contract

Hash   common.Hash // Hash of the current entry being iterated (nil if not standalone)
Parent common.Hash // Hash of the first full ancestor node (nil if current is the root)

Error error // Failure set in case of an internal error in the iterator
}

// NewNodeIterator creates an post-order state node iterator.
func NewNodeIterator(state *StateDB) *NodeIterator {
return &NodeIterator{
state: state,
}
}
```

```
// Next moves the iterator to the next node, returning whether there are any
// further nodes. In case of an internal error this method returns false and
// sets the Error field to the encountered failure.
func (it *NodeIterator) Next() bool {
// If the iterator failed previously, don't do anything
if it.Error != nil {
return false
}
// Otherwise step forward with the iterator and report any errors
if err := it.step(); err != nil {
it.Error = err
return false
}
return it.retrieve()
}

// step moves the iterator to the next entry of the state trie.
func (it *NodeIterator) step() error {
// Abort if we reached the end of the iteration
if it.state == nil {
return nil
}
// Initialize the iterator if we've just started
if it.stateIt == nil {
it.stateIt = it.state.trie.NodeIterator(nil)
}
// If we had data nodes previously, we surely have at least state nodes
if it.dataIt != nil {
if cont := it.dataIt.Next(true); !cont {
if it.dataIt.Error() != nil {
return it.dataIt.Error()
}
it.dataIt = nil
}
return nil
}
// If we had source code previously, discard that
if it.code != nil {
it.code = nil
return nil
}
// Step to the next state trie node, terminating if we're out of nodes
```

```
if cont := it.stateIt.Next(true); !cont {
if it.stateIt.Error() != nil {
return it.stateIt.Error()
}
it.state, it.stateIt = nil, nil
return nil
}
// If the state trie node is an internal entry, leave as is
if !it.stateIt.Leaf() {
return nil
}
// Otherwise we've reached an account node, initiate data iteration
var account Account
if err := rlp.Decode(bytes.NewReader(it.stateIt.LeafBlob()), &account); err != nil {
return err
}
dataTrie, err := it.state.db.OpenStorageTrie(common.BytesToHash(it.stateIt.LeafKey()),
account.Root)
if err != nil {
return err
}
it.dataIt = dataTrie.NodeIterator(nil)
if !it.dataIt.Next(true) {
it.dataIt = nil
}
if !bytes.Equal(account.CodeHash, emptyCodeHash) {
it.codeHash = common.BytesToHash(account.CodeHash)
addrHash := common.BytesToHash(it.stateIt.LeafKey())
it.code, err = it.state.db.ContractCode(addrHash, common.BytesToHash(account.CodeHash))
if err != nil {
return fmt.Errorf("code %x: %v", account.CodeHash, err)
}
}
it.accountHash = it.stateIt.Parent()
return nil
}

// retrieve pulls and caches the current state entry the iterator is traversing.
// The method returns whether there are any more data left for inspection.
func (it *NodeIterator) retrieve() bool {
// Clear out any previously set values
it.Hash = common.Hash{}
```

```
// If the iteration's done, return no available data
if it.state == nil {
return false
}
// Otherwise retrieve the current entry
switch {
case it.dataIt != nil:
it.Hash, it.Parent = it.dataIt.Hash(), it.dataIt.Parent()
if it.Parent == (common.Hash{}) {
it.Parent = it.accountHash
}
case it.code != nil:
it.Hash, it.Parent = it.codeHash, it.accountHash
case it.stateIt != nil:
it.Hash, it.Parent = it.stateIt.Hash(), it.stateIt.Parent()
}
return true
}
```

90:F:\git\coin\ethereum\go-ethereum\core\state\iterator_test.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"bytes"
"testing"

"github.com/ethereum/go-ethereum/common"
)

// Tests that the node iterator indeed walks over the entire database contents.
func TestNodeIteratorCoverage(t *testing.T) {
// Create some arbitrary test state to iterate
db, mem, root, _ := makeTestState()

state, err := New(root, db)
if err != nil {
t.Fatalf("failed to create state trie at %x: %v", root, err)
}
// Gather all the node hashes found by the iterator
```

```
hashes := make(map[common.Hash]struct{})
for it := NewNodeIterator(state); it.Next(); {
if it.Hash != (common.Hash{}) {
hashes[it.Hash] = struct{}{}
}
}

// Cross check the hashes and the database itself
for hash := range hashes {
if _, err := mem.Get(hash.Bytes()); err != nil {
t.Errorf("failed to retrieve reported node %x: %v", hash, err)
}
}
for _, key := range mem.Keys() {
if bytes.HasPrefix(key, []byte("secure-key-")) {
continue
}
if _, ok := hashes[common.BytesToHash(key)]; !ok {
t.Errorf("state entry not reported %x", key)
}
}
}

91:F:\git\coin\ethereum\go-ethereum\core\state\journal.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"math/big"

"github.com/ethereum/go-ethereum/common"
)

type journalEntry interface {
undo(*StateDB)
}

type journal []journalEntry

type (
// Changes to the account trie.
```

```
createObjectChange struct {
account *common.Address
}
resetObjectChange struct {
prev *stateObject
}
suicideChange struct {
account     *common.Address
prev        bool // whether account had already suicided
prevbalance *big.Int
}

// Changes to individual accounts.
balanceChange struct {
account *common.Address
prev    *big.Int
}
nonceChange struct {
account *common.Address
prev    uint64
}
storageChange struct {
account       *common.Address
key, prevalue common.Hash
}
codeChange struct {
account            *common.Address
prevcode, prevhash []byte
}

// Changes to other state values.
refundChange struct {
prev *big.Int
}
addLogChange struct {
txhash common.Hash
}
addPreimageChange struct {
hash common.Hash
}
touchChange struct {
account   *common.Address
```

```
prev      bool
prevDirty bool
}
)

func (ch createObjectChange) undo(s *StateDB) {
delete(s.stateObjects, *ch.account)
delete(s.stateObjectsDirty, *ch.account)
}

func (ch resetObjectChange) undo(s *StateDB) {
s.setStateObject(ch.prev)
}

func (ch suicideChange) undo(s *StateDB) {
obj := s.getStateObject(*ch.account)
if obj != nil {
obj.suicided = ch.prev
obj.setBalance(ch.prevbalance)
// if the object wasn't suicided before, remove
// it from the list of destructed objects as well.
if !obj.suicided {
delete(s.stateObjectsDestructed, *ch.account)
}
}
}

var ripemd = common.HexToAddress("0000000000000000000000000000000000000003")

func (ch touchChange) undo(s *StateDB) {
if !ch.prev && *ch.account != ripemd {
s.getStateObject(*ch.account).touched = ch.prev
if !ch.prevDirty {
delete(s.stateObjectsDirty, *ch.account)
}
}
}

func (ch balanceChange) undo(s *StateDB) {
s.getStateObject(*ch.account).setBalance(ch.prev)
}
```

```
func (ch nonceChange) undo(s *StateDB) {
s.getStateObject(*ch.account).setNonce(ch.prev)
}

func (ch codeChange) undo(s *StateDB) {
s.getStateObject(*ch.account).setCode(common.BytesToHash(ch.prevhash), ch.prevcode)
}

func (ch storageChange) undo(s *StateDB) {
s.getStateObject(*ch.account).setState(ch.key, ch.prevalue)
}

func (ch refundChange) undo(s *StateDB) {
s.refund = ch.prev
}

func (ch addLogChange) undo(s *StateDB) {
logs := s.logs[ch.txhash]
if len(logs) == 1 {
delete(s.logs, ch.txhash)
} else {
s.logs[ch.txhash] = logs[:len(logs)-1]
}
}

func (ch addPreimageChange) undo(s *StateDB) {
delete(s.preimages, ch.hash)
}

92:F:\git\coin\ethereum\go-ethereum\core\state\main_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"testing"

checker "gopkg.in/check.v1"
)

func Test(t *testing.T) { checker.TestingT(t) }
```

93:F:\git\coin\ethereum\go-ethereum\core\state\managed_state.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"sync"

"github.com/ethereum/go-ethereum/common"
)

type account struct {
stateObject *stateObject
nstart   uint64
nonces   []bool
}

type ManagedState struct {
*StateDB

mu sync.RWMutex

accounts map[common.Address]*account
}

// ManagedState returns a new managed state with the statedb as it's backing layer
func ManageState(statedb *StateDB) *ManagedState {
return &ManagedState{
StateDB: statedb.Copy(),
accounts: make(map[common.Address]*account),
}
}

// SetState sets the backing layer of the managed state
func (ms *ManagedState) SetState(statedb *StateDB) {
ms.mu.Lock()
defer ms.mu.Unlock()
ms.StateDB = statedb
}

