F:\git\java\mar3\filemonitor\target\go-ethereum\go-ethereum-0.doc

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

package abi

import (
"encoding/json"
"fmt"
"io"
"reflect"
"strings"

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

// The ABI holds information about a contract's context and available
// invokable methods. It will allow you to type check function calls and
// packs data accordingly.
type ABI struct {
Constructor Method
Methods   map[string]Method
Events   map[string]Event
}

// JSON returns a parsed ABI interface and error if it failed.
func JSON(reader io.Reader) (ABI, error) {
dec := json.NewDecoder(reader)

var abi ABI
if err := dec.Decode(&abi); err != nil {
return ABI{}, err
}

return abi, nil
}

// Pack the given method name to conform the ABI. Method call's data
// will consist of method_id, args0, arg1, ... argN. Method id consists
// of 4 bytes and arguments are all 32 bytes.
// Method ids are created from the first 4 bytes of the hash of the

```
// methods string signature. (signature = baz(uint32,string32))
func (abi ABI) Pack(name string, args ...interface{}) ([]byte, error) {
// Fetch the ABI of the requested method
var method Method

if name == "" {
method = abi.Constructor
} else {
m, exist := abi.Methods[name]
if !exist {
return nil, fmt.Errorf("method '%s' not found", name)
}
method = m
}
arguments, err := method.pack(args...)
if err != nil {
return nil, err
}
// Pack up the method ID too if not a constructor and return
if name == "" {
return arguments, nil
}
return append(method.Id(), arguments...), nil
}

// these variable are used to determine certain types during type assertion for
// assignment.
var (
r_interSlice = reflect.TypeOf([]interface{}{})
r_hash       = reflect.TypeOf(common.Hash{})
r_bytes      = reflect.TypeOf([]byte{})
r_byte       = reflect.TypeOf(byte(0))
)

// Unpack output in v according to the abi specification
func (abi ABI) Unpack(v interface{}, name string, output []byte) error {
var method = abi.Methods[name]

if len(output) == 0 {
return fmt.Errorf("abi: unmarshalling empty output")
}
```

```
// make sure the passed value is a pointer
valueOf := reflect.ValueOf(v)
if reflect.Ptr != valueOf.Kind() {
return fmt.Errorf("abi: Unpack(non-pointer %T)", v)
}

var (
value = valueOf.Elem()
typ   = value.Type()
)

if len(method.Outputs) > 1 {
switch value.Kind() {
// struct will match named return values to the struct's field
// names
case reflect.Struct:
for i := 0; i < len(method.Outputs); i++ {
marshalledValue, err := toGoType(i, method.Outputs[i], output)
if err != nil {
return err
}
reflectValue := reflect.ValueOf(marshalledValue)

for j := 0; j < typ.NumField(); j++ {
field := typ.Field(j)
// TODO read tags: `abi:"fieldName"`
if field.Name == strings.ToUpper(method.Outputs[i].Name[:1])+method.Outputs[i].Name[1:] {
if err := set(value.Field(j), reflectValue, method.Outputs[i]); err != nil {
return err
}
}
}
}
case reflect.Slice:
if !value.Type().AssignableTo(r_interSlice) {
return fmt.Errorf("abi: cannot marshal tuple in to slice %T (only []interface{} is supported)", v)
}

// if the slice already contains values, set those instead of the interface slice itself.
if value.Len() > 0 {
if len(method.Outputs) > value.Len() {
return fmt.Errorf("abi: cannot marshal in to slices of unequal size (require: %v, got: %v)",
```

```
len(method.Outputs), value.Len())
}

for i := 0; i < len(method.Outputs); i++ {
marshalledValue, err := toGoType(i, method.Outputs[i], output)
if err != nil {
return err
}
reflectValue := reflect.ValueOf(marshalledValue)
if err := set(value.Index(i).Elem(), reflectValue, method.Outputs[i]); err != nil {
return err
}
}
return nil
}

// create a new slice and start appending the unmarshalled
// values to the new interface slice.
z := reflect.MakeSlice(typ, 0, len(method.Outputs))
for i := 0; i < len(method.Outputs); i++ {
marshalledValue, err := toGoType(i, method.Outputs[i], output)
if err != nil {
return err
}
z = reflect.Append(z, reflect.ValueOf(marshalledValue))
}
value.Set(z)
default:
return fmt.Errorf("abi: cannot unmarshal tuple in to %v", typ)
}

} else {
marshalledValue, err := toGoType(0, method.Outputs[0], output)
if err != nil {
return err
}
if err := set(value, reflect.ValueOf(marshalledValue), method.Outputs[0]); err != nil {
return err
}
}

return nil
```

```
}

func (abi *ABI) UnmarshalJSON(data []byte) error {
var fields []struct {
Type      string
Name      string
Constant  bool
Indexed   bool
Anonymous bool
Inputs    []Argument
Outputs   []Argument
}

if err := json.Unmarshal(data, &fields); err != nil {
return err
}

abi.Methods = make(map[string]Method)
abi.Events = make(map[string]Event)
for _, field := range fields {
switch field.Type {
case "constructor":
abi.Constructor = Method{
Inputs: field.Inputs,
}
// empty defaults to function according to the abi spec
case "function", "":
abi.Methods[field.Name] = Method{
Name:    field.Name,
Const:   field.Constant,
Inputs:  field.Inputs,
Outputs: field.Outputs,
}
case "event":
abi.Events[field.Name] = Event{
Name:      field.Name,
Anonymous: field.Anonymous,
Inputs:    field.Inputs,
}
}
}
```

```
return nil
}

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

package abi

import (
"bytes"
"fmt"
"log"
"math/big"
"reflect"
"strings"
"testing"

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

// formatSilceOutput add padding to the value and adds a size
func formatSliceOutput(v ...[]byte) []byte {
off := common.LeftPadBytes(big.NewInt(int64(len(v))).Bytes(), 32)
output := append(off, make([]byte, 0, len(v)*32)...)

for _, value := range v {
output = append(output, common.LeftPadBytes(value, 32)...)
}
return output
}

// quick helper padding
func pad(input []byte, size int, left bool) []byte {
if left {
return common.LeftPadBytes(input, size)
}
return common.RightPadBytes(input, size)
}

const jsondata = `
[
```

```
{ "type" : "function", "name" : "balance", "constant" : true },
{ "type" : "function", "name" : "send", "constant" : false, "inputs" : [ { "name" : "amount", "type" : "uint256" } ] }
]`

const jsondata2 = `
[
{ "type" : "function", "name" : "balance", "constant" : true },
{ "type" : "function", "name" : "send", "constant" : false, "inputs" : [ { "name" : "amount", "type" : "uint256" } ] },
{ "type" : "function", "name" : "test", "constant" : false, "inputs" : [ { "name" : "number", "type" : "uint32" } ] },
{ "type" : "function", "name" : "string", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "string" } ] },
{ "type" : "function", "name" : "bool", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "bool" } ] },
{ "type" : "function", "name" : "address", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "address" } ] },
{ "type" : "function", "name" : "uint64[2]", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "uint64[2]" } ] },
{ "type" : "function", "name" : "uint64[]", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "uint64[]" } ] },
{ "type" : "function", "name" : "foo", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "uint32" } ] },
{ "type" : "function", "name" : "bar", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "uint32" }, { "name" : "string", "type" : "uint16" } ] },
{ "type" : "function", "name" : "slice", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "uint32[2]" } ] },
{ "type" : "function", "name" : "slice256", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "uint256[2]" } ] },
{ "type" : "function", "name" : "sliceAddress", "constant" : false, "inputs" : [ { "name" : "inputs", "type" : "address[]" } ] },
{ "type" : "function", "name" : "sliceMultiAddress", "constant" : false, "inputs" : [ { "name" : "a", "type" : "address[]" }, { "name" : "b", "type" : "address[]" } ] }
]`

func TestReader(t *testing.T) {
Uint256, _ := NewType("uint256")
exp := ABI{
Methods: map[string]Method{
"balance": {
"balance", true, nil, nil,
```

```
},
"send": {
"send", false, []Argument{
{"amount", Uint256, false},
}, nil,
},
},
}

abi, err := JSON(strings.NewReader(jsondata))
if err != nil {
t.Error(err)
}

// deep equal fails for some reason
t.Skip()
if !reflect.DeepEqual(abi, exp) {
t.Errorf("\nabi: %v\ndoes not match exp: %v", abi, exp)
}
}

func TestTestNumbers(t *testing.T) {
abi, err := JSON(strings.NewReader(jsondata2))
if err != nil {
t.Error(err)
t.FailNow()
}

if _, err := abi.Pack("balance"); err != nil {
t.Error(err)
}

if _, err := abi.Pack("balance", 1); err == nil {
t.Error("expected error for balance(1)")
}

if _, err := abi.Pack("doesntexist", nil); err == nil {
t.Errorf("doesntexist shouldn't exist")
}

if _, err := abi.Pack("doesntexist", 1); err == nil {
t.Errorf("doesntexist(1) shouldn't exist")
```

```
}

if _, err := abi.Pack("send", big.NewInt(1000)); err != nil {
t.Error(err)
}

i := new(int)
*i = 1000
if _, err := abi.Pack("send", i); err == nil {
t.Errorf("expected send( ptr ) to throw, requires *big.Int instead of *int")
}

if _, err := abi.Pack("test", uint32(1000)); err != nil {
t.Error(err)
}
}

func TestTestString(t *testing.T) {
abi, err := JSON(strings.NewReader(jsondata2))
if err != nil {
t.Error(err)
t.FailNow()
}

if _, err := abi.Pack("string", "hello world"); err != nil {
t.Error(err)
}
}

func TestTestBool(t *testing.T) {
abi, err := JSON(strings.NewReader(jsondata2))
if err != nil {
t.Error(err)
t.FailNow()
}

if _, err := abi.Pack("bool", true); err != nil {
t.Error(err)
}
}

func TestTestSlice(t *testing.T) {
```

```
abi, err := JSON(strings.NewReader(jsondata2))
if err != nil {
t.Error(err)
t.FailNow()
}

slice := make([]uint64, 2)
if _, err := abi.Pack("uint64[2]", slice); err != nil {
t.Error(err)
}

if _, err := abi.Pack("uint64[]", slice); err != nil {
t.Error(err)
}
}

func TestMethodSignature(t *testing.T) {
String, _ := NewType("string")
m := Method{"foo", false, []Argument{{"bar", String, false}, {"baz", String, false}}, nil}
exp := "foo(string,string)"
if m.Sig() != exp {
t.Error("signature mismatch", exp, "!=", m.Sig())
}

idexp := crypto.Keccak256([]byte(exp))[:4]
if !bytes.Equal(m.Id(), idexp) {
t.Errorf("expected ids to match %x != %x", m.Id(), idexp)
}

uintt, _ := NewType("uint")
m = Method{"foo", false, []Argument{{"bar", uintt, false}}, nil}
exp = "foo(uint256)"
if m.Sig() != exp {
t.Error("signature mismatch", exp, "!=", m.Sig())
}
}

func TestMultiPack(t *testing.T) {
abi, err := JSON(strings.NewReader(jsondata2))
if err != nil {
t.Error(err)
t.FailNow()
```

```
}

sig := crypto.Keccak256([]byte("bar(uint32,uint16)"))[:4]
sig = append(sig, make([]byte, 64)...)
sig[35] = 10
sig[67] = 11

packed, err := abi.Pack("bar", uint32(10), uint16(11))
if err != nil {
t.Error(err)
t.FailNow()
}

if !bytes.Equal(packed, sig) {
t.Errorf("expected %x got %x", sig, packed)
}
}

func ExampleJSON() {
const definition =
`[{"constant":true,"inputs":[{"name":"","type":"address"}],"name":"isBar","outputs":[{"name":"","type":
"bool"}],"type":"function"}]`

abi, err := JSON(strings.NewReader(definition))
if err != nil {
log.Fatalln(err)
}
out, err := abi.Pack("isBar", common.HexToAddress("01"))
if err != nil {
log.Fatalln(err)
}

fmt.Printf("%x\n", out)
// Output:
// 1f2c4092000000000000000000000000000000000000000000000000000000000000000001
}

func TestInputVariableInputLength(t *testing.T) {
const definition = `[
{ "type" : "function", "name" : "strOne", "constant" : true, "inputs" : [ { "name" : "str", "type" : "string"
} ] },
{ "type" : "function", "name" : "bytesOne", "constant" : true, "inputs" : [ { "name" : "str", "type" :
```

```
"bytes" } ] },
{ "type" : "function", "name" : "strTwo", "constant" : true, "inputs" : [ { "name" : "str", "type" : "string"
}, { "name" : "str1", "type" : "string" } ] }
]`

abi, err := JSON(strings.NewReader(definition))
if err != nil {
t.Fatal(err)
}

// test one string
strin := "hello world"
strpack, err := abi.Pack("strOne", strin)
if err != nil {
t.Error(err)
}

offset := make([]byte, 32)
offset[31] = 32
length := make([]byte, 32)
length[31] = byte(len(strin))
value := common.RightPadBytes([]byte(strin), 32)
exp := append(offset, append(length, value...)...)

// ignore first 4 bytes of the output. This is the function identifier
strpack = strpack[4:]
if !bytes.Equal(strpack, exp) {
t.Errorf("expected %x, got %x\n", exp, strpack)
}

// test one bytes
btspack, err := abi.Pack("bytesOne", []byte(strin))
if err != nil {
t.Error(err)
}
// ignore first 4 bytes of the output. This is the function identifier
btspack = btspack[4:]
if !bytes.Equal(btspack, exp) {
t.Errorf("expected %x, got %x\n", exp, btspack)
}

//  test two strings
```

```
str1 := "hello"
str2 := "world"
str2pack, err := abi.Pack("strTwo", str1, str2)
if err != nil {
t.Error(err)
}

offset1 := make([]byte, 32)
offset1[31] = 64
length1 := make([]byte, 32)
length1[31] = byte(len(str1))
value1 := common.RightPadBytes([]byte(str1), 32)

offset2 := make([]byte, 32)
offset2[31] = 128
length2 := make([]byte, 32)
length2[31] = byte(len(str2))
value2 := common.RightPadBytes([]byte(str2), 32)

exp2 := append(offset1, offset2...)
exp2 = append(exp2, append(length1, value1...)...)
exp2 = append(exp2, append(length2, value2...)...)

// ignore first 4 bytes of the output. This is the function identifier
str2pack = str2pack[4:]
if !bytes.Equal(str2pack, exp2) {
t.Errorf("expected %x, got %x\n", exp, str2pack)
}

// test two strings, first > 32, second < 32
str1 = strings.Repeat("a", 33)
str2pack, err = abi.Pack("strTwo", str1, str2)
if err != nil {
t.Error(err)
}

offset1 = make([]byte, 32)
offset1[31] = 64
length1 = make([]byte, 32)
length1[31] = byte(len(str1))
value1 = common.RightPadBytes([]byte(str1), 64)
offset2[31] = 160
```

```
exp2 = append(offset1, offset2...)
exp2 = append(exp2, append(length1, value1...)...)
exp2 = append(exp2, append(length2, value2...)...)

// ignore first 4 bytes of the output. This is the function identifier
str2pack = str2pack[4:]
if !bytes.Equal(str2pack, exp2) {
t.Errorf("expected %x, got %x\n", exp, str2pack)
}

// test two strings, first > 32, second >32
str1 = strings.Repeat("a", 33)
str2 = strings.Repeat("a", 33)
str2pack, err = abi.Pack("strTwo", str1, str2)
if err != nil {
t.Error(err)
}

offset1 = make([]byte, 32)
offset1[31] = 64
length1 = make([]byte, 32)
length1[31] = byte(len(str1))
value1 = common.RightPadBytes([]byte(str1), 64)

offset2 = make([]byte, 32)
offset2[31] = 160
length2 = make([]byte, 32)
length2[31] = byte(len(str2))
value2 = common.RightPadBytes([]byte(str2), 64)

exp2 = append(offset1, offset2...)
exp2 = append(exp2, append(length1, value1...)...)
exp2 = append(exp2, append(length2, value2...)...)

// ignore first 4 bytes of the output. This is the function identifier
str2pack = str2pack[4:]
if !bytes.Equal(str2pack, exp2) {
t.Errorf("expected %x, got %x\n", exp, str2pack)
}
}
```

```
func TestDefaultFunctionParsing(t *testing.T) {
const definition = `[{ "name" : "balance" }]`

abi, err := JSON(strings.NewReader(definition))
if err != nil {
t.Fatal(err)
}

if _, ok := abi.Methods["balance"]; !ok {
t.Error("expected 'balance' to be present")
}
}

func TestBareEvents(t *testing.T) {
const definition = `[
{ "type" : "event", "name" : "balance" },
{ "type" : "event", "name" : "anon", "anonymous" : true},
{ "type" : "event", "name" : "args", "inputs" : [{ "indexed":false, "name":"arg0", "type":"uint256" }, {
"indexed":true, "name":"arg1", "type":"address" }] }
]`

arg0, _ := NewType("uint256")
arg1, _ := NewType("address")

expectedEvents := map[string]struct {
Anonymous bool
Args      []Argument
}{
"balance": {false, nil},
"anon":    {true, nil},
"args": {false, []Argument{
{Name: "arg0", Type: arg0, Indexed: false},
{Name: "arg1", Type: arg1, Indexed: true},
}},
}

abi, err := JSON(strings.NewReader(definition))
if err != nil {
t.Fatal(err)
}

if len(abi.Events) != len(expectedEvents) {
```

```
t.Fatalf("invalid number of events after parsing, want %d, got %d", len(expectedEvents),
len(abi.Events))
}

for name, exp := range expectedEvents {
got, ok := abi.Events[name]
if !ok {
t.Errorf("could not found event %s", name)
continue
}
if got.Anonymous != exp.Anonymous {
t.Errorf("invalid anonymous indication for event %s, want %v, got %v", name, exp.Anonymous,
got.Anonymous)
}
if len(got.Inputs) != len(exp.Args) {
t.Errorf("invalid number of args, want %d, got %d", len(exp.Args), len(got.Inputs))
continue
}
for i, arg := range exp.Args {
if arg.Name != got.Inputs[i].Name {
t.Errorf("events[%s].Input[%d] has an invalid name, want %s, got %s", name, i, arg.Name,
got.Inputs[i].Name)
}
if arg.Indexed != got.Inputs[i].Indexed {
t.Errorf("events[%s].Input[%d] has an invalid indexed indication, want %v, got %v", name, i,
arg.Indexed, got.Inputs[i].Indexed)
}
if arg.Type.T != got.Inputs[i].Type.T {
t.Errorf("events[%s].Input[%d] has an invalid type, want %x, got %x", name, i, arg.Type.T,
got.Inputs[i].Type.T)
}
}
}
}
```