// RemoveNonce removed the nonce from the managed state and all future pending nonces
func (ms *ManagedState) RemoveNonce(addr common.Address, n uint64) {

```
if ms.hasAccount(addr) {
ms.mu.Lock()
defer ms.mu.Unlock()

account := ms.getAccount(addr)
if n-account.nstart <= uint64(len(account.nonces)) {
reslice := make([]bool, n-account.nstart)
copy(reslice, account.nonces[:n-account.nstart])
account.nonces = reslice
}
}
}

// NewNonce returns the new canonical nonce for the managed account
func (ms *ManagedState) NewNonce(addr common.Address) uint64 {
ms.mu.Lock()
defer ms.mu.Unlock()

account := ms.getAccount(addr)
for i, nonce := range account.nonces {
if !nonce {
return account.nstart + uint64(i)
}
}
account.nonces = append(account.nonces, true)

return uint64(len(account.nonces)-1) + account.nstart
}

// GetNonce returns the canonical nonce for the managed or unmanaged account.
//
// Because GetNonce mutates the DB, we must take a write lock.
func (ms *ManagedState) GetNonce(addr common.Address) uint64 {
ms.mu.Lock()
defer ms.mu.Unlock()

if ms.hasAccount(addr) {
account := ms.getAccount(addr)
return uint64(len(account.nonces)) + account.nstart
} else {
return ms.StateDB.GetNonce(addr)
}
```

```
}

// SetNonce sets the new canonical nonce for the managed state
func (ms *ManagedState) SetNonce(addr common.Address, nonce uint64) {
ms.mu.Lock()
defer ms.mu.Unlock()

so := ms.GetOrNewStateObject(addr)
so.SetNonce(nonce)

ms.accounts[addr] = newAccount(so)
}

// HasAccount returns whether the given address is managed or not
func (ms *ManagedState) HasAccount(addr common.Address) bool {
ms.mu.RLock()
defer ms.mu.RUnlock()
return ms.hasAccount(addr)
}

func (ms *ManagedState) hasAccount(addr common.Address) bool {
_, ok := ms.accounts[addr]
return ok
}

// populate the managed state
func (ms *ManagedState) getAccount(addr common.Address) *account {
if account, ok := ms.accounts[addr]; !ok {
so := ms.GetOrNewStateObject(addr)
ms.accounts[addr] = newAccount(so)
} else {
// Always make sure the state account nonce isn't actually higher
// than the tracked one.
so := ms.StateDB.getStateObject(addr)
if so != nil && uint64(len(account.nonces))+account.nstart < so.Nonce() {
ms.accounts[addr] = newAccount(so)
}

}

return ms.accounts[addr]
}
```

```
func newAccount(so *stateObject) *account {
return &account{so, so.Nonce(), nil}
}

94:F:\git\coin\ethereum\go-ethereum\core\state\managed_state_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/ethdb"
)

var addr = common.BytesToAddress([]byte("test"))

func create() (*ManagedState, *account) {
db, _ := ethdb.NewMemDatabase()
statedb, _ := New(common.Hash{}, NewDatabase(db))
ms := ManageState(statedb)
ms.StateDB.SetNonce(addr, 100)
ms.accounts[addr] = newAccount(ms.StateDB.getStateObject(addr))
return ms, ms.accounts[addr]
}

func TestNewNonce(t *testing.T) {
ms, _ := create()

nonce := ms.NewNonce(addr)
if nonce != 100 {
t.Error("expected nonce 100. got", nonce)
}

nonce = ms.NewNonce(addr)
if nonce != 101 {
t.Error("expected nonce 101. got", nonce)
}
}
```

```
func TestRemove(t *testing.T) {
ms, account := create()

nn := make([]bool, 10)
for i := range nn {
nn[i] = true
}
account.nonces = append(account.nonces, nn...)

i := uint64(5)
ms.RemoveNonce(addr, account.nstart+i)
if len(account.nonces) != 5 {
t.Error("expected", i, "'th index to be false")
}
}

func TestReuse(t *testing.T) {
ms, account := create()

nn := make([]bool, 10)
for i := range nn {
nn[i] = true
}
account.nonces = append(account.nonces, nn...)

i := uint64(5)
ms.RemoveNonce(addr, account.nstart+i)
nonce := ms.NewNonce(addr)
if nonce != 105 {
t.Error("expected nonce to be 105. got", nonce)
}
}

func TestRemoteNonceChange(t *testing.T) {
ms, account := create()
nn := make([]bool, 10)
for i := range nn {
nn[i] = true
}
account.nonces = append(account.nonces, nn...)
ms.NewNonce(addr)
```

```
ms.StateDB.stateObjects[addr].data.Nonce = 200
nonce := ms.NewNonce(addr)
if nonce != 200 {
t.Error("expected nonce after remote update to be", 200, "got", nonce)
}
ms.NewNonce(addr)
ms.NewNonce(addr)
ms.NewNonce(addr)
ms.StateDB.stateObjects[addr].data.Nonce = 200
nonce = ms.NewNonce(addr)
if nonce != 204 {
t.Error("expected nonce after remote update to be", 204, "got", nonce)
}
}

func TestSetNonce(t *testing.T) {
ms, _ := create()

var addr common.Address
ms.SetNonce(addr, 10)

if ms.GetNonce(addr) != 10 {
t.Error("Expected nonce of 10, got", ms.GetNonce(addr))
}

addr[0] = 1
ms.StateDB.SetNonce(addr, 1)

if ms.GetNonce(addr) != 1 {
t.Error("Expected nonce of 1, got", ms.GetNonce(addr))
}
}

95:F:\git\coin\ethereum\go-ethereum\core\state\statedb.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package state provides a caching layer atop the Ethereum state trie.
package state

import (
"fmt"
"math/big"
```

```
"sort"
"sync"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/rlp"
"github.com/ethereum/go-ethereum/trie"
)

type revision struct {
id           int
journalIndex int
}

// StateDBs within the ethereum protocol are used to store anything
// within the merkle trie. StateDBs take care of caching and storing
// nested states. It's the general query interface to retrieve:
// * Contracts
// * Accounts
type StateDB struct {
db   Database
trie Trie

// This map holds 'live' objects, which will get modified while processing a state transition.
stateObjects          map[common.Address]*stateObject
stateObjectsDirty     map[common.Address]struct{}
stateObjectsDestructed map[common.Address]struct{}

// DB error.
// State objects are used by the consensus core and VM which are
// unable to deal with database-level errors. Any error that occurs
// during a database read is memoized here and will eventually be returned
// by StateDB.Commit.
dbErr error

// The refund counter, also used by state transitioning.
refund *big.Int

thash, bhash common.Hash
txIndex      int
```

```
logs         map[common.Hash][]*types.Log
logSize      uint

preimages map[common.Hash][]byte

// Journal of state modifications. This is the backbone of
// Snapshot and RevertToSnapshot.
journal        journal
validRevisions []revision
nextRevisionId int

lock sync.Mutex
}

// Create a new state from a given trie
func New(root common.Hash, db Database) (*StateDB, error) {
tr, err := db.OpenTrie(root)
if err != nil {
return nil, err
}
return &StateDB{
db:                   db,
trie:                 tr,
stateObjects:         make(map[common.Address]*stateObject),
stateObjectsDirty:    make(map[common.Address]struct{}),
stateObjectsDestructed: make(map[common.Address]struct{}),
refund:               new(big.Int),
logs:                 make(map[common.Hash][]*types.Log),
preimages:            make(map[common.Hash][]byte),
}, nil
}

// setError remembers the first non-nil error it is called with.
func (self *StateDB) setError(err error) {
if self.dbErr == nil {
self.dbErr = err
}
}

func (self *StateDB) Error() error {
return self.dbErr
}
```

```
// Reset clears out all emphemeral state objects from the state db, but keeps
// the underlying state trie to avoid reloading data for the next operations.
func (self *StateDB) Reset(root common.Hash) error {
tr, err := self.db.OpenTrie(root)
if err != nil {
return err
}
self.trie = tr
self.stateObjects = make(map[common.Address]*stateObject)
self.stateObjectsDirty = make(map[common.Address]struct{})
self.stateObjectsDestructed = make(map[common.Address]struct{})
self.thash = common.Hash{}
self.bhash = common.Hash{}
self.txIndex = 0
self.logs = make(map[common.Hash][]*types.Log)
self.logSize = 0
self.preimages = make(map[common.Hash][]byte)
self.clearJournalAndRefund()
return nil
}

func (self *StateDB) AddLog(log *types.Log) {
self.journal = append(self.journal, addLogChange{txhash: self.thash})

log.TxHash = self.thash
log.BlockHash = self.bhash
log.TxIndex = uint(self.txIndex)
log.Index = self.logSize
self.logs[self.thash] = append(self.logs[self.thash], log)
self.logSize++
}

func (self *StateDB) GetLogs(hash common.Hash) []*types.Log {
return self.logs[hash]
}

func (self *StateDB) Logs() []*types.Log {
var logs []*types.Log
for _, lgs := range self.logs {
logs = append(logs, lgs...)
}
```

```
return logs
}

// AddPreimage records a SHA3 preimage seen by the VM.
func (self *StateDB) AddPreimage(hash common.Hash, preimage []byte) {
if _, ok := self.preimages[hash]; !ok {
self.journal = append(self.journal, addPreimageChange{hash: hash})
pi := make([]byte, len(preimage))
copy(pi, preimage)
self.preimages[hash] = pi
}
}

// Preimages returns a list of SHA3 preimages that have been submitted.
func (self *StateDB) Preimages() map[common.Hash][]byte {
return self.preimages
}

func (self *StateDB) AddRefund(gas *big.Int) {
self.journal = append(self.journal, refundChange{prev: new(big.Int).Set(self.refund)})
self.refund.Add(self.refund, gas)
}

// Exist reports whether the given account address exists in the state.
// Notably this also returns true for suicided accounts.
func (self *StateDB) Exist(addr common.Address) bool {
return self.getStateObject(addr) != nil
}

// Empty returns whether the state object is either non-existent
// or empty according to the EIP161 specification (balance = nonce = code = 0)
func (self *StateDB) Empty(addr common.Address) bool {
so := self.getStateObject(addr)
return so == nil || so.empty()
}

// Retrieve the balance from the given address or 0 if object not found
func (self *StateDB) GetBalance(addr common.Address) *big.Int {
stateObject := self.getStateObject(addr)
if stateObject != nil {
return stateObject.Balance()
}
```

```
return common.Big0
}

func (self *StateDB) GetNonce(addr common.Address) uint64 {
stateObject := self.getStateObject(addr)
if stateObject != nil {
return stateObject.Nonce()
}

return 0
}

func (self *StateDB) GetCode(addr common.Address) []byte {
stateObject := self.getStateObject(addr)
if stateObject != nil {
return stateObject.Code(self.db)
}
return nil
}

func (self *StateDB) GetCodeSize(addr common.Address) int {
stateObject := self.getStateObject(addr)
if stateObject == nil {
return 0
}
if stateObject.code != nil {
return len(stateObject.code)
}
size, err := self.db.ContractCodeSize(stateObject.addrHash,
common.BytesToHash(stateObject.CodeHash()))
if err != nil {
self.setError(err)
}
return size
}

func (self *StateDB) GetCodeHash(addr common.Address) common.Hash {
stateObject := self.getStateObject(addr)
if stateObject == nil {
return common.Hash{}
}
return common.BytesToHash(stateObject.CodeHash())
```

```
}

func (self *StateDB) GetState(a common.Address, b common.Hash) common.Hash {
stateObject := self.getStateObject(a)
if stateObject != nil {
return stateObject.GetState(self.db, b)
}
return common.Hash{}
}

// StorageTrie returns the storage trie of an account.
// The return value is a copy and is nil for non-existent accounts.
func (self *StateDB) StorageTrie(a common.Address) Trie {
stateObject := self.getStateObject(a)
if stateObject == nil {
return nil
}
cpy := stateObject.deepCopy(self, nil)
return cpy.updateTrie(self.db)
}

func (self *StateDB) HasSuicided(addr common.Address) bool {
stateObject := self.getStateObject(addr)
if stateObject != nil {
return stateObject.suicided
}
return false
}

/*
 * SETTERS
 */

// AddBalance adds amount to the account associated with addr
func (self *StateDB) AddBalance(addr common.Address, amount *big.Int) {
stateObject := self.GetOrNewStateObject(addr)
if stateObject != nil {
stateObject.AddBalance(amount)
}
}

// SubBalance subtracts amount from the account associated with addr
```

```
func (self *StateDB) SubBalance(addr common.Address, amount *big.Int) {
stateObject := self.GetOrNewStateObject(addr)
if stateObject != nil {
stateObject.SubBalance(amount)
}
}

func (self *StateDB) SetBalance(addr common.Address, amount *big.Int) {
stateObject := self.GetOrNewStateObject(addr)
if stateObject != nil {
stateObject.SetBalance(amount)
}
}

func (self *StateDB) SetNonce(addr common.Address, nonce uint64) {
stateObject := self.GetOrNewStateObject(addr)
if stateObject != nil {
stateObject.SetNonce(nonce)
}
}

func (self *StateDB) SetCode(addr common.Address, code []byte) {
stateObject := self.GetOrNewStateObject(addr)
if stateObject != nil {
stateObject.SetCode(crypto.Keccak256Hash(code), code)
}
}

func (self *StateDB) SetState(addr common.Address, key common.Hash, value common.Hash) {
stateObject := self.GetOrNewStateObject(addr)
if stateObject != nil {
stateObject.SetState(self.db, key, value)
}
}

// Suicide marks the given account as suicided.
// This clears the account balance.
//
// The account's state object is still available until the state is committed,
// getStateObject will return a non-nil account after Suicide.
func (self *StateDB) Suicide(addr common.Address) bool {
stateObject := self.getStateObject(addr)
```

```
if stateObject == nil {
return false
}
self.journal = append(self.journal, suicideChange{
account:     &addr,
prev:        stateObject.suicided,
prevbalance: new(big.Int).Set(stateObject.Balance()),
})
stateObject.markSuicided()
stateObject.data.Balance = new(big.Int)
self.stateObjectsDestructed[addr] = struct{}{}

return true
}

//
// Setting, updating & deleting state object methods
//

// updateStateObject writes the given object to the trie.
func (self *StateDB) updateStateObject(stateObject *stateObject) {
addr := stateObject.Address()
data, err := rlp.EncodeToBytes(stateObject)
if err != nil {
panic(fmt.Errorf("can't encode object at %x: %v", addr[:], err))
}
self.setError(self.trie.TryUpdate(addr[:], data))
}

// deleteStateObject removes the given object from the state trie.
func (self *StateDB) deleteStateObject(stateObject *stateObject) {
stateObject.deleted = true
addr := stateObject.Address()
self.setError(self.trie.TryDelete(addr[:]))
}

// Retrieve a state object given my the address. Returns nil if not found.
func (self *StateDB) getStateObject(addr common.Address) (stateObject *stateObject) {
// Prefer 'live' objects.
if obj := self.stateObjects[addr]; obj != nil {
if obj.deleted {
return nil
```

```
}
return obj
}

// Load the object from the database.
enc, err := self.trie.TryGet(addr[:])
if len(enc) == 0 {
self.setError(err)
return nil
}
var data Account
if err := rlp.DecodeBytes(enc, &data); err != nil {
log.Error("Failed to decode state object", "addr", addr, "err", err)
return nil
}
// Insert into the live set.
obj := newObject(self, addr, data, self.MarkStateObjectDirty)
self.setStateObject(obj)
return obj
}

func (self *StateDB) setStateObject(object *stateObject) {
self.stateObjects[object.Address()] = object
}

// Retrieve a state object or create a new state object if nil
func (self *StateDB) GetOrNewStateObject(addr common.Address) *stateObject {
stateObject := self.getStateObject(addr)
if stateObject == nil || stateObject.deleted {
stateObject, _ = self.createObject(addr)
}
return stateObject
}

// MarkStateObjectDirty adds the specified object to the dirty map to avoid costly
// state object cache iteration to find a handful of modified ones.
func (self *StateDB) MarkStateObjectDirty(addr common.Address) {
self.stateObjectsDirty[addr] = struct{}{}
}

// createObject creates a new state object. If there is an existing account with