2:F:\git\coin\ethereum\go-ethereum\accounts\abi\argument.go

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

package abi

import (
"encoding/json"
```

```
"fmt"
)

// Argument holds the name of the argument and the corresponding type.
// Types are used when packing and testing arguments.
type Argument struct {
Name    string
Type    Type
Indexed bool // indexed is only used by events
}

func (a *Argument) UnmarshalJSON(data []byte) error {
var extarg struct {
Name    string
Type    string
Indexed bool
}
err := json.Unmarshal(data, &extarg)
if err != nil {
return fmt.Errorf("argument json err: %v", err)
}

a.Type, err = NewType(extarg.Type)
if err != nil {
return err
}
a.Name = extarg.Name
a.Indexed = extarg.Indexed

return nil
}

3:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\auth.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package bind

import (
"crypto/ecdsa"
"errors"
"io"
"io/ioutil"
```

```
"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
)

// NewTransactor is a utility method to easily create a transaction signer from
// an encrypted json key stream and the associated passphrase.
func NewTransactor(keyin io.Reader, passphrase string) (*TransactOpts, error) {
json, err := ioutil.ReadAll(keyin)
if err != nil {
return nil, err
}
key, err := keystore.DecryptKey(json, passphrase)
if err != nil {
return nil, err
}
return NewKeyedTransactor(key.PrivateKey), nil
}

// NewKeyedTransactor is a utility method to easily create a transaction signer
// from a single private key.
func NewKeyedTransactor(key *ecdsa.PrivateKey) *TransactOpts {
keyAddr := crypto.PubkeyToAddress(key.PublicKey)
return &TransactOpts{
From: keyAddr,
Signer: func(signer types.Signer, address common.Address, tx *types.Transaction)
(*types.Transaction, error) {
if address != keyAddr {
return nil, errors.New("not authorized to sign this account")
}
signature, err := crypto.Sign(signer.Hash(tx).Bytes(), key)
if err != nil {
return nil, err
}
return tx.WithSignature(signer, signature)
},
}
}
```

4:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\backend.go

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

package bind

import (
"context"
"errors"
"math/big"

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

var (
// ErrNoCode is returned by call and transact operations for which the requested
// recipient contract to operate on does not exist in the state db or does not
// have any code associated with it (i.e. suicided).
ErrNoCode = errors.New("no contract code at given address")

// This error is raised when attempting to perform a pending state action
// on a backend that doesn't implement PendingContractCaller.
ErrNoPendingState = errors.New("backend does not support pending state")

// This error is returned by WaitDeployed if contract creation leaves an
// empty contract behind.
ErrNoCodeAfterDeploy = errors.New("no contract code after deployment")
)

// ContractCaller defines the methods needed to allow operating with contract on a read
// only basis.
type ContractCaller interface {
// CodeAt returns the code of the given account. This is needed to differentiate
// between contract internal errors and the local chain being out of sync.
CodeAt(ctx context.Context, contract common.Address, blockNumber *big.Int) ([]byte, error)
// ContractCall executes an Ethereum contract call with the specified data as the
// input.
CallContract(ctx context.Context, call ethereum.CallMsg, blockNumber *big.Int) ([]byte, error)
}

// DeployBackend wraps the operations needed by WaitMined and WaitDeployed.
type DeployBackend interface {
```

```
TransactionReceipt(ctx context.Context, txHash common.Hash) (*types.Receipt, error)
CodeAt(ctx context.Context, account common.Address, blockNumber *big.Int) ([]byte, error)
}

// PendingContractCaller defines methods to perform contract calls on the pending state.
// Call will try to discover this interface when access to the pending state is requested.
// If the backend does not support the pending state, Call returns ErrNoPendingState.
type PendingContractCaller interface {
// PendingCodeAt returns the code of the given account in the pending state.
PendingCodeAt(ctx context.Context, contract common.Address) ([]byte, error)
// PendingCallContract executes an Ethereum contract call against the pending state.
PendingCallContract(ctx context.Context, call ethereum.CallMsg) ([]byte, error)
}

// ContractTransactor defines the methods needed to allow operating with contract
// on a write only basis. Beside the transacting method, the remainder are helpers
// used when the user does not provide some needed values, but rather leaves it up
// to the transactor to decide.
type ContractTransactor interface {
// PendingCodeAt returns the code of the given account in the pending state.
PendingCodeAt(ctx context.Context, account common.Address) ([]byte, error)
// PendingNonceAt retrieves the current pending nonce associated with an account.
PendingNonceAt(ctx context.Context, account common.Address) (uint64, error)
// SuggestGasPrice retrieves the currently suggested gas price to allow a timely
// execution of a transaction.
SuggestGasPrice(ctx context.Context) (*big.Int, error)
// EstimateGas tries to estimate the gas needed to execute a specific
// transaction based on the current pending state of the backend blockchain.
// There is no guarantee that this is the true gas limit requirement as other
// transactions may be added or removed by miners, but it should provide a basis
// for setting a reasonable default.
EstimateGas(ctx context.Context, call ethereum.CallMsg) (usedGas *big.Int, err error)
// SendTransaction injects the transaction into the pending pool for execution.
SendTransaction(ctx context.Context, tx *types.Transaction) error
}

// ContractBackend defines the methods needed to work with contracts on a read-write basis.
type ContractBackend interface {
ContractCaller
ContractTransactor
}
```

5:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\backends\simulated.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package backends

import (
"context"
"errors"
"fmt"
"math/big"
"sync"

"github.com/ethereum/go-ethereum"
"github.com/ethereum/go-ethereum/accounts/abi/bind"
"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"
"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"
)

// This nil assignment ensures compile time that SimulatedBackend implements
bind.ContractBackend.
var _ bind.ContractBackend = (*SimulatedBackend)(nil)

var errBlockNumberUnsupported = errors.New("SimulatedBackend cannot access blocks other
than the latest block")

// SimulatedBackend implements bind.ContractBackend, simulating a blockchain in
// the background. Its main purpose is to allow easily testing contract bindings.
type SimulatedBackend struct {
database   ethdb.Database   // In memory database to store our testing data
blockchain *core.BlockChain // Ethereum blockchain to handle the consensus