// the given address, it is overwritten and returned as the second return value.
```

```
func (self *StateDB) createObject(addr common.Address) (newobj, prev *stateObject) {
prev = self.getStateObject(addr)
newobj = newObject(self, addr, Account{}, self.MarkStateObjectDirty)
newobj.setNonce(0) // sets the object to dirty
if prev == nil {
self.journal = append(self.journal, createObjectChange{account: &addr})
} else {
self.journal = append(self.journal, resetObjectChange{prev: prev})
}
self.setStateObject(newobj)
return newobj, prev
}

// CreateAccount explicitly creates a state object. If a state object with the address
// already exists the balance is carried over to the new account.
//
// CreateAccount is called during the EVM CREATE operation. The situation might arise that
// a contract does the following:
//
//   1. sends funds to sha(account ++ (nonce + 1))
//   2. tx_create(sha(account ++ nonce)) (note that this gets the address of 1)
//
// Carrying over the balance ensures that Ether doesn't disappear.
func (self *StateDB) CreateAccount(addr common.Address) {
new, prev := self.createObject(addr)
if prev != nil {
new.setBalance(prev.data.Balance)
}
}

func (db *StateDB) ForEachStorage(addr common.Address, cb func(key, value common.Hash)
bool) {
so := db.getStateObject(addr)
if so == nil {
return
}

// When iterating over the storage check the cache first
for h, value := range so.cachedStorage {
cb(h, value)
}
```

```
it := trie.NewIterator(so.getTrie(db.db).NodeIterator(nil))
for it.Next() {
// ignore cached values
key := common.BytesToHash(db.trie.GetKey(it.Key))
if _, ok := so.cachedStorage[key]; !ok {
cb(key, common.BytesToHash(it.Value))
}
}
}

// Copy creates a deep, independent copy of the state.
// Snapshots of the copied state cannot be applied to the copy.
func (self *StateDB) Copy() *StateDB {
self.lock.Lock()
defer self.lock.Unlock()

// Copy all the basic fields, initialize the memory ones
state := &StateDB{
db:                  self.db,
trie:                self.trie,
stateObjects:        make(map[common.Address]*stateObject, len(self.stateObjectsDirty)),
stateObjectsDirty:   make(map[common.Address]struct{}, len(self.stateObjectsDirty)),
stateObjectsDestructed: make(map[common.Address]struct{}, len(self.stateObjectsDestructed)),
refund:              new(big.Int).Set(self.refund),
logs:                make(map[common.Hash][]*types.Log, len(self.logs)),
logSize:             self.logSize,
preimages:           make(map[common.Hash][]byte),
}
// Copy the dirty states, logs, and preimages
for addr := range self.stateObjectsDirty {
state.stateObjects[addr] = self.stateObjects[addr].deepCopy(state, state.MarkStateObjectDirty)
state.stateObjectsDirty[addr] = struct{}{}
if self.stateObjects[addr].suicided {
state.stateObjectsDestructed[addr] = struct{}{}
}
}
for hash, logs := range self.logs {
state.logs[hash] = make([]*types.Log, len(logs))
copy(state.logs[hash], logs)
}
for hash, preimage := range self.preimages {
state.preimages[hash] = preimage
```

```
}
return state
}

// Snapshot returns an identifier for the current revision of the state.
func (self *StateDB) Snapshot() int {
id := self.nextRevisionId
self.nextRevisionId++
self.validRevisions = append(self.validRevisions, revision{id, len(self.journal)})
return id
}

// RevertToSnapshot reverts all state changes made since the given revision.
func (self *StateDB) RevertToSnapshot(revid int) {
// Find the snapshot in the stack of valid snapshots.
idx := sort.Search(len(self.validRevisions), func(i int) bool {
return self.validRevisions[i].id >= revid
})
if idx == len(self.validRevisions) || self.validRevisions[idx].id != revid {
panic(fmt.Errorf("revision id %v cannot be reverted", revid))
}
snapshot := self.validRevisions[idx].journalIndex

// Replay the journal to undo changes.
for i := len(self.journal) - 1; i >= snapshot; i-- {
self.journal[i].undo(self)
}
self.journal = self.journal[:snapshot]

// Remove invalidated snapshots from the stack.
self.validRevisions = self.validRevisions[:idx]
}

// GetRefund returns the current value of the refund counter.
// The return value must not be modified by the caller and will become
// invalid at the next call to AddRefund.
func (self *StateDB) GetRefund() *big.Int {
return self.refund
}

// IntermediateRoot computes the current root hash of the state trie.
// It is called in between transactions to get the root hash that
```

```
// goes into transaction receipts.
func (s *StateDB) IntermediateRoot(deleteEmptyObjects bool) common.Hash {
for addr := range s.stateObjectsDirty {
stateObject := s.stateObjects[addr]
if stateObject.suicided || (deleteEmptyObjects && stateObject.empty()) {
s.deleteStateObject(stateObject)
} else {
stateObject.updateRoot(s.db)
s.updateStateObject(stateObject)
}
}
// Invalidate journal because reverting across transactions is not allowed.
s.clearJournalAndRefund()
return s.trie.Hash()
}

// Prepare sets the current transaction hash and index and block hash which is
// used when the EVM emits new state logs.
func (self *StateDB) Prepare(thash, bhash common.Hash, ti int) {
self.thash = thash
self.bhash = bhash
self.txIndex = ti
}

// Finalise finalises the state by removing the self destructed objects
// in the current stateObjectsDestructed buffer and clears the journal
// as well as the refunds.
//
// Please note that Finalise is used by EIP#98 and is used instead of
// IntermediateRoot.
func (s *StateDB) Finalise() {
for addr := range s.stateObjectsDestructed {
s.deleteStateObject(s.stateObjects[addr])
}
s.clearJournalAndRefund()
}

// DeleteSuicides flags the suicided objects for deletion so that it
// won't be referenced again when called / queried up on.
//
// DeleteSuicides should not be used for consensus related updates
// under any circumstances.
```

```
func (s *StateDB) DeleteSuicides() {
// Reset refund so that any used-gas calculations can use this method.
s.clearJournalAndRefund()

for addr := range s.stateObjectsDirty {
stateObject := s.stateObjects[addr]

// If the object has been removed by a suicide
// flag the object as deleted.
if stateObject.suicided {
stateObject.deleted = true
}
delete(s.stateObjectsDirty, addr)
}
}

func (s *StateDB) clearJournalAndRefund() {
s.journal = nil
s.validRevisions = s.validRevisions[:0]
s.refund = new(big.Int)
}

// CommitTo writes the state to the given database.
func (s *StateDB) CommitTo(dbw trie.DatabaseWriter, deleteEmptyObjects bool) (root
common.Hash, err error) {
defer s.clearJournalAndRefund()

// Commit objects to the trie.
for addr, stateObject := range s.stateObjects {
_, isDirty := s.stateObjectsDirty[addr]
switch {
case stateObject.suicided || (isDirty && deleteEmptyObjects && stateObject.empty()):
// If the object has been removed, don't bother syncing it
// and just mark it for deletion in the trie.
s.deleteStateObject(stateObject)
case isDirty:
// Write any contract code associated with the state object
if stateObject.code != nil && stateObject.dirtyCode {
if err := dbw.Put(stateObject.CodeHash(), stateObject.code); err != nil {
return common.Hash{}, err
}
stateObject.dirtyCode = false
```

```
}
// Write any storage changes in the state object to its storage trie.
if err := stateObject.CommitTrie(s.db, dbw); err != nil {
return common.Hash{}, err
}
// Update the object in the main account trie.
s.updateStateObject(stateObject)
}
delete(s.stateObjectsDirty, addr)
}
// Write trie changes.
root, err = s.trie.CommitTo(dbw)
log.Debug("Trie cache stats after commit", "misses", trie.CacheMisses(), "unloads",
trie.CacheUnloads())
return root, err
}
```