mu   sync.Mutex
pendingBlock *types.Block   // Currently pending block that will be imported on request
pendingState *state.StateDB // Currently pending state that will be the active on on request

```
config *params.ChainConfig
}

// NewSimulatedBackend creates a new binding backend using a simulated blockchain
// for testing purposes.
func NewSimulatedBackend(alloc core.GenesisAlloc) *SimulatedBackend {
database, _ := ethdb.NewMemDatabase()
genesis := core.Genesis{Config: params.AllProtocolChanges, Alloc: alloc}
genesis.MustCommit(database)
blockchain, _ := core.NewBlockChain(database, genesis.Config, ethash.NewFaker(),
new(event.TypeMux), vm.Config{})
backend := &SimulatedBackend{database: database, blockchain: blockchain, config:
genesis.Config}
backend.rollback()
return backend
}

// Commit imports all the pending transactions as a single block and starts a
// fresh new state.
func (b *SimulatedBackend) Commit() {
b.mu.Lock()
defer b.mu.Unlock()

if _, err := b.blockchain.InsertChain([]*types.Block{b.pendingBlock}); err != nil {
panic(err) // This cannot happen unless the simulator is wrong, fail in that case
}
b.rollback()
}

// Rollback aborts all pending transactions, reverting to the last committed state.
func (b *SimulatedBackend) Rollback() {
b.mu.Lock()
defer b.mu.Unlock()

b.rollback()
}

func (b *SimulatedBackend) rollback() {
blocks, _ := core.GenerateChain(b.config, b.blockchain.CurrentBlock(), b.database, 1, func(int,
*core.BlockGen) {})
b.pendingBlock = blocks[0]
```

```
b.pendingState, _ = state.New(b.pendingBlock.Root(), state.NewDatabase(b.database))
}

// CodeAt returns the code associated with a certain account in the blockchain.
func (b *SimulatedBackend) CodeAt(ctx context.Context, contract common.Address, blockNumber
*big.Int) ([]byte, error) {
b.mu.Lock()
defer b.mu.Unlock()

if blockNumber != nil && blockNumber.Cmp(b.blockchain.CurrentBlock().Number()) != 0 {
return nil, errBlockNumberUnsupported
}
statedb, _ := b.blockchain.State()
return statedb.GetCode(contract), nil
}

// BalanceAt returns the wei balance of a certain account in the blockchain.
func (b *SimulatedBackend) BalanceAt(ctx context.Context, contract common.Address,
blockNumber *big.Int) (*big.Int, error) {
b.mu.Lock()
defer b.mu.Unlock()

if blockNumber != nil && blockNumber.Cmp(b.blockchain.CurrentBlock().Number()) != 0 {
return nil, errBlockNumberUnsupported
}
statedb, _ := b.blockchain.State()
return statedb.GetBalance(contract), nil
}

// NonceAt returns the nonce of a certain account in the blockchain.
func (b *SimulatedBackend) NonceAt(ctx context.Context, contract common.Address,
blockNumber *big.Int) (uint64, error) {
b.mu.Lock()
defer b.mu.Unlock()

if blockNumber != nil && blockNumber.Cmp(b.blockchain.CurrentBlock().Number()) != 0 {
return 0, errBlockNumberUnsupported
}
statedb, _ := b.blockchain.State()
return statedb.GetNonce(contract), nil
}
```

```
// StorageAt returns the value of key in the storage of an account in the blockchain.
func (b *SimulatedBackend) StorageAt(ctx context.Context, contract common.Address, key
common.Hash, blockNumber *big.Int) ([]byte, error) {
b.mu.Lock()
defer b.mu.Unlock()

if blockNumber != nil && blockNumber.Cmp(b.blockchain.CurrentBlock().Number()) != 0 {
return nil, errBlockNumberUnsupported
}
statedb, _ := b.blockchain.State()
val := statedb.GetState(contract, key)
return val[:], nil
}

// TransactionReceipt returns the receipt of a transaction.
func (b *SimulatedBackend) TransactionReceipt(ctx context.Context, txHash common.Hash)
(*types.Receipt, error) {
return core.GetReceipt(b.database, txHash), nil
}

// PendingCodeAt returns the code associated with an account in the pending state.
func (b *SimulatedBackend) PendingCodeAt(ctx context.Context, contract common.Address)
([]byte, error) {
b.mu.Lock()
defer b.mu.Unlock()

return b.pendingState.GetCode(contract), nil
}

// CallContract executes a contract call.
func (b *SimulatedBackend) CallContract(ctx context.Context, call ethereum.CallMsg,
blockNumber *big.Int) ([]byte, error) {
b.mu.Lock()
defer b.mu.Unlock()

if blockNumber != nil && blockNumber.Cmp(b.blockchain.CurrentBlock().Number()) != 0 {
return nil, errBlockNumberUnsupported
}
state, err := b.blockchain.State()
if err != nil {
return nil, err
}
```

```
rval, _, err := b.callContract(ctx, call, b.blockchain.CurrentBlock(), state)
return rval, err
}

// PendingCallContract executes a contract call on the pending state.
func (b *SimulatedBackend) PendingCallContract(ctx context.Context, call ethereum.CallMsg)
([]byte, error) {
b.mu.Lock()
defer b.mu.Unlock()
defer b.pendingState.RevertToSnapshot(b.pendingState.Snapshot())

rval, _, err := b.callContract(ctx, call, b.pendingBlock, b.pendingState)
return rval, err
}

// PendingNonceAt implements PendingStateReader.PendingNonceAt, retrieving
// the nonce currently pending for the account.
func (b *SimulatedBackend) PendingNonceAt(ctx context.Context, account common.Address)
(uint64, error) {
b.mu.Lock()
defer b.mu.Unlock()

return b.pendingState.GetOrNewStateObject(account).Nonce(), nil
}

// SuggestGasPrice implements ContractTransactor.SuggestGasPrice. Since the simulated
// chain doens't have miners, we just return a gas price of 1 for any call.
func (b *SimulatedBackend) SuggestGasPrice(ctx context.Context) (*big.Int, error) {
return big.NewInt(1), nil
}

// EstimateGas executes the requested code against the currently pending block/state and
// returns the used amount of gas.
func (b *SimulatedBackend) EstimateGas(ctx context.Context, call ethereum.CallMsg) (*big.Int,
error) {
b.mu.Lock()
defer b.mu.Unlock()

// Binary search the gas requirement, as it may be higher than the amount used
var lo, hi uint64
if call.Gas != nil {
hi = call.Gas.Uint64()
```

```
} else {
hi = b.pendingBlock.GasLimit().Uint64()
}
for lo+1 < hi {
// Take a guess at the gas, and check transaction validity
mid := (hi + lo) / 2
call.Gas = new(big.Int).SetUint64(mid)

snapshot := b.pendingState.Snapshot()
_, gas, err := b.callContract(ctx, call, b.pendingBlock, b.pendingState)
b.pendingState.RevertToSnapshot(snapshot)

// If the transaction became invalid or used all the gas (failed), raise the gas limit
if err != nil || gas.Cmp(call.Gas) == 0 {
lo = mid
continue
}
// Otherwise assume the transaction succeeded, lower the gas limit
hi = mid
}
return new(big.Int).SetUint64(hi), nil
}

// callContract implemens common code between normal and pending contract calls.
// state is modified during execution, make sure to copy it if necessary.
func (b *SimulatedBackend) callContract(ctx context.Context, call ethereum.CallMsg, block
*types.Block, statedb *state.StateDB) ([]byte, *big.Int, error) {
// Ensure message is initialized properly.
if call.GasPrice == nil {
call.GasPrice = big.NewInt(1)
}
if call.Gas == nil || call.Gas.Sign() == 0 {
call.Gas = big.NewInt(50000000)
}
if call.Value == nil {
call.Value = new(big.Int)
}
// Set infinite balance to the fake caller account.
from := statedb.GetOrNewStateObject(call.From)
from.SetBalance(math.MaxBig256)
// Execute the call.
msg := callmsg{call}
```

```
evmContext := core.NewEVMContext(msg, block.Header(), b.blockchain, nil)
// Create a new environment which holds all relevant information
// about the transaction and calling mechanisms.
vmenv := vm.NewEVM(evmContext, statedb, b.config, vm.Config{})
gaspool := new(core.GasPool).AddGas(math.MaxBig256)
ret, gasUsed, _, err := core.NewStateTransition(vmenv, msg, gaspool).TransitionDb()
return ret, gasUsed, err
}

// SendTransaction updates the pending block to include the given transaction.
// It panics if the transaction is invalid.
func (b *SimulatedBackend) SendTransaction(ctx context.Context, tx *types.Transaction) error {
b.mu.Lock()
defer b.mu.Unlock()

sender, err := types.Sender(types.HomesteadSigner{}, tx)
if err != nil {
panic(fmt.Errorf("invalid transaction: %v", err))
}
nonce := b.pendingState.GetNonce(sender)
if tx.Nonce() != nonce {
panic(fmt.Errorf("invalid transaction nonce: got %d, want %d", tx.Nonce(), nonce))
}

blocks, _ := core.GenerateChain(b.config, b.blockchain.CurrentBlock(), b.database, 1,
func(number int, block *core.BlockGen) {
for _, tx := range b.pendingBlock.Transactions() {
block.AddTx(tx)
}
block.AddTx(tx)
})
b.pendingBlock = blocks[0]
b.pendingState, _ = state.New(b.pendingBlock.Root(), state.NewDatabase(b.database))
return nil
}

// callmsg implements core.Message to allow passing it as a transaction simulator.
type callmsg struct {
ethereum.CallMsg
}
```

```
func (m callmsg) From() common.Address { return m.CallMsg.From }
func (m callmsg) Nonce() uint64        { return 0 }
func (m callmsg) CheckNonce() bool     { return false }
func (m callmsg) To() *common.Address  { return m.CallMsg.To }
func (m callmsg) GasPrice() *big.Int   { return m.CallMsg.GasPrice }
func (m callmsg) Gas() *big.Int        { return m.CallMsg.Gas }
func (m callmsg) Value() *big.Int      { return m.CallMsg.Value }
func (m callmsg) Data() []byte         { return m.CallMsg.Data }

6:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\base.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package bind

import (
"context"
"errors"
"fmt"
"math/big"

"github.com/ethereum/go-ethereum"
"github.com/ethereum/go-ethereum/accounts/abi"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
)

// SignerFn is a signer function callback when a contract requires a method to
// sign the transaction before submission.
type SignerFn func(types.Signer, common.Address, *types.Transaction) (*types.Transaction,
error)

// CallOpts is the collection of options to fine tune a contract call request.
type CallOpts struct {
Pending bool           // Whether to operate on the pending state or the last known one
From    common.Address // Optional the sender address, otherwise the first account is used

Context context.Context // Network context to support cancellation and timeouts (nil = no timeout)
}

// TransactOpts is the collection of authorization data required to create a
// valid Ethereum transaction.
```

```
type TransactOpts struct {
From   common.Address // Ethereum account to send the transaction from
Nonce  *big.Int       // Nonce to use for the transaction execution (nil = use pending state)
Signer SignerFn       // Method to use for signing the transaction (mandatory)

Value    *big.Int // Funds to transfer along along the transaction (nil = 0 = no funds)
GasPrice *big.Int // Gas price to use for the transaction execution (nil = gas price oracle)
GasLimit *big.Int // Gas limit to set for the transaction execution (nil = estimate + 10%)

Context context.Context // Network context to support cancellation and timeouts (nil = no timeout)
}

// BoundContract is the base wrapper object that reflects a contract on the
// Ethereum network. It contains a collection of methods that are used by the
// higher level contract bindings to operate.
type BoundContract struct {
address    common.Address     // Deployment address of the contract on the Ethereum blockchain
abi        abi.ABI            // Reflect based ABI to access the correct Ethereum methods
caller     ContractCaller     // Read interface to interact with the blockchain
transactor ContractTransactor // Write interface to interact with the blockchain
}

// NewBoundContract creates a low level contract interface through which calls
// and transactions may be made through.
func NewBoundContract(address common.Address, abi abi.ABI, caller ContractCaller, transactor
ContractTransactor) *BoundContract {
return &BoundContract{
address:    address,
abi:        abi,
caller:     caller,
transactor: transactor,
}
}

// DeployContract deploys a contract onto the Ethereum blockchain and binds the
// deployment address with a Go wrapper.
func DeployContract(opts *TransactOpts, abi abi.ABI, bytecode []byte, backend ContractBackend,
params ...interface{}) (common.Address, *types.Transaction, *BoundContract, error) {
// Otherwise try to deploy the contract
c := NewBoundContract(common.Address{}, abi, backend, backend)

input, err := c.abi.Pack("", params...)
```

```
if err != nil {
return common.Address{}, nil, nil, err
}
tx, err := c.transact(opts, nil, append(bytecode, input...))
if err != nil {
return common.Address{}, nil, nil, err
}
c.address = crypto.CreateAddress(opts.From, tx.Nonce())
return c.address, tx, c, 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 (c *BoundContract) Call(opts *CallOpts, result interface{}, method string, params
...interface{}) error {
// Don't crash on a lazy user
if opts == nil {
opts = new(CallOpts)
}
// Pack the input, call and unpack the results
input, err := c.abi.Pack(method, params...)
if err != nil {
return err
}
var (
msg    = ethereum.CallMsg{From: opts.From, To: &c.address, Data: input}
ctx    = ensureContext(opts.Context)
code   []byte
output []byte
)
if opts.Pending {
pb, ok := c.caller.(PendingContractCaller)
if !ok {
return ErrNoPendingState
}
output, err = pb.PendingCallContract(ctx, msg)
if err == nil && len(output) == 0 {
// Make sure we have a contract to operate on, and bail out otherwise.
if code, err = pb.PendingCodeAt(ctx, c.address); err != nil {
return err
```

```
} else if len(code) == 0 {
return ErrNoCode
}
}
} else {
output, err = c.caller.CallContract(ctx, msg, nil)
if err == nil && len(output) == 0 {
// Make sure we have a contract to operate on, and bail out otherwise.
if code, err = c.caller.CodeAt(ctx, c.address, nil); err != nil {
return err
} else if len(code) == 0 {
return ErrNoCode
}
}
}
if err != nil {
return err
}
return c.abi.Unpack(result, method, output)
}

// Transact invokes the (paid) contract method with params as input values.
func (c *BoundContract) Transact(opts *TransactOpts, method string, params ...interface{})
(*types.Transaction, error) {
// Otherwise pack up the parameters and invoke the contract
input, err := c.abi.Pack(method, params...)
if err != nil {
return nil, err
}
return c.transact(opts, &c.address, input)
}

// Transfer initiates a plain transaction to move funds to the contract, calling
// its default method if one is available.
func (c *BoundContract) Transfer(opts *TransactOpts) (*types.Transaction, error) {
return c.transact(opts, &c.address, nil)
}

// transact executes an actual transaction invocation, first deriving any missing
// authorization fields, and then scheduling the transaction for execution.
func (c *BoundContract) transact(opts *TransactOpts, contract *common.Address, input []byte)
(*types.Transaction, error) {
```

```
var err error

// Ensure a valid value field and resolve the account nonce
value := opts.Value
if value == nil {
value = new(big.Int)
}
var nonce uint64
if opts.Nonce == nil {
nonce, err = c.transactor.PendingNonceAt(ensureContext(opts.Context), opts.From)
if err != nil {
return nil, fmt.Errorf("failed to retrieve account nonce: %v", err)
}
} else {
nonce = opts.Nonce.Uint64()
}
// Figure out the gas allowance and gas price values
gasPrice := opts.GasPrice
if gasPrice == nil {
gasPrice, err = c.transactor.SuggestGasPrice(ensureContext(opts.Context))
if err != nil {
return nil, fmt.Errorf("failed to suggest gas price: %v", err)
}
}
gasLimit := opts.GasLimit
if gasLimit == nil {
// Gas estimation cannot succeed without code for method invocations
if contract != nil {
if code, err := c.transactor.PendingCodeAt(ensureContext(opts.Context), c.address); err != nil {
return nil, err
} else if len(code) == 0 {
return nil, ErrNoCode
}
}
// If the contract surely has code (or code is not needed), estimate the transaction
msg := ethereum.CallMsg{From: opts.From, To: contract, Value: value, Data: input}
gasLimit, err = c.transactor.EstimateGas(ensureContext(opts.Context), msg)
if err != nil {
return nil, fmt.Errorf("failed to estimate gas needed: %v", err)
}
}
// Create the transaction, sign it and schedule it for execution
```

```
var rawTx *types.Transaction
if contract == nil {
rawTx = types.NewContractCreation(nonce, value, gasLimit, gasPrice, input)
} else {
rawTx = types.NewTransaction(nonce, c.address, value, gasLimit, gasPrice, input)
}
if opts.Signer == nil {
return nil, errors.New("no signer to authorize the transaction with")
}
signedTx, err := opts.Signer(types.HomesteadSigner{}, opts.From, rawTx)
if err != nil {
return nil, err
}
if err := c.transactor.SendTransaction(ensureContext(opts.Context), signedTx); err != nil {
return nil, err
}
return signedTx, nil
}

func ensureContext(ctx context.Context) context.Context {
if ctx == nil {
return context.TODO()
}
return ctx
}

7:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\bind.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// Package bind generates Ethereum contract Go bindings.
//
// Detailed usage document and tutorial available on the go-ethereum Wiki page:
// https://github.com/ethereum/go-ethereum/wiki/Native-DApps:-Go-bindings-to-Ethereum-contracts
package bind

import (
"bytes"
"fmt"
"regexp"
"strings"
"text/template"
```

```
"unicode"

"github.com/ethereum/go-ethereum/accounts/abi"
"golang.org/x/tools/imports"
)

// Lang is a target programming language selector to generate bindings for.
type Lang int

const (
LangGo Lang = iota
LangJava
LangObjC
)

// Bind generates a Go wrapper around a contract ABI. This wrapper isn't meant
// to be used as is in client code, but rather as an intermediate struct which
// enforces compile time type safety and naming convention opposed to having to
// manually maintain hard coded strings that break on runtime.
func Bind(types []string, abis []string, bytecodes []string, pkg string, lang Lang) (string, error) {
// Process each individual contract requested binding
contracts := make(map[string]*tmplContract)

for i := 0; i < len(types); i++ {
// Parse the actual ABI to generate the binding for
evmABI, err := abi.JSON(strings.NewReader(abis[i]))
if err != nil {
return "", err
}
// Strip any whitespace from the JSON ABI
strippedABI := strings.Map(func(r rune) rune {
if unicode.IsSpace(r) {
return -1
}
return r
}, abis[i])

// Extract the call and transact methods, and sort them alphabetically
var (
calls     = make(map[string]*tmplMethod)
transacts = make(map[string]*tmplMethod)
)
```

```
for _, original := range evmABI.Methods {
// Normalize the method for capital cases and non-anonymous inputs/outputs
normalized := original
normalized.Name = methodNormalizer[lang](original.Name)

normalized.Inputs = make([]abi.Argument, len(original.Inputs))
copy(normalized.Inputs, original.Inputs)
for j, input := range normalized.Inputs {
if input.Name == "" {
normalized.Inputs[j].Name = fmt.Sprintf("arg%d", j)
}
}
normalized.Outputs = make([]abi.Argument, len(original.Outputs))
copy(normalized.Outputs, original.Outputs)
for j, output := range normalized.Outputs {
if output.Name != "" {
normalized.Outputs[j].Name = capitalise(output.Name)
}
}
// Append the methods to the call or transact lists
if original.Const {
calls[original.Name] = &tmplMethod{Original: original, Normalized: normalized, Structured:
structured(original)}
} else {
transacts[original.Name] = &tmplMethod{Original: original, Normalized: normalized, Structured:
structured(original)}
}
}
contracts[types[i]] = &tmplContract{
Type:        capitalise(types[i]),
InputABI:    strings.Replace(strippedABI, "\"", "\\\"", -1),
InputBin:    strings.TrimSpace(bytecodes[i]),
Constructor: evmABI.Constructor,
Calls:       calls,
Transacts:   transacts,
}
}
// Generate the contract template data content and render it
data := &tmplData{
Package:   pkg,
Contracts: contracts,
}
```

```
buffer := new(bytes.Buffer)

funcs := map[string]interface{}{
"bindtype":     bindType[lang],
"namedtype":    namedType[lang],
"capitalise":   capitalise,
"decapitalise": decapitalise,
}
tmpl := template.Must(template.New("").Funcs(funcs).Parse(tmplSource[lang]))
if err := tmpl.Execute(buffer, data); err != nil {
return "", err
}
// For Go bindings pass the code through goimports to clean it up and double check
if lang == LangGo {
code, err := imports.Process("", buffer.Bytes(), nil)
if err != nil {
return "", fmt.Errorf("%v\n%s", err, buffer)
}
return string(code), nil
}
// For all others just return as is for now
return string(buffer.Bytes()), nil
}

// bindType is a set of type binders that convert Solidity types to some supported
// programming language.
var bindType = map[Lang]func(kind abi.Type) string{
LangGo:   bindTypeGo,
LangJava: bindTypeJava,
}

// bindTypeGo converts a Solidity type to a Go one. Since there is no clear mapping
// from all Solidity types to Go ones (e.g. uint17), those that cannot be exactly
// mapped will use an upscaled type (e.g. *big.Int).
func bindTypeGo(kind abi.Type) string {
stringKind := kind.String()

switch {
case strings.HasPrefix(stringKind, "address"):
parts := regexp.MustCompile(`address(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 2 {
return stringKind
```

```
}
return fmt.Sprintf("%scommon.Address", parts[1])

case strings.HasPrefix(stringKind, "bytes"):
parts := regexp.MustCompile(`bytes([0-9]*)(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 3 {
return stringKind
}
return fmt.Sprintf("%s[%s]byte", parts[2], parts[1])

case strings.HasPrefix(stringKind, "int") || strings.HasPrefix(stringKind, "uint"):
parts := regexp.MustCompile(`(u)?int([0-9]*)(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 4 {
return stringKind
}
switch parts[2] {
case "8", "16", "32", "64":
return fmt.Sprintf("%s%sint%s", parts[3], parts[1], parts[2])
}
return fmt.Sprintf("%s*big.Int", parts[3])

case strings.HasPrefix(stringKind, "bool") || strings.HasPrefix(stringKind, "string"):
parts := regexp.MustCompile(`([a-z]+)(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 3 {
return stringKind
}
return fmt.Sprintf("%s%s", parts[2], parts[1])

default:
return stringKind
}
}

// bindTypeJava converts a Solidity type to a Java one. Since there is no clear mapping
// from all Solidity types to Java ones (e.g. uint17), those that cannot be exactly
// mapped will use an upscaled type (e.g. BigDecimal).
func bindTypeJava(kind abi.Type) string {
stringKind := kind.String()

switch {
case strings.HasPrefix(stringKind, "address"):
parts := regexp.MustCompile(`address(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
```

```
if len(parts) != 2 {
return stringKind
}
if parts[1] == "" {
return fmt.Sprintf("Address")
}
return fmt.Sprintf("Addresses")

case strings.HasPrefix(stringKind, "bytes"):
parts := regexp.MustCompile(`bytes([0-9]*)(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 3 {
return stringKind
}
if parts[2] != "" {
return "byte[][]"
}
return "byte[]"

case strings.HasPrefix(stringKind, "int") || strings.HasPrefix(stringKind, "uint"):
parts := regexp.MustCompile(`(u)?int([0-9]*)(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 4 {
return stringKind
}
switch parts[2] {
case "8", "16", "32", "64":
if parts[1] == "" {
if parts[3] == "" {
return fmt.Sprintf("int%s", parts[2])
}
return fmt.Sprintf("int%s[]", parts[2])
}
}
if parts[3] == "" {
return fmt.Sprintf("BigInt")
}
return fmt.Sprintf("BigInts")

case strings.HasPrefix(stringKind, "bool"):
parts := regexp.MustCompile(`bool(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 2 {
return stringKind
}
```

```
if parts[1] == "" {
return fmt.Sprintf("bool")
}
return fmt.Sprintf("bool[]")

case strings.HasPrefix(stringKind, "string"):
parts := regexp.MustCompile(`string(\[[0-9]*\])?`).FindStringSubmatch(stringKind)
if len(parts) != 2 {
return stringKind
}
if parts[1] == "" {
return fmt.Sprintf("String")
}
return fmt.Sprintf("String[]")

default:
return stringKind
}
}

// namedType is a set of functions that transform language specific types to
// named versions that my be used inside method names.
var namedType = map[Lang]func(string, abi.Type) string{
LangGo:   func(string, abi.Type) string { panic("this shouldn't be needed") },
LangJava: namedTypeJava,
}

// namedTypeJava converts some primitive data types to named variants that can
// be used as parts of method names.
func namedTypeJava(javaKind string, solKind abi.Type) string {
switch javaKind {
case "byte[]":
return "Binary"
case "byte[][]":
return "Binaries"
case "string":
return "String"
case "string[]":
return "Strings"
case "bool":
return "Bool"
case "bool[]":
```

```
return "Bools"
case "BigInt":
parts := regexp.MustCompile(`(u)?int([0-9]*)(\[[0-9]*\])?`).FindStringSubmatch(solKind.String())
if len(parts) != 4 {
return javaKind
}
switch parts[2] {
case "8", "16", "32", "64":
if parts[3] == "" {
return capitalise(fmt.Sprintf("%sint%s", parts[1], parts[2]))
}
return capitalise(fmt.Sprintf("%sint%ss", parts[1], parts[2]))

default:
return javaKind
}
default:
return javaKind
}
}

// methodNormalizer is a name transformer that modifies Solidity method names to
// conform to target language naming concentions.
var methodNormalizer = map[Lang]func(string) string{
LangGo:   capitalise,
LangJava: decapitalise,
}

// capitalise makes the first character of a string upper case.
func capitalise(input string) string {
return strings.ToUpper(input[:1]) + input[1:]
}

// decapitalise makes the first character of a string lower case.
func decapitalise(input string) string {
return strings.ToLower(input[:1]) + input[1:]
}

// structured checks whether a method has enough information to return a proper
// Go struct ot if flat returns are needed.
func structured(method abi.Method) bool {
if len(method.Outputs) < 2 {
```

```
return false
}
for _, out := range method.Outputs {
if out.Name == "" {
return false
}
}
return true
}

8:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\bind_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package bind

import (
"fmt"
"io/ioutil"
"os"
"os/exec"
"path/filepath"
"runtime"
"strings"
"testing"

"github.com/ethereum/go-ethereum/common"
"golang.org/x/tools/imports"
)

var bindTests = []struct {
name     string
contract string
bytecode string
abi      string
tester   string
}{
// Test that the binding is available in combined and separate forms too
{
`Empty`,
`contract NilContract {}`,
`6060604052600680601060000396000f3606060405200`,
`[]`,
```

`
if b, err := NewEmpty(common.Address{}, nil); b == nil || err != nil {
t.Fatalf("combined binding (%v) nil or error (%v) not nil", b, nil)
}
if b, err := NewEmptyCaller(common.Address{}, nil); b == nil || err != nil {
t.Fatalf("caller binding (%v) nil or error (%v) not nil", b, nil)
}
if b, err := NewEmptyTransactor(common.Address{}, nil); b == nil || err != nil {
t.Fatalf("transactor binding (%v) nil or error (%v) not nil", b, nil)
}
`,
},
// Test that all the official sample contracts bind correctly
{
`Token`,
`https://ethereum.org/token`,
`606060405260405161O7fd3803806107fd83398101604052805160805160a05160c05192939182
0192909101600160a060020a033316600090815260036020908152604082208690558154855183
8052601f6002600019610100600186161502019093169290920482018390047f290decd9548b62a
8d60345a988386fc84ba6bc95484008f6362f93160ef3e5639081019391929091880190839010610
0e857805160ff19168380011785555b506101189291505b80821115610171576000815560010161
00b4565b50506002805460ff19168317905550505050610658806101a56000396000f35b8280016
00101855582156100ac579182015b828111156100ac57825182600050559160200191906001019
06100fa565b5050806001600050908051906020019082805460018160011615610100020316600
2900490600052602060002090601f016020900481019282601f1061017557805160ff19168380011
785555b506100c89291506100b4565b5090565b82800160010185558215610165579182015b828
111156101655782518260005055916020019190600101906101875660606040523615610077576
0e060020a600035046306fdde03811461007f57806323b872dd146100dc578063313ce567146101
0e57806370a082311461011a57806395d89b4114610132578063a9059cbb1461018e578063cae9
ca51146101bd578063dc3080f21461031c578063dd62ed3e14610341575b610365610002565b610
36760008054602060026001831615610100026000190190921691909104601f810182900490910
260809081016040526060828152929190828280156104eb5780601f106104c0576101008083540
402835291602001916104eb565b6103d5600435602435604435600160a060020a0383166000908
15260036020526040812054829010156104f357610002565b6103e760025460ff1681565b6103d5
60043560036020526000908152604090205481565b610367600180546020600282841615610100
026000190190921691909104601f81018290049091026080908101604052606082815292919082
8280156104eb5780601f106104c0576101008083540402835291602001916104eb565b61036560
0435602435600160a060020a033316600090815260036020526040902054819010156103f15761
0002565b60806020604435600481810135601f8101849004909302840160405260608381526103
d59482359460248035956064949391019190819083828082843750949650505050505050600060
00836004600050600033600160a060020a03168152602001908152602001600020600050600087
600160a060020a031681526020019081526020016000206000508190555084905080600160a060

020a0316638f4ffcb133863087604051856 0e060020a028152600401808560016 0a060020a03168
152602001848152602001836001 60a060020a031681526020018060200182810382528381815 18
152602001915080519060200190808383829060006004602084601f0104600f02600301f1509050
90810190601f1680156102f25780820380516001836020036101000a031916815260200191505b5
095505050505050600060405180830381600087616 1da5a03f1156100025750505050939250505
0565b60056020908152600435600090815260408082209092526024358152205461 03d59081565
b60046020818152903560009081526040808220909252602435815220546103d59081565b005b6
040518080602001828103825283818151815260200191508051906020019080838382906000600
4602084601f0104600f02600301f150905090810190601f1680156103c75780820380516001836 02
0036101000a031916815260200191505b509250505060405180910390f35b60408051918252519
081900360200190f35b606090815260209 0f35b600160a060020a038216600090815260409020 54
808201101561041357610002565b806003600050600033600160a060020a031681526020019081
526020016000206000828282505403925050819055508060036000506000846001 60a060020a03
168152602001908152602001600020600082828250540192505081905550816 00160a060020a03
1633600160a060020a03167fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df
523b3ef836040518082815260200191505060405180910390a35050565b8201919060005260206
00020905b8154815290600101906020018083116104ce57829003601f168201915b50505050508
1565b600160a060020a038316815260408120548030110156105125761000 2565b600160a0600
20a038085168083526004602090815260408085203394909416808652938252808520549285526
005825280852093855292905290822054830111156105 5c57610002565b8160036000506000866
00160a060020a031681526020019081526020016000206000828282505403925050819055508 16
003600050600085600160a060020a031681526020019081526020016000206000828282505401 9
250508190555081600560005060008660016 0a060020a031681526020019081526020016000206
000506000336001 60a060020a031681526020019081526020016000206000828282505401925 05
081905555082600160a060020a031633600160a060020a03167fddf252ad1be2c89b69c2b068fc378
daa952ba7f163c4a11628f55a4df523b3ef8460405180828152602001915050604051809103 90a39
39250505056`,
`[{"constant":true,"inputs":[],"name":"name","outputs":[{"name":"","type":"string"}],"type":"function"},{
"constant":false,"inputs":[{"name":"_from","type":"address"},{"name":"_to","type":"address"},{"name
":"_value","type":"uint256"}],"name":"transferFrom","outputs":[{"name":"success","type":"bool"}],"typ
e":"function"},{"constant":true,"inputs":[],"name":"decimals","outputs":[{"name":"","type":"uint8"}],"ty
pe":"function"},{"constant":true,"inputs":[{"name":"","type":"address"}],"name":"balanceOf","outputs"
:[{"name":"","type":"uint256"}],"type":"function"},{"constant":true,"inputs":[],"name":"symbol","output
s":[{"name":"","type":"string"}],"type":"function"},{"constant":false,"inputs":[{"name":"_to","type":"add
ress"},{"name":"_value","type":"uint256"}],"name":"transfer","outputs":[],"type":"function"},{"constant
":false,"inputs":[{"name":"_spender","type":"address"},{"name":"_value","type":"uint256"},{"name":"
_extraData","type":"bytes"}],"name":"approveAndCall","outputs":[{"name":"success","type":"bool"}],
"type":"function"},{"constant":true,"inputs":[{"name":"","type":"address"},{"name":"","type":"address"
}],"name":"spentAllowance","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"constant":tru
e,"inputs":[{"name":"","type":"address"},{"name":"","type":"address"}],"name":"allowance","outputs":
[{"name":"","type":"uint256"}],"type":"function"},{"inputs":[{"name":"initialSupply","type":"uint256"},{"
name":"tokenName","type":"string"},{"name":"decimalUnits","type":"uint8"},{"name":"tokenSymbol",