96:F:\git\coin\ethereum\go-ethereum\core\state\statedb_test.go

```
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"bytes"
"encoding/binary"
"fmt"
"math"
"math/big"
"math/rand"
"reflect"
"strings"
"testing"
"testing/quick"

check "gopkg.in/check.v1"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/ethdb"
)

// Tests that updating a state trie does not leak any database writes prior to
```

```
// actually committing the state.
func TestUpdateLeaks(t *testing.T) {
// Create an empty state database
db, _ := ethdb.NewMemDatabase()
state, _ := New(common.Hash{}, NewDatabase(db))

// Update it with some accounts
for i := byte(0); i < 255; i++ {
addr := common.BytesToAddress([]byte{i})
state.AddBalance(addr, big.NewInt(int64(11*i)))
state.SetNonce(addr, uint64(42*i))
if i%2 == 0 {
state.SetState(addr, common.BytesToHash([]byte{i, i, i}), common.BytesToHash([]byte{i, i, i, i}))
}
if i%3 == 0 {
state.SetCode(addr, []byte{i, i, i, i, i})
}
state.IntermediateRoot(false)
}
// Ensure that no data was leaked into the database
for _, key := range db.Keys() {
value, _ := db.Get(key)
t.Errorf("State leaked into database: %x -> %x", key, value)
}
}

// Tests that no intermediate state of an object is stored into the database,
// only the one right before the commit.
func TestIntermediateLeaks(t *testing.T) {
// Create two state databases, one transitioning to the final state, the other final from the beginning
transDb, _ := ethdb.NewMemDatabase()
finalDb, _ := ethdb.NewMemDatabase()
transState, _ := New(common.Hash{}, NewDatabase(transDb))
finalState, _ := New(common.Hash{}, NewDatabase(finalDb))

modify := func(state *StateDB, addr common.Address, i, tweak byte) {
state.SetBalance(addr, big.NewInt(int64(11*i)+int64(tweak)))
state.SetNonce(addr, uint64(42*i+tweak))
if i%2 == 0 {
state.SetState(addr, common.Hash{i, i, i, 0}, common.Hash{})
state.SetState(addr, common.Hash{i, i, i, tweak}, common.Hash{i, i, i, i, tweak})
}
```

```
if i%3 == 0 {
state.SetCode(addr, []byte{i, i, i, i, i, tweak})
}
}

// Modify the transient state.
for i := byte(0); i < 255; i++ {
modify(transState, common.Address{byte(i)}, i, 0)
}
// Write modifications to trie.
transState.IntermediateRoot(false)

// Overwrite all the data with new values in the transient database.
for i := byte(0); i < 255; i++ {
modify(transState, common.Address{byte(i)}, i, 99)
modify(finalState, common.Address{byte(i)}, i, 99)
}

// Commit and cross check the databases.
if _, err := transState.CommitTo(transDb, false); err != nil {
t.Fatalf("failed to commit transition state: %v", err)
}
if _, err := finalState.CommitTo(finalDb, false); err != nil {
t.Fatalf("failed to commit final state: %v", err)
}
for _, key := range finalDb.Keys() {
if _, err := transDb.Get(key); err != nil {
val, _ := finalDb.Get(key)
t.Errorf("entry missing from the transition database: %x -> %x", key, val)
}
}
for _, key := range transDb.Keys() {
if _, err := finalDb.Get(key); err != nil {
val, _ := transDb.Get(key)
t.Errorf("extra entry in the transition database: %x -> %x", key, val)
}
}
}

func TestSnapshotRandom(t *testing.T) {
config := &quick.Config{MaxCount: 1000}
err := quick.Check((*snapshotTest).run, config)
```

```
if cerr, ok := err.(*quick.CheckError); ok {
test := cerr.In[0].(*snapshotTest)
t.Errorf("%v:\n%s", test.err, test)
} else if err != nil {
t.Error(err)
}
}

// A snapshotTest checks that reverting StateDB snapshots properly undoes all changes
// captured by the snapshot. Instances of this test with pseudorandom content are created
// by Generate.
//
// The test works as follows:
//
// A new state is created and all actions are applied to it. Several snapshots are taken
// in between actions. The test then reverts each snapshot. For each snapshot the actions
// leading up to it are replayed on a fresh, empty state. The behaviour of all public
// accessor methods on the reverted state must match the return value of the equivalent
// methods on the replayed state.
type snapshotTest struct {
addrs     []common.Address // all account addresses
actions   []testAction     // modifications to the state
snapshots []int            // actions indexes at which snapshot is taken
err       error            // failure details are reported through this field
}

type testAction struct {
name   string
fn     func(testAction, *StateDB)
args   []int64
noAddr bool
}

// newTestAction creates a random action that changes state.
func newTestAction(addr common.Address, r *rand.Rand) testAction {
actions := []testAction{
{
name: "SetBalance",
fn: func(a testAction, s *StateDB) {
s.SetBalance(addr, big.NewInt(a.args[0]))
},
args: make([]int64, 1),
```

```
},
{
name: "AddBalance",
fn: func(a testAction, s *StateDB) {
s.AddBalance(addr, big.NewInt(a.args[0]))
},
args: make([]int64, 1),
},
{
name: "SetNonce",
fn: func(a testAction, s *StateDB) {
s.SetNonce(addr, uint64(a.args[0]))
},
args: make([]int64, 1),
},
{
name: "SetState",
fn: func(a testAction, s *StateDB) {
var key, val common.Hash
binary.BigEndian.PutUint16(key[:], uint16(a.args[0]))
binary.BigEndian.PutUint16(val[:], uint16(a.args[1]))
s.SetState(addr, key, val)
},
args: make([]int64, 2),
},
{
name: "SetCode",
fn: func(a testAction, s *StateDB) {
code := make([]byte, 16)
binary.BigEndian.PutUint64(code, uint64(a.args[0]))
binary.BigEndian.PutUint64(code[8:], uint64(a.args[1]))
s.SetCode(addr, code)
},
args: make([]int64, 2),
},
{
name: "CreateAccount",
fn: func(a testAction, s *StateDB) {
s.CreateAccount(addr)
},
},
{
```

```
name: "Suicide",
fn: func(a testAction, s *StateDB) {
s.Suicide(addr)
},
},
{
name: "AddRefund",
fn: func(a testAction, s *StateDB) {
s.AddRefund(big.NewInt(a.args[0]))
},
args:   make([]int64, 1),
noAddr: true,
},
{
name: "AddLog",
fn: func(a testAction, s *StateDB) {
data := make([]byte, 2)
binary.BigEndian.PutUint16(data, uint16(a.args[0]))
s.AddLog(&types.Log{Address: addr, Data: data})
},
args: make([]int64, 1),
},
}
action := actions[r.Intn(len(actions))]
var nameargs []string
if !action.noAddr {
nameargs = append(nameargs, addr.Hex())
}
for _, i := range action.args {
action.args[i] = rand.Int63n(100)
nameargs = append(nameargs, fmt.Sprint(action.args[i]))
}
action.name += strings.Join(nameargs, ", ")
return action
}

// Generate returns a new snapshot test of the given size. All randomness is
// derived from r.
func (*snapshotTest) Generate(r *rand.Rand, size int) reflect.Value {
// Generate random actions.
addrs := make([]common.Address, 50)
for i := range addrs {
```

```
addrs[i][0] = byte(i)
}
actions := make([]testAction, size)
for i := range actions {
addr := addrs[r.Intn(len(addrs))]
actions[i] = newTestAction(addr, r)
}
// Generate snapshot indexes.
nsnapshots := int(math.Sqrt(float64(size)))
if size > 0 && nsnapshots == 0 {
nsnapshots = 1
}
snapshots := make([]int, nsnapshots)
snaplen := len(actions) / nsnapshots
for i := range snapshots {
// Try to place the snapshots some number of actions apart from each other.
snapshots[i] = (i * snaplen) + r.Intn(snaplen)
}
return reflect.ValueOf(&snapshotTest{addrs, actions, snapshots, nil})
}

func (test *snapshotTest) String() string {
out := new(bytes.Buffer)
sindex := 0
for i, action := range test.actions {
if len(test.snapshots) > sindex && i == test.snapshots[sindex] {
fmt.Fprintf(out, "---- snapshot %d ----\n", sindex)
sindex++
}
fmt.Fprintf(out, "%4d: %s\n", i, action.name)
}
return out.String()
}

func (test *snapshotTest) run() bool {
// Run all actions and create snapshots.
var (
db, _        = ethdb.NewMemDatabase()
state, _     = New(common.Hash{}, NewDatabase(db))
snapshotRevs = make([]int, len(test.snapshots))
sindex       = 0
)
```

```
for i, action := range test.actions {
if len(test.snapshots) > sindex && i == test.snapshots[sindex] {
snapshotRevs[sindex] = state.Snapshot()
sindex++
}
action.fn(action, state)
}

// Revert all snapshots in reverse order. Each revert must yield a state
// that is equivalent to fresh state with all actions up the snapshot applied.
for sindex--; sindex >= 0; sindex-- {
checkstate, _ := New(common.Hash{}, NewDatabase(db))
for _, action := range test.actions[:test.snapshots[sindex]] {
action.fn(action, checkstate)
}
state.RevertToSnapshot(snapshotRevs[sindex])
if err := test.checkEqual(state, checkstate); err != nil {
test.err = fmt.Errorf("state mismatch after revert to snapshot %d\n%v", sindex, err)
return false
}
}
return true
}

// checkEqual checks that methods of state and checkstate return the same values.
func (test *snapshotTest) checkEqual(state, checkstate *StateDB) error {
for _, addr := range test.addrs {
var err error
checkeq := func(op string, a, b interface{}) bool {
if err == nil && !reflect.DeepEqual(a, b) {
err = fmt.Errorf("got %s(%s) == %v, want %v", op, addr.Hex(), a, b)
return false
}
return true
}
// Check basic accessor methods.
checkeq("Exist", state.Exist(addr), checkstate.Exist(addr))
checkeq("HasSuicided", state.HasSuicided(addr), checkstate.HasSuicided(addr))
checkeq("GetBalance", state.GetBalance(addr), checkstate.GetBalance(addr))
checkeq("GetNonce", state.GetNonce(addr), checkstate.GetNonce(addr))
checkeq("GetCode", state.GetCode(addr), checkstate.GetCode(addr))
checkeq("GetCodeHash", state.GetCodeHash(addr), checkstate.GetCodeHash(addr))
```

```
checkeq("GetCodeSize", state.GetCodeSize(addr), checkstate.GetCodeSize(addr))
// Check storage.
if obj := state.getStateObject(addr); obj != nil {
state.ForEachStorage(addr, func(key, val common.Hash) bool {
return checkeq("GetState("+key.Hex()+")", val, checkstate.GetState(addr, key))
})
checkstate.ForEachStorage(addr, func(key, checkval common.Hash) bool {
return checkeq("GetState("+key.Hex()+")", state.GetState(addr, key), checkval)
})
}
if err != nil {
return err
}
}

if state.GetRefund().Cmp(checkstate.GetRefund()) != 0 {
return fmt.Errorf("got GetRefund() == %d, want GetRefund() == %d",
state.GetRefund(), checkstate.GetRefund())
}
if !reflect.DeepEqual(state.GetLogs(common.Hash{}), checkstate.GetLogs(common.Hash{})) {
return fmt.Errorf("got GetLogs(common.Hash{}) == %v, want GetLogs(common.Hash{}) == %v",
state.GetLogs(common.Hash{}), checkstate.GetLogs(common.Hash{}))
}
return nil
}

func (s *StateSuite) TestTouchDelete(c *check.C) {
s.state.GetOrNewStateObject(common.Address{})
root, _ := s.state.CommitTo(s.db, false)
s.state.Reset(root)

snapshot := s.state.Snapshot()
s.state.AddBalance(common.Address{}, new(big.Int))
if len(s.state.stateObjectsDirty) != 1 {
c.Fatal("expected one dirty state object")
}

s.state.RevertToSnapshot(snapshot)
if len(s.state.stateObjectsDirty) != 0 {
c.Fatal("expected no dirty state object")
}
}
```