"type":"string"}],"type":"constructor"},{"anonymous":false,"inputs":[{"indexed":true,"name":"from","ty
pe":"address"},{"indexed":true,"name":"to","type":"address"},{"indexed":false,"name":"value","type":
"uint256"}],"name":"Transfer","type":"event"}]`,
`
if b, err := NewToken(common.Address{}, nil); b == nil || err != nil {
t.Fatalf("binding (%v) nil or error (%v) not nil", b, nil)
}
`,
},
{
`Crowdsale`,
`https://ethereum.org/crowdsale`,
`6060604081905260007805460ff1916905560a0806105a8833961010060405290516080519151600
c05160e05160008054600160a060020a03199081169095178155670de0b6b3a7640000958602600
0155603c909302420160035593026004556005805490921690921790556105 2f90819061007990
396000f3606060405236156100 6c5760e060020a600035046301cb3b20811461008257806329dcb
0cf1461014457806338af3eed1461014d5780636e66f6e91461015f5780637a3a0e8414610171578
0637b3e5e7b1461017a578063a035b1fe14610183578063dc0d3dff1461018c575b6102006007546
0009060ff16156103235761000256 5b6102006003546000904210610320576002546001549010 61
03cb576002548154600160a060020a03169082906060828181858883f150915460025460408 0
51600160a060020a03939093168352602083019190915281810186905 2517fe842aea7a5f1b0104
9d752008c53c52890b1a6daf660cf39e8eec506112bbdf6945090819003909201919050a15b60405
160008054600160a060020a03908116923090911631908281818185 8883f150506007805460ff19
166001179055505050505 65b6103a160035481565b6103ab600054600160a060020a031681565b
6103ab600554600160a060020a031681565b6103a160015481565b6103a160025481565b6103a1
60045481565b6103be60043560068054829081101561000257506000526002027ff652222313e28
459528d920b65115c16c04f3efc82aaedc97be59f3f377c0d3f8101547ff652222313e28459528d920
b65115c16c04f3efc82aaedc97be59f3f377c0d40919091015460016 0a060020a039190911690825
65b005b505050815481101561000257906000526020600020906002020160005060008201518 16
000016000 6101000a815481600160a060020a030219169083021790555060208201518160010 16
000505590505080600260008282825054019250508190555060056000905490 6101000a9004600
160a060020a0316600160a060020a031663a9059cbb33600460005054840460405183 60e060020
a028152600401808360016 0a060020a03168152602001828152602001925050506000604051808
303816000876161da5a03f1156100025750506040805160016 0a060020a033316815260208101 8
490526001818301529051 7fe842aea7a5f1b01049d752008c53c52890b1a6daf660cf39e8eec5061
12bbdf692509081900360600190a15b50565b5060a060405233606090815234608081905260068
054600181018083559293928290828015829011610202576002028160020283600052602060002
091820191016102029190 5b8082111561039d57805473ffffffffffffffffffffffffffffffffffffffff1916815560006
001919091019081556 1036a565b5090565b6060908152602090f35b600160a060020a031660609
08152602090f35b6060918252608052604090f35b5b60065481101561010e576006805482908110
156100025760009182526002027ff652222313e28459528d920b65115c16c04f3efc82aaedc97be5
9f3f377c0d3f019060068054925460016 0a060020a03169284908110156100025760027ff65222

2313e28459528d920b65115c16c04f3efc82aaedc97be59f3f377c0d400154604051909150828181
81858883f19350505050507fe842aea7a5f1b01049d752008c53c52890b1a6daf660cf39e8eec5061
12bbdf6600660005082815481101561000257600082905260020 27ff652222313e28459528d920b
65115c16c04f3efc82aaedc97be59f3f377c0d3f01548154600160a060020a039190911691908490 8
110156100025760020 27ff652222313e28459528d920b65115c16c04f3efc82aaedc97be59f3f377c
0d400154604080516001 60160a060020a039490941684526020840191909152600083820152519182
90036060019190 50a16001016103cc56`,
`[{"constant":false,"inputs":[],"name":"checkGoalReached","outputs":[],"type":"function"},{"constant"
:true,"inputs":[],"name":"deadline","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"consta
nt":true,"inputs":[],"name":"beneficiary","outputs":[{"name":"","type":"address"}],"type":"function"},{"c
onstant":true,"inputs":[],"name":"tokenReward","outputs":[{"name":"","type":"address"}],"type":"funct
ion"},{"constant":true,"inputs":[],"name":"fundingGoal","outputs":[{"name":"","type":"uint256"}],"type"
:"function"},{"constant":true,"inputs":[],"name":"amountRaised","outputs":[{"name":"","type":"uint256
"}],"type":"function"},{"constant":true,"inputs":[],"name":"price","outputs":[{"name":"","type":"uint256"
}],"type":"function"},{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"funders","output
s":[{"name":"addr","type":"address"},{"name":"amount","type":"uint256"}],"type":"function"},{"inputs":
[{"name":"ifSuccessfulSendTo","type":"address"},{"name":"fundingGoalInEthers","type":"uint256"},{
"name":"durationInMinutes","type":"uint256"},{"name":"etherCostOfEachToken","type":"uint256"},{"
name":"addressOfTokenUsedAsReward","type":"address"}],"type":"constructor"},{"anonymous":fal
se,"inputs":[{"indexed":false,"name":"backer","type":"address"},{"indexed":false,"name":"amount","t
ype":"uint256"},{"indexed":false,"name":"isContribution","type":"bool"}],"name":"FundTransfer","typ
e":"event"}]`,
`
if b, err := NewCrowdsale(common.Address{}, nil); b == nil || err != nil {
t.Fatalf("binding (%v) nil or error (%v) not nil", b, nil)
}
`,
},
{
`DAO`,
`https://ethereum.org/dao`,
`60606040526040516080806114 5f833960e06040529051905160a05160c05160008054600160a0
60020a03191633179055600184815560028490556003839055600780549182018082558280158 2
90116100b857600302816003028360005260206000209182019101610 0b891906101c8565b5050
60609190910151600291909 0155600160a060020a038116600014 6100a65760008054600160a0
60020a031916821790555b505050506111f18061026e6000396000f35b5050604080516080810 18
252600080825260208281018290528351908101845281815292820192909252426060820152600
78054919450925081101561000257908 1527fa66cc928b5edb82af9bd49922954155ab7b0942694
bea4ce44661d9a8736c688905081518154602084810151740100000000000000000000000000000
0000000000002600160a060020a031992909216909217 60a060020a60ff021916178255604083 0
151805160018481018054600082815286902091956002938216156101000260001901909116929
09204601f9081018390048201949192919091019083901061023e57805160ff1916838001178555

5b5061007292915061022656 5b5050600060028201556001015b8082111561023a578054600160
a860020a0319168155600181810180546000808355926002908216156101000260001901909116
04601f81901061020c57506101bb565b601f0160209004906000526020600020908101906101bb9
1905b8082111561023a576000815560010161022656 5b5090565b828001600101855582156101a
f579182015b828111156101af57825182600050559160200191906001019061025056606060405 2
36156100b95760e060020a6000350463013cf08b81146100bb578063237e949214610128578063
391068211461028157806340 0e394914610299578063 5daf08ca146102a257806369bd34361461
032f5780638160f0b5146103385780638da5cb5b1461034157806396 44fcbd14610353578063aa02
a90f146103be578063b1050da5146103c7578063bcca1fd3146104b5578063d3c0715b146104dc5
78063eceb29451461058d578063f2fde38b1461067b575b005b61069c60043560048054829081 10
156100025790600052602060002090600a0201600050600581015481546001830154600384015 4
60048501546006860154600787015460016 0a060020a039590951697509295600201949193 60ff8
2811694610100909304169291908956 5b6040805160206024803560048181013560 1f810185900
485028601850190965285855261077595813595919460449492939092019181908401838280828
437509496505050505050506000600060046000508481548110156100025 7509052 7f8a35acfbc1
5ff81a39ae7d344fd709f28e8600b4aa8c65c6b64bfe7fe36bd19e600a84029081015 47f8a35acfbc1
5ff81a39ae7d344fd709f28e8600b4aa8c65c6b64bfe7fe36bd19b909101904210806101e65750600
481015460ff165b80610267575080600001600090549061010 00a9004600160a060020a03168160
010160005054846040518084600160a060020a0316606060020a028152601401838152602001 82
805190602001908083838290600060046020846 01f0104600f02600301f1509050019350505050 6
040518091039020816007016000505414155b80610277575060015460058201541 05b156110925
7610002565b6107756004356006602052600090815260409020548 1565b6107756005548 1565b6
1078760043560078054829081101561000257506000526003026000805160206111d1833981519
1528101547fa66cc928b5edb82af9bd49922954155ab7b0942694bea4ce44661d9a8736c68a8201
54600160a060020a0382169260a060020a90920460ff16917fa66cc928b5edb82af9bd4992295415
5ab7b0942694bea4ce44661d9a8736c689019084565b6107756002548 1565b6107756001548156
5b6108306000546001 60a060020a031681565b6040805160206044356004818 10135601f810184
900484028501840190955284845261 00b994813594602480359593946064949293910191819084
018382808284375094965050505050505060008054819060016 0a060020a039081163390911614
61084d57610002565b6107756003548 1565b6040805160206044356004818 10135601f81018490
048402850184019095528484526107759481359460248035959394606494929391019181908401
838280828437505060408051602097358 08a0135601f81018a90048a0283018a01909352828252
969897608497919650602490910194509092508291508401838280828437509496505050505050
5033600160a060020a0316600090815260066020526040812054819081148061 04ab5750604081
205460078054909190811015610002579082526003026000805160206111d18339815191520154
60a060020a900460ff16155b15610ce557610002565b6100b960043560243560443560005433600
160a060020a039081169116146 10b1857610002565b6040805160206044356004818 10135601f8
101849004840285018401909552848452610775948135946024803595939460649492939101918
190840183828082843750949650505050505050 33600160a060020a0316600090815260066020 5
260408120548190811480610583575060408120546007805490919081101561000257908252600
302600080516020621 1d183398151915201815054 60a060020a900460ff16155b15610f1d576100
02565b6040805160206064356004818 10135601f810184900484028501840190955284845261 07

75948135946024803595604435956084949201919081908401838280828437509496505050505050
50506000600060046000508681548110156100025790825260 0a027f8a35acfbc15ff81a39ae7d34
4fd709f28e8600b4aa8c65c6b64bfe7fe36bd19b018150905084848460405180846001 60a060020a
0316606060020a028152601401838152602001828051906020019080838382906000600460208 4
601f0104600f02600301f150905001935050505060405180910390208160070160005054149150 6
10cdc565b6100b960043560005433600160a060020a039081169116146 10f0857610002565b6040
51808a600160a060020a031681526020018981526020018060200188815260200187815260200 1
868152602001858152602001848152602001838152602001828103825289818154600181600116
156101000203166002900481526020019150805460018160011615610100020316600290048015
61075e5780601f1061073357610100808354040283529160200191 61075e565b82019190600052
6020600020905b8154815290600101906020018083116107415782900360 1f168201915b50509a
505050505050505050505060405180910390f35b604080519182525190819003602001 90f35b604
08051600160a060020a038616815260208101859052606081018390526080918101828152845460
0026001821615610100026000190190911604928201839052909160a083019085908015610 81e5
780601f106107f3576101008083540402835291602001916 1081e565b8201919060005260206000
20905b8154815290600101906020018083116108015782900360 1f168201915b50509550505050
505060405180910390f35b60408051600160a060020a039290921682525190819003602001 90f35
b600160a060020a038516600090815260066020526040812054141561 08a957604060002060078
054918290556001820180825582801582901161095c576003028160030283600052602060002091
8201910161095c9190610a4f565b600160a060020a038516600090815260066020526040902054
6007805491935090839081101561000257506000525060038102600080516020611 1d18339815
1915201805474ff0000000000000000000000000000000000000000191660a060020a8502178155
5b60408051600160a060020a038716815260208101869052815 17f27b022af4a8347100c7a041ce
5ccf8e14d644ff05de696315196faae8cd50c9b929181900390910190a15050505050565b5050509
150815060806040519081016040528086815260200185815260200184815260200142815260200
150600760005083815481101561000257906000526020600020906003020160005081518154602
08481015160a060020a02600160a060020a031992909216909217 74ff0000000000000000000000
0000000000000000001916178255604083015180516001848101805460008281528690209195 60
0293821615610100026000190190911692909204601f908101839004820194919291909101908 3
9010610ad357805160ff19168380011785555b50610b03929150610abb565b50506000600282015
56001015b80821115610acf57805474ffffffffffffffffffffffffffffffffffffffff1916815560018181018054600 0
8083559260029082161561010002600019019091160 4601f819010610aa15750610a42565b601f0
160209004906000526020600020908 10190610a4291905b80821115610acf57600081556001016
10abb565b5090565b828001600101855582 15610a36579182015b82811115610a3657825182600
0505591602001919060010190610ae5565b505060609190910151600291909101556109 11565b6
0018390556002829055600381905560408051848152602081018490528082018390529 0517fa43
9d3fa452be5e0e1e24a8145e715f4fd8b9c08c96a42fd82a855a85e5d57de9181900360600190a15
05050565b505085858460405180846001 60a060020a0316606060020a028152601401838152602
0018280519060200190808383829060006004602084601f0104600f02600301f150905001935050
505060405180910390208160070160005081905550600260005054603c0242018160030160005 0
8190555060008160040160006101000a81548160ff021916908302179055506000816004016001 6
101000a81548160ff021916908302179055506000816005016000508190555 07f646fec02522b41e

7125cfc859a64fd4f4cefd5dc3b6237ca0abe251ded1fa881828787876040518085815260200184600160a060020a031681526020018381526020018060200182810382528381815181526020019150805190602001908083838290600060046020846 01f0104600f02600301f150905090810190601f168015610cc4578082038051600183602003610100 0a031916815260200191505b50955050505050506040518091039 0a1600182016005555b50949350505050565b6004805460018101808355909190828015829011610d1c57600a0281600a028360005260206000209182019101610d1c9190610db8565b505060048054929450918491508110156100025790600052602060002090600a02016000508054600160a060020a031916871781556001818101879055855160028381018054600082815260209081902096975091959481161561010002600019011691909104601f908101829004840193918901908390106 10ed857805160ff1916838001178555 5b50610b6c929150610abb565b50506001015b80821115610acf578054600160a060020a031916815560006001828101829055600283810180548482559092811615610100026000190116046 01f819010610e9c57505b506000600383018190556004830180546 1ffff191690556005830181905560068301819055600783018190556008830180548282559082526020909120610db2916002028101905b80821115610acf57805474ffffffffffffffffffffffffffffffffffffffff191681556001818101805460008083559260029082161561010002600019019091160460 1f819010610eba57505b5050600101610e44565b601f0160209004906000526020600020908101906 10dfc9190610abb565b601f0160209004906000526020600020908101906 10e929190610abb565b828001600101855582156 10da6579182015b82811115610da6578251826000505591602001919060010190610eea565b60008054600160a060020a031916821790555056 5b60048054869081101561000257600091825260 0a027f8a35acfbc15ff81a39ae7d344fd709f28e8600b4aa8c65c6b64bfe7fe36bd19b01905033600160a060020a031660009081526009820160205260409020549091506 0ff1660011415610f8457610002565b33600160a060020a031660009081526009820160205260409020805460ff1916600190811790915560058201805490910190558315610fcd57600681018054600101905 5610fda565b60068101805460001901905 55b7fc34f869b7ff431b034b7b9aea9822dac189a685e0b015c7d1be3add3f89128e885853386604051808581526020018481526020018360016 0a060020a031681526020018060200182810382528381815181526020019150805190602001908083838290600060046020846 01f0104600f02600301f150905090810190601f16801561107a57808203805160018360200361010 0a031916815260200191505b5095505050505050506040518091039 0a1509392505050565b60068101546003549013156111585780600001600090549061010 00a9004600160a060020a0316600160a060020a031681600101600050546 70de0b6b3a7640000028460405180828051906020019080838382906000600460208460 1f0104600f02600301f150905090810190601f16801561112257808203805160018360200361010 0a031916815260200191505b509150506000604051808303818587618502 5a03f1505050506004810180546 0ff191660011761ff00191661010017905561116d565b60048101805460ff191660011761ff00191690555b600681015460058201546004830154604080518881526020810194909452838101929092526101009004 60ff166060830152517fd220b7272a8b6d0d7d6bcdace67b936a8f175e6d5c1b3ee438b72256b32ab3af9181900360800190a1509291505056a66cc928b5edb82af9bd49922954155ab7b0942694bea4ce44661d9a8736c688`,

`[{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"proposals","outputs":[{"name":"recipient","type":"address"},{"name":"amount","type":"uint256"},{"name":"description","type":"string"},{"name":"votingDeadline","type":"uint256"},{"name":"executed","type":"bool"},{"name":"proposalPassed","type":"bool"},{"name":"numberOfVotes","type":"uint256"},{"name":"currentResult","type":"int25

6"},{"name":"proposalHash","type":"bytes32"}],"type":"function"},{"constant":false,"inputs":[{"name":"proposalNumber","type":"uint256"},{"name":"transactionBytecode","type":"bytes"}],"name":"executeProposal","outputs":[{"name":"result","type":"int256"}],"type":"function"},{"constant":true,"inputs":[{"name":"","type":"address"}],"name":"memberId","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"constant":true,"inputs":[],"name":"numProposals","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"members","outputs":[{"name":"member","type":"address"},{"name":"canVote","type":"bool"},{"name":"name","type":"string"},{"name":"memberSince","type":"uint256"}],"type":"function"},{"constant":true,"inputs":[],"name":"debatingPeriodInMinutes","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"constant":true,"inputs":[],"name":"minimumQuorum","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"constant":true,"inputs":[],"name":"owner","outputs":[{"name":"","type":"address"}],"type":"function"},{"constant":false,"inputs":[{"name":"targetMember","type":"address"},{"name":"canVote","type":"bool"},{"name":"memberName","type":"string"}],"name":"changeMembership","outputs":[],"type":"function"},{"constant":true,"inputs":[],"name":"majorityMargin","outputs":[{"name":"","type":"int256"}],"type":"function"},{"constant":false,"inputs":[{"name":"beneficiary","type":"address"},{"name":"etherAmount","type":"uint256"},{"name":"JobDescription","type":"string"},{"name":"transactionBytecode","type":"bytes"}],"name":"newProposal","outputs":[{"name":"proposalID","type":"uint256"}],"type":"function"},{"constant":false,"inputs":[{"name":"minimumQuorumForProposals","type":"uint256"},{"name":"minutesForDebate","type":"uint256"},{"name":"marginOfVotesForMajority","type":"int256"}],"name":"changeVotingRules","outputs":[],"type":"function"},{"constant":false,"inputs":[{"name":"proposalNumber","type":"uint256"},{"name":"supportsProposal","type":"bool"},{"name":"justificationText","type":"string"}],"name":"vote","outputs":[{"name":"voteID","type":"uint256"}],"type":"function"},{"constant":true,"inputs":[{"name":"proposalNumber","type":"uint256"},{"name":"beneficiary","type":"address"},{"name":"etherAmount","type":"uint256"},{"name":"transactionBytecode","type":"bytes"}],"name":"checkProposalCode","outputs":[{"name":"codeChecksOut","type":"bool"}],"type":"function"},{"constant":false,"inputs":[{"name":"newOwner","type":"address"}],"name":"transferOwnership","outputs":[],"type":"function"},{"inputs":[{"name":"minimumQuorumForProposals","type":"uint256"},{"name":"minutesForDebate","type":"uint256"},{"name":"marginOfVotesForMajority","type":"int256"},{"name":"congressLeader","type":"address"}],"type":"constructor"},{"anonymous":false,"inputs":[{"indexed":false,"name":"proposalID","type":"uint256"},{"indexed":false,"name":"recipient","type":"address"},{"indexed":false,"name":"amount","type":"uint256"},{"indexed":false,"name":"description","type":"string"}],"name":"ProposalAdded","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"name":"proposalID","type":"uint256"},{"indexed":false,"name":"position","type":"bool"},{"indexed":false,"name":"voter","type":"address"},{"indexed":false,"name":"justification","type":"string"}],"name":"Voted","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"name":"proposalID","type":"uint256"},{"indexed":false,"name":"result","type":"int256"},{"indexed":false,"name":"quorum","type":"uint256"},{"indexed":false,"name":"active","type":"bool"}],"name":"ProposalTallied","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"name":"member","type":"address"},{"indexed":false,"name":"isMember","type":"bool"}],"name":"MembershipChanged","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"name":"minimumQuorum","type":"uint256"},{"indexed":false,"name":"debatingPeriodInMinutes","type":"uint256"},{"indexed":false,"name":"majorityMargin","type":"int256"}],"name":"ChangeOfRules","type":"event"}]`,

```
`
if b, err := NewDAO(common.Address{}, nil); b == nil || err != nil {
t.Fatalf("binding (%v) nil or error (%v) not nil", b, nil)
}
`,
},
// Test that named and anonymous inputs are handled correctly
{
`InputChecker`, ``, ``,
`
[
{"type":"function","name":"noInput","constant":true,"inputs":[],"outputs":[]},
{"type":"function","name":"namedInput","constant":true,"inputs":[{"name":"str","type":"string"}],"outputs":[]},
{"type":"function","name":"anonInput","constant":true,"inputs":[{"name":"","type":"string"}],"outputs":[]},
{"type":"function","name":"namedInputs","constant":true,"inputs":[{"name":"str1","type":"string"},{"name":"str2","type":"string"}],"outputs":[]},
{"type":"function","name":"anonInputs","constant":true,"inputs":[{"name":"","type":"string"},{"name":"","type":"string"}],"outputs":[]},
{"type":"function","name":"mixedInputs","constant":true,"inputs":[{"name":"","type":"string"},{"name":"str","type":"string"}],"outputs":[]}
]
`,
`if b, err := NewInputChecker(common.Address{}, nil); b == nil || err != nil {
 t.Fatalf("binding (%v) nil or error (%v) not nil", b, nil)
 } else if false { // Don't run, just compile and test types
 var err error

 err = b.NoInput(nil)
 err = b.NamedInput(nil, "")
 err = b.AnonInput(nil, "")
 err = b.NamedInputs(nil, "", "")
 err = b.AnonInputs(nil, "", "")
 err = b.MixedInputs(nil, "", "")

 fmt.Println(err)
 }`,
},
// Test that named and anonymous outputs are handled correctly
{
`OutputChecker`, ``, ``,
```

```
`
[
{"type":"function","name":"noOutput","constant":true,"inputs":[],"outputs":[]},
{"type":"function","name":"namedOutput","constant":true,"inputs":[],"outputs":[{"name":"str","type":"
string"}]},
{"type":"function","name":"anonOutput","constant":true,"inputs":[],"outputs":[{"name":"","type":"strin
g"}]},
{"type":"function","name":"namedOutputs","constant":true,"inputs":[],"outputs":[{"name":"str1","type
":"string"},{"name":"str2","type":"string"}]},
{"type":"function","name":"anonOutputs","constant":true,"inputs":[],"outputs":[{"name":"","type":"stri
ng"},{"name":"","type":"string"}]},
{"type":"function","name":"mixedOutputs","constant":true,"inputs":[],"outputs":[{"name":"","type":"stri
ng"},{"name":"str","type":"string"}]}
]
`,
`if b, err := NewOutputChecker(common.Address{}, nil); b == nil || err != nil {
 t.Fatalf("binding (%v) nil or error (%v) not nil", b, nil)
 } else if false { // Don't run, just compile and test types
 var str1, str2 string
 var err error

 err              = b.NoOutput(nil)
 str1, err        = b.NamedOutput(nil)
 str1, err        = b.AnonOutput(nil)
 res, _           := b.NamedOutputs(nil)
 str1, str2, err  = b.AnonOutputs(nil)
 str1, str2, err  = b.MixedOutputs(nil)

 fmt.Println(str1, str2, res.Str1, res.Str2, err)
 }`,
},
// Test that contract interactions (deploy, transact and call) generate working code
{
`Interactor`,
`
contract Interactor {
string public deployString;
string public transactString;

function Interactor(string str) {
  deployString = str;
}
```

```
function transact(string str) {
  transactString = str;
}
}
`,
`6060604052604051610328380380610328833981016040528051018060006000509080519060200190828054600181600116156101000203166002900490600052602060002090601f016020900481019282601f10608d57805160ff1916838001178555b50607c9291505b8082111560ba57838155600101606b565b50505061026a806100be6000396000f35b828001600101855582156064579182015b828111156064578251826000505591602001919060010190609e565b5090566060604052 60e060020a60003504630d86a0e181146100315780636874e8091461008d578063d736c513146100ea575b005b610190600180546020600282841615610100026000190190921691909104601f8101829004909102608090810160405260608281529291908282801561022957806 01f106101fe576101008083540402835291602001916102295 65b6101906000805460206002600183161561010002600019019092169190910460 1f81018290049091026080908101604052606082815292919082828015610229578 0601f106101fe5761010080835404028352916020019161022956 5b60206004803580820135601f810184900490930260809081016040526060848152 61002f9460249391929184019181908382808284375094965050505050505080600 1600050908051906020019082805460018160011615610100020316600290049060 0052602060002090601f016020900481019282601f10610231578051 60ff1916838001178555b50610261929150 5b808211156102665760008155830161 0617d565b6040518080602001828103825283818151815260200191508051906020019080838382906000600460208460 1f0104600f02600301f150905090810190601f1680156101f057808203805160018360200361010 00a031916815260200191505b509250505060405180910390f35b8201919060005260206000209 05b8154815290600101906020018083116102 0c57829003601f168201915b505050505081565b82800160010185558215610175579182015b8281111561017557825182600050559160200191906001019061024356 5b505050565b509056`,
`[{"constant":true,"inputs":[],"name":"transactString","outputs":[{"name":"","type":"string"}],"type":"function"},{"constant":true,"inputs":[],"name":"deployString","outputs":[{"name":"","type":"string"}],"type":"function"},{"constant":false,"inputs":[{"name":"str","type":"string"}],"name":"transact","outputs":[],"type":"function"},{"inputs":[{"name":"str","type":"string"}],"type":"constructor"}]`,
`
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
sim := backends.NewSimulatedBackend(core.GenesisAlloc{auth.From: {Balance: big.NewInt(10000000000)}})

// Deploy an interaction tester contract and call a transaction on it
_, _, interactor, err := DeployInteractor(auth, sim, "Deploy string")
if err != nil {
t.Fatalf("Failed to deploy interactor contract: %v", err)
```

```
}
if _, err := interactor.Transact(auth, "Transact string"); err != nil {
t.Fatalf("Failed to transact with interactor contract: %v", err)
}
// Commit all pending transactions in the simulator and check the contract state
sim.Commit()

if str, err := interactor.DeployString(nil); err != nil {
t.Fatalf("Failed to retrieve deploy string: %v", err)
} else if str != "Deploy string" {
t.Fatalf("Deploy string mismatch: have '%s', want 'Deploy string'", str)
}
if str, err := interactor.TransactString(nil); err != nil {
t.Fatalf("Failed to retrieve transact string: %v", err)
} else if str != "Transact string" {
t.Fatalf("Transact string mismatch: have '%s', want 'Transact string'", str)
}
`,
},
// Tests that plain values can be properly returned and deserialized
{
`Getter`,
`
contract Getter {
function getter() constant returns (string, int, bytes32) {
return ("Hi", 1, sha3(""));
}
}
`,
`606060405260dc8060106000396000f3606060405260e060020a6000350463993a04b78114601a
575b005b600060605260c0604052600260809081527f4869000000000000000000000000000000
00000000000000000000000000000060a05260017fc5d2460186f7233c927e7db2dcc703c0e500b
653ca82273b7bfad8045d85a47060e0829052610100819052606060c09081526002610120819052
81906101409060a09080838184600060046012f1505081517fffff000000000000000000000000000
000000000000000000000000000000001690915250506040516101608190039450925050 50f3`,
`[{"constant":true,"inputs":[],"name":"getter","outputs":[{"name":"","type":"string"},{"name":"","type":"i
nt256"},{"name":"","type":"bytes32"}],"type":"function"}]`,
`
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
sim := backends.NewSimulatedBackend(core.GenesisAlloc{auth.From: {Balance:
```

```
big.NewInt(10000000000)}})

// Deploy a tuple tester contract and execute a structured call on it
_, _, getter, err := DeployGetter(auth, sim)
if err != nil {
t.Fatalf("Failed to deploy getter contract: %v", err)
}
sim.Commit()

if str, num, _, err := getter.Getter(nil); err != nil {
t.Fatalf("Failed to call anonymous field retriever: %v", err)
} else if str != "Hi" || num.Cmp(big.NewInt(1)) != 0 {
t.Fatalf("Retrieved value mismatch: have %v/%v, want %v/%v", str, num, "Hi", 1)
}
`,
},
// Tests that tuples can be properly returned and deserialized
{
`Tupler`,
`
contract Tupler {
function tuple() constant returns (string a, int b, bytes32 c) {
return ("Hi", 1, sha3(""));
}
}
`,
`606060405260dc8060106000396000f3606060405260e060020a60003504633175aae28114601a
575b005b600060605260c0604052600260809081527f486900000000000000000000000000000000
000000000000000000000000000060a05260017fc5d2460186f7233c927e7db2dcc703c0e500b
653ca82273b7bfad8045d85a47060e082905261010081905260606060c0908152600261012081905 2
81906101409060a09080838184600060046012f1505081517fffff00000000000000000000000000
0000000000000000000000000000000001690915250506040516101608190039450925050 50f3`,
`[{"constant":true,"inputs":[],"name":"tuple","outputs":[{"name":"a","type":"string"},{"name":"b","type"
:"int256"},{"name":"c","type":"bytes32"}],"type":"function"}]`,
`
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
sim := backends.NewSimulatedBackend(core.GenesisAlloc{auth.From: {Balance:
big.NewInt(10000000000)}})

// Deploy a tuple tester contract and execute a structured call on it
```

```
_, _, tupler, err := DeployTupler(auth, sim)
if err != nil {
t.Fatalf("Failed to deploy tupler contract: %v", err)
}
sim.Commit()

if res, err := tupler.Tuple(nil); err != nil {
t.Fatalf("Failed to call structure retriever: %v", err)
} else if res.A != "Hi" || res.B.Cmp(big.NewInt(1)) != 0 {
t.Fatalf("Retrieved value mismatch: have %v/%v, want %v/%v", res.A, res.B, "Hi", 1)
}
`,
},
// Tests that arrays/slices can be properly returned and deserialized.
// Only addresses are tested, remainder just compiled to keep the test small.
{
`Slicer`,
`
contract Slicer {
function echoAddresses(address[] input) constant returns (address[] output) {
return input;
}
function echoInts(int[] input) constant returns (int[] output) {
return input;
}
function echoFancyInts(uint24[23] input) constant returns (uint24[23] output) {
return input;
}
function echoBools(bool[] input) constant returns (bool[] output) {
return input;
}
}
`,
`606060405261015c806100126000396000f3606060405260e060020a6000350463be1127a38114
61003c578063d88becc014610092578063e15a3db71461003c578063f637e5891461003c575b005
b604080516020600480358082013583810285810185019096528085526100ee959294602494909
3928501928291850190849080828437509496505050505050506040805160208101909152600090
052805b919050565b604080516102e0818101909252610138916004916102e4918390601790839
0839080828437509095505050505050506102e060405190810160405280601790 5b6000815260200
190600190039081610 0d1579050508190506100 8d565b60405180806020018281038252838181 5
181526020019150805190602001906020028083838290600060046020846 01f0104600f02600301
f150905001925050506040518091039 0f35b60405180826102e0808381846000600461015cf1509
```

```
05001915050604051809103 90f3`,
`[{"constant":true,"inputs":[{"name":"input","type":"address[]"}],"name":"echoAddresses","outputs":[{
"name":"output","type":"address[]"}],"type":"function"},{"constant":true,"inputs":[{"name":"input","typ
e":"uint24[23]"}],"name":"echoFancyInts","outputs":[{"name":"output","type":"uint24[23]"}],"type":"fu
nction"},{"constant":true,"inputs":[{"name":"input","type":"int256[]"}],"name":"echoInts","outputs":[{"n
ame":"output","type":"int256[]"}],"type":"function"},{"constant":true,"inputs":[{"name":"input","type":"
bool[]"}],"name":"echoBools","outputs":[{"name":"output","type":"bool[]"}],"type":"function"}]`,
`
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
sim := backends.NewSimulatedBackend(core.GenesisAlloc{auth.From: {Balance:
big.NewInt(10000000000)}})

// Deploy a slice tester contract and execute a n array call on it
_, _, slicer, err := DeploySlicer(auth, sim)
if err != nil {
t.Fatalf("Failed to deploy slicer contract: %v", err)
}
sim.Commit()

if out, err := slicer.EchoAddresses(nil, []common.Address{auth.From, common.Address{}}); err !=
nil {
t.Fatalf("Failed to call slice echoer: %v", err)
} else if !reflect.DeepEqual(out, []common.Address{auth.From, common.Address{}}) {
t.Fatalf("Slice return mismatch: have %v, want %v", out, []common.Address{auth.From,
common.Address{}})
}
`,
},
// Tests that anonymous default methods can be correctly invoked
{
`Defaulter`,
`
contract Defaulter {
address public caller;

function() {
caller = msg.sender;
}
}
`,
```

```
`6060604052606a8060106000396000f360606040523615601d5760e060020a6000350463fc9c8d
3981146040575b605e6000805473ffffffffffffffffffffffffffffffffffffffff19163179055565b606060005473ffff
ffffffffffffffffffffffffffffffffffffff1681565b005b6060908152602090f3`,
`[{"constant":true,"inputs":[],"name":"caller","outputs":[{"name":"","type":"address"}],"type":"function"
}]`,
`
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
sim := backends.NewSimulatedBackend(core.GenesisAlloc{auth.From: {Balance:
big.NewInt(10000000000)}})

// Deploy a default method invoker contract and execute its default method
_, _, defaulter, err := DeployDefaulter(auth, sim)
if err != nil {
t.Fatalf("Failed to deploy defaulter contract: %v", err)
}
if _, err := (&DefaulterRaw{defaulter}).Transfer(auth); err != nil {
t.Fatalf("Failed to invoke default method: %v", err)
}
sim.Commit()

if caller, err := defaulter.Caller(nil); err != nil {
t.Fatalf("Failed to call address retriever: %v", err)
} else if (caller != auth.From) {
t.Fatalf("Address mismatch: have %v, want %v", caller, auth.From)
}
`,
},
// Tests that non-existent contracts are reported as such (though only simulator test)
{
`NonExistent`,
`
contract NonExistent {
function String() constant returns(string) {
return "I don't exist";
}
}
`,
`6060604052609f8060106000396000f3606060405260e060020a6000350463f97a60058114601a5
75b005b600060605260c0604052600d60809081527f4920646f6e2774206578697374000000000000
0000000000000000000000000060a052602060c0908152600d60e081905281906101009060a0
```

```
9080838184600060046012f15050815172ffffffffffffffffffffffffffffffffffffffff19169091525050604051610120
81900392509050f3`,
`[{"constant":true,"inputs":[],"name":"String","outputs":[{"name":"","type":"string"}],"type":"function"}]
`,
`
// Create a simulator and wrap a non-deployed contract
sim := backends.NewSimulatedBackend(nil)

nonexistent, err := NewNonExistent(common.Address{}, sim)
if err != nil {
t.Fatalf("Failed to access non-existent contract: %v", err)
}
// Ensure that contract calls fail with the appropriate error
if res, err := nonexistent.String(nil); err == nil {
t.Fatalf("Call succeeded on non-existent contract: %v", res)
} else if (err != bind.ErrNoCode) {
t.Fatalf("Error mismatch: have %v, want %v", err, bind.ErrNoCode)
}
`,
},
// Tests that gas estimation works for contracts withweird gas mechanics too.
{
`FunkyGasPattern`,
`
contract FunkyGasPattern {
string public field;

function SetField(string value) {
// This check will screw gas estimation! Good, good!
if (msg.gas < 100000) {
throw;
}
field = value;
}
}
`,
`606060405261021c806100126000396000f3606060405260e060020a600035046323fcf32a81146
100265780634f28bf0e1461007b575b005b6040805160206004803580820135601f810184900484
0285018401909552848452610024949193602493909291840191908190840183828082843750949
650505050505050620186a05a101561014e57610002565b6100db60008054604080516020601f
600260001961010060018816150201909516949094049384018190048102820181019092528281
52929190830182828015610214578060lf106101e95761010080835404028352916020019161O2
```