97:F:\git\coin\ethereum\go-ethereum\core\state\state_object.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"bytes"
"fmt"
"io"
"math/big"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/rlp"
"github.com/ethereum/go-ethereum/trie"
)

var emptyCodeHash = crypto.Keccak256(nil)

type Code []byte

func (self Code) String() string {
return string(self) //strings.Join(Disassemble(self), " ")
}

type Storage map[common.Hash]common.Hash

func (self Storage) String() (str string) {
for key, value := range self {
str += fmt.Sprintf("%X : %X\n", key, value)
}

return
}

func (self Storage) Copy() Storage {
cpy := make(Storage)
for key, value := range self {
cpy[key] = value
}

```
return cpy
}

// stateObject represents an Ethereum account which is being modified.
//
// The usage pattern is as follows:
// First you need to obtain a state object.
// Account values can be accessed and modified through the object.
// Finally, call CommitTrie to write the modified storage trie into a database.
type stateObject struct {
address  common.Address
addrHash common.Hash // hash of ethereum address of the account
data     Account
db       *StateDB

// DB error.
// State objects are used by the consensus core and VM which are
// unable to deal with database-level errors. Any error that occurs
// during a database read is memoized here and will eventually be returned
// by StateDB.Commit.
dbErr error

// Write caches.
trie Trie // storage trie, which becomes non-nil on first access
code Code // contract bytecode, which gets set when code is loaded

cachedStorage Storage // Storage entry cache to avoid duplicate reads
dirtyStorage  Storage // Storage entries that need to be flushed to disk

// Cache flags.
// When an object is marked suicided it will be delete from the trie
// during the "update" phase of the state transition.
dirtyCode bool // true if the code was updated
suicided  bool
touched   bool
deleted   bool
onDirty   func(addr common.Address) // Callback method to mark a state object newly dirty
}

// empty returns whether the account is considered empty.
func (s *stateObject) empty() bool {
return s.data.Nonce == 0 && s.data.Balance.Sign() == 0 && bytes.Equal(s.data.CodeHash,
```

```
emptyCodeHash)
}

// Account is the Ethereum consensus representation of accounts.
// These objects are stored in the main account trie.
type Account struct {
Nonce    uint64
Balance  *big.Int
Root     common.Hash // merkle root of the storage trie
CodeHash []byte
}

// newObject creates a state object.
func newObject(db *StateDB, address common.Address, data Account, onDirty func(addr
common.Address)) *stateObject {
if data.Balance == nil {
data.Balance = new(big.Int)
}
if data.CodeHash == nil {
data.CodeHash = emptyCodeHash
}
return &stateObject{
db:            db,
address:       address,
addrHash:      crypto.Keccak256Hash(address[:]),
data:          data,
cachedStorage: make(Storage),
dirtyStorage:  make(Storage),
onDirty:       onDirty,
}
}

// EncodeRLP implements rlp.Encoder.
func (c *stateObject) EncodeRLP(w io.Writer) error {
return rlp.Encode(w, c.data)
}

// setError remembers the first non-nil error it is called with.
func (self *stateObject) setError(err error) {
if self.dbErr == nil {
self.dbErr = err
}
```

```
}

func (self *stateObject) markSuicided() {
self.suicided = true
if self.onDirty != nil {
self.onDirty(self.Address())
self.onDirty = nil
}
}

func (c *stateObject) touch() {
c.db.journal = append(c.db.journal, touchChange{
account:   &c.address,
prev:      c.touched,
prevDirty: c.onDirty == nil,
})
if c.onDirty != nil {
c.onDirty(c.Address())
c.onDirty = nil
}
c.touched = true
}

func (c *stateObject) getTrie(db Database) Trie {
if c.trie == nil {
var err error
c.trie, err = db.OpenStorageTrie(c.addrHash, c.data.Root)
if err != nil {
c.trie, _ = db.OpenStorageTrie(c.addrHash, common.Hash{})
c.setError(fmt.Errorf("can't create storage trie: %v", err))
}
}
return c.trie
}

// GetState returns a value in account storage.
func (self *stateObject) GetState(db Database, key common.Hash) common.Hash {
value, exists := self.cachedStorage[key]
if exists {
return value
}
// Load from DB in case it is missing.
```

```
enc, err := self.getTrie(db).TryGet(key[:])
if err != nil {
self.setError(err)
return common.Hash{}
}
if len(enc) > 0 {
_, content, _, err := rlp.Split(enc)
if err != nil {
self.setError(err)
}
value.SetBytes(content)
}
if (value != common.Hash{}) {
self.cachedStorage[key] = value
}
return value
}

// SetState updates a value in account storage.
func (self *stateObject) SetState(db Database, key, value common.Hash) {
self.db.journal = append(self.db.journal, storageChange{
account:  &self.address,
key:      key,
prevalue: self.GetState(db, key),
})
self.setState(key, value)
}

func (self *stateObject) setState(key, value common.Hash) {
self.cachedStorage[key] = value
self.dirtyStorage[key] = value

if self.onDirty != nil {
self.onDirty(self.Address())
self.onDirty = nil
}
}

// updateTrie writes cached storage modifications into the object's storage trie.
func (self *stateObject) updateTrie(db Database) Trie {
tr := self.getTrie(db)
for key, value := range self.dirtyStorage {
```

```
delete(self.dirtyStorage, key)
if (value == common.Hash{}) {
self.setError(tr.TryDelete(key[:]))
continue
}
// Encoding []byte cannot fail, ok to ignore the error.
v, _ := rlp.EncodeToBytes(bytes.TrimLeft(value[:], "\x00"))
self.setError(tr.TryUpdate(key[:], v))
}
return tr
}

// UpdateRoot sets the trie root to the current root hash of
func (self *stateObject) updateRoot(db Database) {
self.updateTrie(db)
self.data.Root = self.trie.Hash()
}

// CommitTrie the storage trie of the object to dwb.
// This updates the trie root.
func (self *stateObject) CommitTrie(db Database, dbw trie.DatabaseWriter) error {
self.updateTrie(db)
if self.dbErr != nil {
return self.dbErr
}
root, err := self.trie.CommitTo(dbw)
if err == nil {
self.data.Root = root
}
return err
}

// AddBalance removes amount from c's balance.
// It is used to add funds to the destination account of a transfer.
func (c *stateObject) AddBalance(amount *big.Int) {
// EIP158: We must check emptiness for the objects such that the account
// clearing (0,0,0 objects) can take effect.
if amount.Sign() == 0 {
if c.empty() {
c.touch()
}
```

```
return
}
c.SetBalance(new(big.Int).Add(c.Balance(), amount))
}

// SubBalance removes amount from c's balance.
// It is used to remove funds from the origin account of a transfer.
func (c *stateObject) SubBalance(amount *big.Int) {
if amount.Sign() == 0 {
return
}
c.SetBalance(new(big.Int).Sub(c.Balance(), amount))
}

func (self *stateObject) SetBalance(amount *big.Int) {
self.db.journal = append(self.db.journal, balanceChange{
account: &self.address,
prev:    new(big.Int).Set(self.data.Balance),
})
self.setBalance(amount)
}

func (self *stateObject) setBalance(amount *big.Int) {
self.data.Balance = amount
if self.onDirty != nil {
self.onDirty(self.Address())
self.onDirty = nil
}
}

// Return the gas back to the origin. Used by the Virtual machine or Closures
func (c *stateObject) ReturnGas(gas *big.Int) {}

func (self *stateObject) deepCopy(db *StateDB, onDirty func(addr common.Address)) *stateObject
{
stateObject := newObject(db, self.address, self.data, onDirty)
if self.trie != nil {
stateObject.trie = db.db.CopyTrie(self.trie)
}
stateObject.code = self.code
stateObject.dirtyStorage = self.dirtyStorage.Copy()
stateObject.cachedStorage = self.dirtyStorage.Copy()
```

```
stateObject.suicided = self.suicided
stateObject.dirtyCode = self.dirtyCode
stateObject.deleted = self.deleted
return stateObject
}

//
// Attribute accessors
//

// Returns the address of the contract/account
func (c *stateObject) Address() common.Address {
return c.address
}

// Code returns the contract code associated with this object, if any.
func (self *stateObject) Code(db Database) []byte {
if self.code != nil {
return self.code
}
if bytes.Equal(self.CodeHash(), emptyCodeHash) {
return nil
}
code, err := db.ContractCode(self.addrHash, common.BytesToHash(self.CodeHash()))
if err != nil {
self.setError(fmt.Errorf("can't load code hash %x: %v", self.CodeHash(), err))
}
self.code = code
return code
}

func (self *stateObject) SetCode(codeHash common.Hash, code []byte) {
prevcode := self.Code(self.db.db)
self.db.journal = append(self.db.journal, codeChange{
account:  &self.address,
prevhash: self.CodeHash(),
prevcode: prevcode,
})
self.setCode(codeHash, code)
}

func (self *stateObject) setCode(codeHash common.Hash, code []byte) {
```

```
self.code = code
self.data.CodeHash = codeHash[:]
self.dirtyCode = true
if self.onDirty != nil {
self.onDirty(self.Address())
self.onDirty = nil
}
}

func (self *stateObject) SetNonce(nonce uint64) {
self.db.journal = append(self.db.journal, nonceChange{
account: &self.address,
prev:    self.data.Nonce,
})
self.setNonce(nonce)
}

func (self *stateObject) setNonce(nonce uint64) {
self.data.Nonce = nonce
if self.onDirty != nil {
self.onDirty(self.Address())
self.onDirty = nil
}
}

func (self *stateObject) CodeHash() []byte {
return self.data.CodeHash
}

func (self *stateObject) Balance() *big.Int {
return self.data.Balance
}

func (self *stateObject) Nonce() uint64 {
return self.data.Nonce
}

// Never called, but must be present to allow stateObject to be used
// as a vm.Account interface that also satisfies the vm.ContractRef
// interface. Interfaces are awesome.
func (self *stateObject) Value() *big.Int {
panic("Value on stateObject should never be called")
```

```
}

98:F:\git\coin\ethereum\go-ethereum\core\state\state_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"bytes"
"math/big"
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethdb"
checker "gopkg.in/check.v1"
)

type StateSuite struct {
db    *ethdb.MemDatabase
state *StateDB
}

var _ = checker.Suite(&StateSuite{})

var toAddr = common.BytesToAddress

func (s *StateSuite) TestDump(c *checker.C) {
// generate a few entries
obj1 := s.state.GetOrNewStateObject(toAddr([]byte{0x01}))
obj1.AddBalance(big.NewInt(22))
obj2 := s.state.GetOrNewStateObject(toAddr([]byte{0x01, 0x02}))
obj2.SetCode(crypto.Keccak256Hash([]byte{3, 3, 3, 3, 3, 3, 3}), []byte{3, 3, 3, 3, 3, 3, 3})
obj3 := s.state.GetOrNewStateObject(toAddr([]byte{0x02}))
obj3.SetBalance(big.NewInt(44))

// write some of them to the trie
s.state.updateStateObject(obj1)
s.state.updateStateObject(obj2)
s.state.CommitTo(s.db, false)

// check that dump contains the state objects that are in trie
```

```
got := string(s.state.Dump())
want := `{
   "root": "71edff0130dd2385947095001c73d9e28d862fc286fca2b922ca6f6f3cddfdd2",
   "accounts": {
      "0000000000000000000000000000000000000001": {
         "balance": "22",
         "nonce": 0,
         "root": "56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
         "codeHash":
"c5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470",
         "code": "",
         "storage": {}
      },
      "0000000000000000000000000000000000000002": {
         "balance": "44",
         "nonce": 0,
         "root": "56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
         "codeHash":
"c5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470",
         "code": "",
         "storage": {}
      },
      "0000000000000000000000000000000000000102": {
         "balance": "0",
         "nonce": 0,
         "root": "56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
         "codeHash":
"87874902497a5bb968da31a2998d8f22e949d1ef6214bcdedd8bae24cca4b9e3",
         "code": "03030303030303",
         "storage": {}
      }
   }
}`
if got != want {
c.Errorf("dump mismatch:\ngot: %s\nwant: %s\n", got, want)
}
}

func (s *StateSuite) SetUpTest(c *checker.C) {
s.db, _ = ethdb.NewMemDatabase()
s.state, _ = New(common.Hash{}, NewDatabase(s.db))
}
```

```
func (s *StateSuite) TestNull(c *checker.C) {
address := common.HexToAddress("0x823140710bf13990e4500136726d8b55")
s.state.CreateAccount(address)
//value := common.FromHex("0x823140710bf13990e4500136726d8b55")
var value common.Hash
s.state.SetState(address, common.Hash{}, value)
s.state.CommitTo(s.db, false)
value = s.state.GetState(address, common.Hash{})
if !common.EmptyHash(value) {
c.Errorf("expected empty hash. got %x", value)
}
}

func (s *StateSuite) TestSnapshot(c *checker.C) {
stateobjaddr := toAddr([]byte("aa"))
var storageaddr common.Hash
data1 := common.BytesToHash([]byte{42})
data2 := common.BytesToHash([]byte{43})

// set initial state object value
s.state.SetState(stateobjaddr, storageaddr, data1)
// get snapshot of current state
snapshot := s.state.Snapshot()

// set new state object value
s.state.SetState(stateobjaddr, storageaddr, data2)
// restore snapshot
s.state.RevertToSnapshot(snapshot)

// get state storage value
res := s.state.GetState(stateobjaddr, storageaddr)

c.Assert(data1, checker.DeepEquals, res)
}

func (s *StateSuite) TestSnapshotEmpty(c *checker.C) {
s.state.RevertToSnapshot(s.state.Snapshot())
}

// use testing instead of checker because checker does not support
// printing/logging in tests (-check.vv does not work)
```

```
func TestSnapshot2(t *testing.T) {
db, _ := ethdb.NewMemDatabase()
state, _ := New(common.Hash{}, NewDatabase(db))

stateobjaddr0 := toAddr([]byte("so0"))
stateobjaddr1 := toAddr([]byte("so1"))
var storageaddr common.Hash

data0 := common.BytesToHash([]byte{17})
data1 := common.BytesToHash([]byte{18})

state.SetState(stateobjaddr0, storageaddr, data0)
state.SetState(stateobjaddr1, storageaddr, data1)

// db, trie are already non-empty values
so0 := state.getStateObject(stateobjaddr0)
so0.SetBalance(big.NewInt(42))
so0.SetNonce(43)
so0.SetCode(crypto.Keccak256Hash([]byte{'c', 'a', 'f', 'e'}), []byte{'c', 'a', 'f', 'e'})
so0.suicided = false
so0.deleted = false
state.setStateObject(so0)

root, _ := state.CommitTo(db, false)
state.Reset(root)

// and one with deleted == true
so1 := state.getStateObject(stateobjaddr1)
so1.SetBalance(big.NewInt(52))
so1.SetNonce(53)
so1.SetCode(crypto.Keccak256Hash([]byte{'c', 'a', 'f', 'e', '2'}), []byte{'c', 'a', 'f', 'e', '2'})
so1.suicided = true
so1.deleted = true
state.setStateObject(so1)

so1 = state.getStateObject(stateobjaddr1)
if so1 != nil {
t.Fatalf("deleted object not nil when getting")
}

snapshot := state.Snapshot()
state.RevertToSnapshot(snapshot)
```

```
so0Restored := state.getStateObject(stateobjaddr0)
// Update lazily-loaded values before comparing.
so0Restored.GetState(state.db, storageaddr)
so0Restored.Code(state.db)
// non-deleted is equal (restored)
compareStateObjects(so0Restored, so0, t)

// deleted should be nil, both before and after restore of state copy
so1Restored := state.getStateObject(stateobjaddr1)
if so1Restored != nil {
t.Fatalf("deleted object not nil after restoring snapshot: %+v", so1Restored)
}
}

func compareStateObjects(so0, so1 *stateObject, t *testing.T) {
if so0.Address() != so1.Address() {
t.Fatalf("Address mismatch: have %v, want %v", so0.address, so1.address)
}
if so0.Balance().Cmp(so1.Balance()) != 0 {
t.Fatalf("Balance mismatch: have %v, want %v", so0.Balance(), so1.Balance())
}
if so0.Nonce() != so1.Nonce() {
t.Fatalf("Nonce mismatch: have %v, want %v", so0.Nonce(), so1.Nonce())
}
if so0.data.Root != so1.data.Root {
t.Errorf("Root mismatch: have %x, want %x", so0.data.Root[:], so1.data.Root[:])
}
if !bytes.Equal(so0.CodeHash(), so1.CodeHash()) {
t.Fatalf("CodeHash mismatch: have %v, want %v", so0.CodeHash(), so1.CodeHash())
}
if !bytes.Equal(so0.code, so1.code) {
t.Fatalf("Code mismatch: have %v, want %v", so0.code, so1.code)
}

if len(so1.cachedStorage) != len(so0.cachedStorage) {
t.Errorf("Storage size mismatch: have %d, want %d", len(so1.cachedStorage),
len(so0.cachedStorage))
}
for k, v := range so1.cachedStorage {
if so0.cachedStorage[k] != v {
t.Errorf("Storage key %x mismatch: have %v, want %v", k, so0.cachedStorage[k], v)
```

```
}
}
for k, v := range so0.cachedStorage {
if so1.cachedStorage[k] != v {
t.Errorf("Storage key %x mismatch: have %v, want none.", k, v)
}
}

if so0.suicided != so1.suicided {
t.Fatalf("suicided mismatch: have %v, want %v", so0.suicided, so1.suicided)
}
if so0.deleted != so1.deleted {
t.Fatalf("Deleted mismatch: have %v, want %v", so0.deleted, so1.deleted)
}
}

99:F:\git\coin\ethereum\go-ethereum\core\state\sync.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package state

import (
"bytes"
"math/big"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/rlp"
"github.com/ethereum/go-ethereum/trie"
)

// StateSync is the main state synchronisation scheduler, which provides yet the
// unknown state hashes to retrieve, accepts node data associated with said hashes
// and reconstructs the state database step by step until all is done.
type StateSync trie.TrieSync

// NewStateSync create a new state trie download scheduler.
func NewStateSync(root common.Hash, database trie.DatabaseReader) *StateSync {
var syncer *trie.TrieSync

callback := func(leaf []byte, parent common.Hash) error {
var obj struct {
Nonce    uint64
```

```
Balance  *big.Int
Root     common.Hash
CodeHash []byte
}
if err := rlp.Decode(bytes.NewReader(leaf), &obj); err != nil {
return err
}
syncer.AddSubTrie(obj.Root, 64, parent, nil)
syncer.AddRawEntry(common.BytesToHash(obj.CodeHash), 64, parent)

return nil
}
syncer = trie.NewTrieSync(root, database, callback)
return (*StateSync)(syncer)
}

// Missing retrieves the known missing nodes from the state trie for retrieval.
func (s *StateSync) Missing(max int) []common.Hash {
return (*trie.TrieSync)(s).Missing(max)
}

// Process injects a batch of retrieved trie nodes data, returning if something
// was committed to the memcache and also the index of an entry if processing of
// it failed.
func (s *StateSync) Process(list []trie.SyncResult) (bool, int, error) {
return (*trie.TrieSync)(s).Process(list)
}

// Commit flushes the data stored in the internal memcache out to persistent
// storage, returning th enumber of items written and any occurred error.
func (s *StateSync) Commit(dbw trie.DatabaseWriter) (int, error) {
return (*trie.TrieSync)(s).Commit(dbw)
}

// Pending returns the number of state entries currently pending for download.
func (s *StateSync) Pending() int {
return (*trie.TrieSync)(s).Pending()
}

100:F:\git\coin\ethereum\go-ethereum\core\state\sync_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.
```

```
package state

import (
"bytes"
"math/big"
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/trie"
)

// testAccount is the data associated with an account used by the state tests.
type testAccount struct {
address common.Address
balance *big.Int
nonce   uint64
code    []byte
}

// makeTestState create a sample test state to test node-wise reconstruction.
func makeTestState() (Database, *ethdb.MemDatabase, common.Hash, []*testAccount) {
// Create an empty state
mem, _ := ethdb.NewMemDatabase()
db := NewDatabase(mem)
state, _ := New(common.Hash{}, db)

// Fill it with some arbitrary data
accounts := []*testAccount{}
for i := byte(0); i < 96; i++ {
obj := state.GetOrNewStateObject(common.BytesToAddress([]byte{i}))
acc := &testAccount{address: common.BytesToAddress([]byte{i})}

obj.AddBalance(big.NewInt(int64(11 * i)))
acc.balance = big.NewInt(int64(11 * i))

obj.SetNonce(uint64(42 * i))
acc.nonce = uint64(42 * i)

if i%3 == 0 {
obj.SetCode(crypto.Keccak256Hash([]byte{i, i, i, i, i}), []byte{i, i, i, i, i})
```

```
acc.code = []byte{i, i, i, i, i}
}
state.updateStateObject(obj)
accounts = append(accounts, acc)
}
root, _ := state.CommitTo(mem, false)

// Return the generated state
return db, mem, root, accounts
}

// checkStateAccounts cross references a reconstructed state with an expected
// account array.
func checkStateAccounts(t *testing.T, db ethdb.Database, root common.Hash, accounts
[]*testAccount) {
// Check root availability and state contents
state, err := New(root, NewDatabase(db))
if err != nil {
t.Fatalf("failed to create state trie at %x: %v", root, err)
}
if err := checkStateConsistency(db, root); err != nil {
t.Fatalf("inconsistent state trie at %x: %v", root, err)
}
for i, acc := range accounts {
if balance := state.GetBalance(acc.address); balance.Cmp(acc.balance) != 0 {
t.Errorf("account %d: balance mismatch: have %v, want %v", i, balance, acc.balance)
}
if nonce := state.GetNonce(acc.address); nonce != acc.nonce {
t.Errorf("account %d: nonce mismatch: have %v, want %v", i, nonce, acc.nonce)
}
if code := state.GetCode(acc.address); !bytes.Equal(code, acc.code) {
t.Errorf("account %d: code mismatch: have %x, want %x", i, code, acc.code)
}
}
}

// checkTrieConsistency checks that all nodes in a (sub-)trie are indeed present.
func checkTrieConsistency(db ethdb.Database, root common.Hash) error {
if v, _ := db.Get(root[:]); v == nil {
return nil // Consider a non existent state consistent.
}
trie, err := trie.New(root, db)
```

```
if err != nil {
return err
}
it := trie.NodeIterator(nil)
for it.Next(true) {
}
return it.Error()
}

// checkStateConsistency checks that all data of a state root is present.
func checkStateConsistency(db ethdb.Database, root common.Hash) error {
// Create and iterate a state trie rooted in a sub-node
if _, err := db.Get(root.Bytes()); err != nil {
return nil // Consider a non existent state consistent.
}
state, err := New(root, NewDatabase(db))
if err != nil {
return err
}
it := NewNodeIterator(state)
for it.Next() {
}
return it.Error
}

// Tests that an empty state is not scheduled for syncing.
func TestEmptyStateSync(t *testing.T) {
empty :=
common.HexToHash("56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b42
1")
db, _ := ethdb.NewMemDatabase()
if req := NewStateSync(empty, db).Missing(1); len(req) != 0 {
t.Errorf("content requested for empty state: %v", req)
}
}

// Tests that given a root hash, a state can sync iteratively on a single thread,
// requesting retrieval tasks and returning all of them in one go.
func TestIterativeStateSyncIndividual(t *testing.T) { testIterativeStateSync(t, 1) }
func TestIterativeStateSyncBatched(t *testing.T)    { testIterativeStateSync(t, 100) }

func testIterativeStateSync(t *testing.T, batch int) {
```

```
// Create a random state to copy
_, srcMem, srcRoot, srcAccounts := makeTestState()

// Create a destination state and sync with the scheduler
dstDb, _ := ethdb.NewMemDatabase()
sched := NewStateSync(srcRoot, dstDb)

queue := append([]common.Hash{}, sched.Missing(batch)...)
for len(queue) > 0 {
results := make([]trie.SyncResult, len(queue))
for i, hash := range queue {
data, err := srcMem.Get(hash.Bytes())
if err != nil {
t.Fatalf("failed to retrieve node data for %x: %v", hash, err)
}
results[i] = trie.SyncResult{Hash: hash, Data: data}
}
if _, index, err := sched.Process(results); err != nil {
t.Fatalf("failed to process result #%d: %v", index, err)
}
if index, err := sched.Commit(dstDb); err != nil {
t.Fatalf("failed to commit data #%d: %v", index, err)
}
queue = append(queue[:0], sched.Missing(batch)...)
}
// Cross check that the two states are in sync
checkStateAccounts(t, dstDb, srcRoot, srcAccounts)
}

// Tests that the trie scheduler can correctly reconstruct the state even if only
// partial results are returned, and the others sent only later.
func TestIterativeDelayedStateSync(t *testing.T) {
// Create a random state to copy
_, srcMem, srcRoot, srcAccounts := makeTestState()

// Create a destination state and sync with the scheduler
dstDb, _ := ethdb.NewMemDatabase()
sched := NewStateSync(srcRoot, dstDb)

queue := append([]common.Hash{}, sched.Missing(0)...)
for len(queue) > 0 {
// Sync only half of the scheduled nodes
```

```
results := make([]trie.SyncResult, len(queue)/2+1)
for i, hash := range queue[:len(results)] {
data, err := srcMem.Get(hash.Bytes())
if err != nil {
t.Fatalf("failed to retrieve node data for %x: %v", hash, err)
}
results[i] = trie.SyncResult{Hash: hash, Data: data}
}
if _, index, err := sched.Process(results); err != nil {
t.Fatalf("failed to process result #%d: %v", index, err)
}
if index, err := sched.Commit(dstDb); err != nil {
t.Fatalf("failed to commit data #%d: %v", index, err)
}
queue = append(queue[len(results):], sched.Missing(0)...)
}
// Cross check that the two states are in sync
checkStateAccounts(t, dstDb, srcRoot, srcAccounts)
}

// Tests that given a root hash, a trie can sync iteratively on a single thread,
// requesting retrieval tasks and returning all of them in one go, however in a
// random order.
func TestIterativeRandomStateSyncIndividual(t *testing.T) { testIterativeRandomStateSync(t, 1) }
func TestIterativeRandomStateSyncBatched(t *testing.T)    { testIterativeRandomStateSync(t, 100)
}

func testIterativeRandomStateSync(t *testing.T, batch int) {
// Create a random state to copy
_, srcMem, srcRoot, srcAccounts := makeTestState()

// Create a destination state and sync with the scheduler
dstDb, _ := ethdb.NewMemDatabase()
sched := NewStateSync(srcRoot, dstDb)

queue := make(map[common.Hash]struct{})
for _, hash := range sched.Missing(batch) {
queue[hash] = struct{}{}
}
for len(queue) > 0 {
// Fetch all the queued nodes in a random order
results := make([]trie.SyncResult, 0, len(queue))
```

```
for hash := range queue {
data, err := srcMem.Get(hash.Bytes())
if err != nil {
t.Fatalf("failed to retrieve node data for %x: %v", hash, err)
}
results = append(results, trie.SyncResult{Hash: hash, Data: data})
}
// Feed the retrieved results back and queue new tasks
if _, index, err := sched.Process(results); err != nil {
t.Fatalf("failed to process result #%d: %v", index, err)
}
if index, err := sched.Commit(dstDb); err != nil {
t.Fatalf("failed to commit data #%d: %v", index, err)
}
queue = make(map[common.Hash]struct{})
for _, hash := range sched.Missing(batch) {
queue[hash] = struct{}{}
}
}
// Cross check that the two states are in sync
checkStateAccounts(t, dstDb, srcRoot, srcAccounts)
}

// Tests that the trie scheduler can correctly reconstruct the state even if only
// partial results are returned (Even those randomly), others sent only later.
func TestIterativeRandomDelayedStateSync(t *testing.T) {
// Create a random state to copy
_, srcMem, srcRoot, srcAccounts := makeTestState()

// Create a destination state and sync with the scheduler
dstDb, _ := ethdb.NewMemDatabase()
sched := NewStateSync(srcRoot, dstDb)

queue := make(map[common.Hash]struct{})
for _, hash := range sched.Missing(0) {
queue[hash] = struct{}{}
}
for len(queue) > 0 {
// Sync only half of the scheduled nodes, even those in random order
results := make([]trie.SyncResult, 0, len(queue)/2+1)
for hash := range queue {
delete(queue, hash)
```

```
data, err := srcMem.Get(hash.Bytes())
if err != nil {
t.Fatalf("failed to retrieve node data for %x: %v", hash, err)
}
results = append(results, trie.SyncResult{Hash: hash, Data: data})

if len(results) >= cap(results) {
break
}
}
// Feed the retrieved results back and queue new tasks
if _, index, err := sched.Process(results); err != nil {
t.Fatalf("failed to process result #%d: %v", index, err)
}
if index, err := sched.Commit(dstDb); err != nil {
t.Fatalf("failed to commit data #%d: %v", index, err)
}
for _, hash := range sched.Missing(0) {
queue[hash] = struct{}{}
}
}
// Cross check that the two states are in sync
checkStateAccounts(t, dstDb, srcRoot, srcAccounts)
}

// Tests that at any point in time during a sync, only complete sub-tries are in
// the database.
func TestIncompleteStateSync(t *testing.T) {
// Create a random state to copy
_, srcMem, srcRoot, srcAccounts := makeTestState()

checkTrieConsistency(srcMem, srcRoot)

// Create a destination state and sync with the scheduler
dstDb, _ := ethdb.NewMemDatabase()
sched := NewStateSync(srcRoot, dstDb)

added := []common.Hash{}
queue := append([]common.Hash{}, sched.Missing(1)...)
for len(queue) > 0 {
// Fetch a batch of state nodes
```

```
results := make([]trie.SyncResult, len(queue))
for i, hash := range queue {
data, err := srcMem.Get(hash.Bytes())
if err != nil {
t.Fatalf("failed to retrieve node data for %x: %v", hash, err)
}
results[i] = trie.SyncResult{Hash: hash, Data: data}
}
// Process each of the state nodes
if _, index, err := sched.Process(results); err != nil {
t.Fatalf("failed to process result #%d: %v", index, err)
}
if index, err := sched.Commit(dstDb); err != nil {
t.Fatalf("failed to commit data #%d: %v", index, err)
}
for _, result := range results {
added = append(added, result.Hash)
}
// Check that all known sub-tries added so far are complete or missing entirely.
checkSubtries:
for _, hash := range added {
for _, acc := range srcAccounts {
if hash == crypto.Keccak256Hash(acc.code) {
continue checkSubtries // skip trie check of code nodes.
}
}
// Can't use checkStateConsistency here because subtrie keys may have odd
// length and crash in LeafKey.
if err := checkTrieConsistency(dstDb, hash); err != nil {
t.Fatalf("state inconsistent: %v", err)
}
}
// Fetch the next batch to retrieve
queue = append(queue[:0], sched.Missing(1)...)
}
// Sanity check that removing any node from the database is detected
for _, node := range added[1:] {
key := node.Bytes()
value, _ := dstDb.Get(key)

dstDb.Delete(key)
if err := checkStateConsistency(dstDb, added[0]); err == nil {
```

```
t.Fatalf("trie inconsistency not caught, missing: %x", key)
}
dstDb.Put(key, value)
}
}
```