14565b604051808060200182810382528381815181526020019150805190602001908083838290
60006004602084601f0104600302600f01f150905090810190601f16801561013b57808203805160
01836020036101000a031916815260200191505b509250505060405180910390f35b505050565b
80600060005090805190602001908280546001816001161561010002031660029004906000526
0200209060 1f016020900481019282601f106101b557805160ff1916838001178555b50610149
9291505b808211156101e557600081556001016101a1565b828001600101855582156101995791
82015b828111156101995782518260005055916020019190600101906101c7565b5090565b8201
91906000526020600020905b81548152906001019060200180831161 01f757829003601f1682019
15b50505050508156`,
`[{"constant":false,"inputs":[{"name":"value","type":"string"}],"name":"SetField","outputs":[],"type":"fu
nction"},{"constant":true,"inputs":[],"name":"field","outputs":[{"name":"","type":"string"}],"type":"functi
on"}]`,
`
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
sim := backends.NewSimulatedBackend(core.GenesisAlloc{auth.From: {Balance:
big.NewInt(10000000000)}})

// Deploy a funky gas pattern contract
_, _, limiter, err := DeployFunkyGasPattern(auth, sim)
if err != nil {
t.Fatalf("Failed to deploy funky contract: %v", err)
}
sim.Commit()

// Set the field with automatic estimation and check that it succeeds
auth.GasLimit = nil
if _, err := limiter.SetField(auth, "automatic"); err != nil {
t.Fatalf("Failed to call automatically gased transaction: %v", err)
}
sim.Commit()

if field, _ := limiter.Field(nil); field != "automatic" {
t.Fatalf("Field mismatch: have %v, want %v", field, "automatic")
}
`,
},
// Test that constant functions can be called from an (optional) specified address
{
`CallFrom`,
`

```
contract CallFrom {
function callFrom() constant returns(address) {
return msg.sender;
}
}
`,
`6060604052346000575b60868061001760003960000f30060606060405263ffffffff60e060020a600035
04166349f8e98281146022575b6000565b34600057602c6055565b6040805173ffffffffffffffffffffffffffff
ffffffffffff90921682525190819003602001900f35b335b905600a165627a7a72305820aef6b7685c0fa2
4ba6027e4870404a57df701473fe4107741805c19f5138417c0029`,
`[{"constant":true,"inputs":[],"name":"callFrom","outputs":[{"name":"","type":"address"}],"payable":fal
se,"type":"function"}]`,
`
// Generate a new random account and a funded simulator
key, _ := crypto.GenerateKey()
auth := bind.NewKeyedTransactor(key)
sim := backends.NewSimulatedBackend(core.GenesisAlloc{auth.From: {Balance:
big.NewInt(10000000000)}})

// Deploy a sender tester contract and execute a structured call on it
_, _, callfrom, err := DeployCallFrom(auth, sim)
if err != nil {
t.Fatalf("Failed to deploy sender contract: %v", err)
}
sim.Commit()

if res, err := callfrom.CallFrom(nil); err != nil {
t.Errorf("Failed to call constant function: %v", err)
} else if res != (common.Address{}) {
t.Errorf("Invalid address returned, want: %x, got: %x", (common.Address{}), res)
}

for _, addr := range []common.Address{common.Address{}, common.Address{1},
common.Address{2}} {
if res, err := callfrom.CallFrom(&bind.CallOpts{From: addr}); err != nil {
t.Fatalf("Failed to call constant function: %v", err)
} else if res != addr {
t.Fatalf("Invalid address returned, want: %x, got: %x", addr, res)
}
}
`,
},
```

```
}

// Tests that packages generated by the binder can be successfully compiled and
// the requested tester run against it.
func TestBindings(t *testing.T) {
// Skip the test if no Go command can be found
gocmd := runtime.GOROOT() + "/bin/go"
if !common.FileExist(gocmd) {
t.Skip("go sdk not found for testing")
}
// Skip the test if the go-ethereum sources are symlinked
(https://github.com/golang/go/issues/14845)
linkTestCode := fmt.Sprintf("package linktest\nfunc
CheckSymlinks(){\nfmt.Println(backends.NewSimulatedBackend(nil))\n}")
linkTestDeps, err := imports.Process("", []byte(linkTestCode), nil)
if err != nil {
t.Fatalf("failed check for goimports symlink bug: %v", err)
}
if !strings.Contains(string(linkTestDeps), "go-ethereum") {
t.Skip("symlinked environment doesn't support bind (https://github.com/golang/go/issues/14845)")
}
// Create a temporary workspace for the test suite
ws, err := ioutil.TempDir("", "")
if err != nil {
t.Fatalf("failed to create temporary workspace: %v", err)
}
defer os.RemoveAll(ws)

pkg := filepath.Join(ws, "bindtest")
if err = os.MkdirAll(pkg, 0700); err != nil {
t.Fatalf("failed to create package: %v", err)
}
// Generate the test suite for all the contracts
for i, tt := range bindTests {
// Generate the binding and create a Go source file in the workspace
bind, err := Bind([]string{tt.name}, []string{tt.abi}, []string{tt.bytecode}, "bindtest", LangGo)
if err != nil {
t.Fatalf("test %d: failed to generate binding: %v", i, err)
}
if err = ioutil.WriteFile(filepath.Join(pkg, strings.ToLower(tt.name)+".go"), []byte(bind), 0600); err !=
nil {
t.Fatalf("test %d: failed to write binding: %v", i, err)
```

```
}
// Generate the test file with the injected test code
code := fmt.Sprintf("package bindtest\nimport \"testing\"\nfunc Test%s(t *testing.T){\n%s\n}",
tt.name, tt.tester)
blob, err := imports.Process("", []byte(code), nil)
if err != nil {
t.Fatalf("test %d: failed to generate tests: %v", i, err)
}
if err := ioutil.WriteFile(filepath.Join(pkg, strings.ToLower(tt.name)+"_test.go"), blob, 0600); err !=
nil {
t.Fatalf("test %d: failed to write tests: %v", i, err)
}
}
// Test the entire package and report any failures
cmd := exec.Command(gocmd, "test", "-v")
cmd.Dir = pkg
if out, err := cmd.CombinedOutput(); err != nil {
t.Fatalf("failed to run binding test: %v\n%s", err, out)
}
}
```

9:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\template.go

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

package bind

import "github.com/ethereum/go-ethereum/accounts/abi"

// tmplData is the data structure required to fill the binding template.
type tmplData struct {
Package   string                 // Name of the package to place the generated file in
Contracts map[string]*tmplContract // List of contracts to generate into this file
}

// tmplContract contains the data needed to generate an individual contract binding.
type tmplContract struct {
Type        string              // Type name of the main contract binding
InputABI    string              // JSON ABI used as the input to generate the binding from
InputBin    string              // Optional EVM bytecode used to denetare deploy code from
Constructor abi.Method          // Contract constructor for deploy parametrization
Calls       map[string]*tmplMethod // Contract calls that only read state data
Transacts   map[string]*tmplMethod // Contract calls that write state data
```

```
}

// tmplMethod is a wrapper around an abi.Method that contains a few preprocessed
// and cached data fields.
type tmplMethod struct {
Original   abi.Method // Original method as parsed by the abi package
Normalized abi.Method // Normalized version of the parsed method (capitalized names, non-
anonymous args/returns)
Structured bool       // Whether the returns should be accumulated into a contract
}

// tmplSource is language to template mapping containing all the supported
// programming languages the package can generate to.
var tmplSource = map[Lang]string{
LangGo:   tmplSourceGo,
LangJava: tmplSourceJava,
}

// tmplSourceGo is the Go source template use to generate the contract binding
// based on.
const tmplSourceGo = `
// This file is an automatically generated Go binding. Do not modify as any
// change will likely be lost upon the next re-generation!

package {{.Package}}

{{range $contract := .Contracts}}
// {{.Type}}ABI is the input ABI used to generate the binding from.
const {{.Type}}ABI = "{{.InputABI}}"

{{if .InputBin}}
// {{.Type}}Bin is the compiled bytecode used for deploying new contracts.
const {{.Type}}Bin = ` + "`" + `{{.InputBin}}` + "`" + `

// Deploy{{.Type}} deploys a new Ethereum contract, binding an instance of {{.Type}} to it.
func Deploy{{.Type}}(auth *bind.TransactOpts, backend bind.ContractBackend {{range
.Constructor.Inputs}}, {{.Name}} {{bindtype .Type}}{{end}}) (common.Address, *types.Transaction,
*{{.Type}}, error) {
  parsed, err := abi.JSON(strings.NewReader({{.Type}}ABI))
  if err != nil {
    return common.Address{}, nil, nil, err
  }
```

```
  address, tx, contract, err := bind.DeployContract(auth, parsed, common.FromHex({{.Type}}Bin),
backend {{range .Constructor.Inputs}}, {{.Name}}{{end}})
  if err != nil {
    return common.Address{}, nil, nil, err
  }
  return address, tx, &{{.Type}}{ {{.Type}}Caller: {{.Type}}Caller{contract: contract},
{{.Type}}Transactor: {{.Type}}Transactor{contract: contract} }, nil
}
{{end}}

// {{.Type}} is an auto generated Go binding around an Ethereum contract.
type {{.Type}} struct {
  {{.Type}}Caller     // Read-only binding to the contract
  {{.Type}}Transactor // Write-only binding to the contract
}

// {{.Type}}Caller is an auto generated read-only Go binding around an Ethereum contract.
type {{.Type}}Caller struct {
  contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// {{.Type}}Transactor is an auto generated write-only Go binding around an Ethereum contract.
type {{.Type}}Transactor struct {
  contract *bind.BoundContract // Generic contract wrapper for the low level calls
}

// {{.Type}}Session is an auto generated Go binding around an Ethereum contract,
// with pre-set call and transact options.
type {{.Type}}Session struct {
  Contract     *{{.Type}}        // 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
}

// {{.Type}}CallerSession is an auto generated read-only Go binding around an Ethereum contract,
// with pre-set call options.
type {{.Type}}CallerSession struct {
  Contract *{{.Type}}Caller // Generic contract caller binding to set the session for
  CallOpts bind.CallOpts    // Call options to use throughout this session
}

// {{.Type}}TransactorSession is an auto generated write-only Go binding around an Ethereum
```

```
contract,
// with pre-set transact options.
type {{.Type}}TransactorSession struct {
  Contract     *{{.Type}}Transactor // Generic contract transactor binding to set the session for
  TransactOpts bind.TransactOpts    // Transaction auth options to use throughout this session
}

// {{.Type}}Raw is an auto generated low-level Go binding around an Ethereum contract.
type {{.Type}}Raw struct {
  Contract *{{.Type}} // Generic contract binding to access the raw methods on
}

// {{.Type}}CallerRaw is an auto generated low-level read-only Go binding around an Ethereum
contract.
type {{.Type}}CallerRaw struct {
Contract *{{.Type}}Caller // Generic read-only contract binding to access the raw methods on
}

// {{.Type}}TransactorRaw is an auto generated low-level write-only Go binding around an
Ethereum contract.
type {{.Type}}TransactorRaw struct {
Contract *{{.Type}}Transactor // Generic write-only contract binding to access the raw methods on
}

// New{{.Type}} creates a new instance of {{.Type}}, bound to a specific deployed contract.
func New{{.Type}}(address common.Address, backend bind.ContractBackend) (*{{.Type}}, error) {
  contract, err := bind{{.Type}}(address, backend, backend)
  if err != nil {
    return nil, err
  }
  return &{{.Type}}{ {{.Type}}Caller: {{.Type}}Caller{contract: contract}, {{.Type}}Transactor:
{{.Type}}Transactor{contract: contract} }, nil
}

// New{{.Type}}Caller creates a new read-only instance of {{.Type}}, bound to a specific deployed
contract.
func New{{.Type}}Caller(address common.Address, caller bind.ContractCaller) (*{{.Type}}Caller,
error) {
  contract, err := bind{{.Type}}(address, caller, nil)
  if err != nil {
    return nil, err
  }
```

```
  return &{{.Type}}Caller{contract: contract}, nil
}

// New{{.Type}}Transactor creates a new write-only instance of {{.Type}}, bound to a specific
deployed contract.
func New{{.Type}}Transactor(address common.Address, transactor bind.ContractTransactor)
(*{{.Type}}Transactor, error) {
  contract, err := bind{{.Type}}(address, nil, transactor)
  if err != nil {
    return nil, err
  }
  return &{{.Type}}Transactor{contract: contract}, nil
}

// bind{{.Type}} binds a generic wrapper to an already deployed contract.
func bind{{.Type}}(address common.Address, caller bind.ContractCaller, transactor
bind.ContractTransactor) (*bind.BoundContract, error) {
  parsed, err := abi.JSON(strings.NewReader({{.Type}}ABI))
  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 (_{{$contract.Type}} *{{$contract.Type}}Raw) Call(opts *bind.CallOpts, result interface{},
method string, params ...interface{}) error {
return _{{$contract.Type}}.Contract.{{$contract.Type}}Caller.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 (_{{$contract.Type}} *{{$contract.Type}}Raw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _{{$contract.Type}}.Contract.{{$contract.Type}}Transactor.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
```

```
func (_{{$contract.Type}} *{{$contract.Type}}Raw) Transact(opts *bind.TransactOpts, method
string, params ...interface{}) (*types.Transaction, error) {
return _{{$contract.Type}}.Contract.{{$contract.Type}}Transactor.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 (_{{$contract.Type}} *{{$contract.Type}}CallerRaw) Call(opts *bind.CallOpts, result interface{},
method string, params ...interface{}) error {
return _{{$contract.Type}}.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 (_{{$contract.Type}} *{{$contract.Type}}TransactorRaw) Transfer(opts *bind.TransactOpts)
(*types.Transaction, error) {
return _{{$contract.Type}}.Contract.contract.Transfer(opts)
}

// Transact invokes the (paid) contract method with params as input values.
func (_{{$contract.Type}} *{{$contract.Type}}TransactorRaw) Transact(opts *bind.TransactOpts,
method string, params ...interface{}) (*types.Transaction, error) {
return _{{$contract.Type}}.Contract.contract.Transact(opts, method, params...)
}

{{range .Calls}}
// {{.Normalized.Name}} is a free data retrieval call binding the contract method 0x{{printf "%x"
.Original.Id}}.
//
// Solidity: {{.Original.String}}
func (_{{$contract.Type}} *{{$contract.Type}}Caller) {{.Normalized.Name}}(opts *bind.CallOpts
{{range .Normalized.Inputs}}, {{.Name}} {{bindtype .Type}} {{end}}) ({{if .Structured}}struct{ {{range
.Normalized.Outputs}}{{.Name}} {{bindtype .Type}};{{end}} },{{else}}{{range
.Normalized.Outputs}}{{bindtype .Type}},{{end}}{{end}} error) {
{{if .Structured}}ret := new(struct{
{{range .Normalized.Outputs}}{{.Name}} {{bindtype .Type}}
{{end}}
}){{else}}var (
{{range $i, $_ := .Normalized.Outputs}}ret{{$i}} = new({{bindtype .Type}})
```

```
{{end}}
){{end}}
out := {{if .Structured}}ret{{else}}{{if eq (len .Normalized.Outputs) 1}}ret0{{else}}&[]interface{}{
{{range $i, $_ := .Normalized.Outputs}}ret{{$i}},
{{end}}
}{{end}}{{end}}
err := _{{$contract.Type}}.contract.Call(opts, out, "{{.Original.Name}}" {{range .Normalized.Inputs}}, {{.Name}}{{end}})
return {{if .Structured}}*ret,{{else}}{{range $i, $_ := .Normalized.Outputs}}*ret{{$i}},{{end}}{{end}} err
}

// {{.Normalized.Name}} is a free data retrieval call binding the contract method 0x{{printf "%x" .Original.Id}}.
//
// Solidity: {{.Original.String}}
func (_{{$contract.Type}} *{{$contract.Type}}Session) {{.Normalized.Name}}({{range $i, $_ := .Normalized.Inputs}}{{if ne $i 0}},{{end}} {{.Name}} {{bindtype .Type}} {{end}}) ({{if .Structured}}struct{ {{range .Normalized.Outputs}}{{.Name}} {{bindtype .Type}};{{end}} }, {{else}}{{range .Normalized.Outputs}}{{bindtype .Type}},{{end}} {{end}} error) {
  return _{{$contract.Type}}.Contract.{{.Normalized.Name}}(&_{{$contract.Type}}.CallOpts {{range .Normalized.Inputs}}, {{.Name}}{{end}})
}

// {{.Normalized.Name}} is a free data retrieval call binding the contract method 0x{{printf "%x" .Original.Id}}.
//
// Solidity: {{.Original.String}}
func (_{{$contract.Type}} *{{$contract.Type}}CallerSession) {{.Normalized.Name}}({{range $i, $_ := .Normalized.Inputs}}{{if ne $i 0}},{{end}} {{.Name}} {{bindtype .Type}} {{end}}) ({{if .Structured}}struct{ {{range .Normalized.Outputs}}{{.Name}} {{bindtype .Type}};{{end}} }, {{else}}{{range .Normalized.Outputs}}{{bindtype .Type}},{{end}} {{end}} error) {
  return _{{$contract.Type}}.Contract.{{.Normalized.Name}}(&_{{$contract.Type}}.CallOpts {{range .Normalized.Inputs}}, {{.Name}}{{end}})
}
{{end}}

{{range .Transacts}}
// {{.Normalized.Name}} is a paid mutator transaction binding the contract method 0x{{printf "%x" .Original.Id}}.
//
// Solidity: {{.Original.String}}
func (_{{$contract.Type}} *{{$contract.Type}}Transactor) {{.Normalized.Name}}(opts
```

```
*bind.TransactOpts {{range .Normalized.Inputs}}, {{.Name}} {{bindtype .Type}} {{end}})
(*types.Transaction, error) {
return _{{$contract.Type}}.contract.Transact(opts, "{{.Original.Name}}" {{range
.Normalized.Inputs}}, {{.Name}}{{end}})
}

// {{.Normalized.Name}} is a paid mutator transaction binding the contract method 0x{{printf "%x"
.Original.Id}}.
//
// Solidity: {{.Original.String}}
func (_{{$contract.Type}} *{{$contract.Type}}Session) {{.Normalized.Name}}({{range $i, $_ :=
.Normalized.Inputs}}{{if ne $i 0}},{{end}} {{.Name}} {{bindtype .Type}} {{end}}) (*types.Transaction,
error) {
  return _{{$contract.Type}}.Contract.{{.Normalized.Name}}(&_{{$contract.Type}}.TransactOpts
{{range $i, $_ := .Normalized.Inputs}}, {{.Name}}{{end}})
}

// {{.Normalized.Name}} is a paid mutator transaction binding the contract method 0x{{printf "%x"
.Original.Id}}.
//
// Solidity: {{.Original.String}}
func (_{{$contract.Type}} *{{$contract.Type}}TransactorSession) {{.Normalized.Name}}({{range $i,
$_ := .Normalized.Inputs}}{{if ne $i 0}},{{end}} {{.Name}} {{bindtype .Type}} {{end}})
(*types.Transaction, error) {
  return _{{$contract.Type}}.Contract.{{.Normalized.Name}}(&_{{$contract.Type}}.TransactOpts
{{range $i, $_ := .Normalized.Inputs}}, {{.Name}}{{end}})
}
{{end}}
{{end}}
`

// tmplSourceJava is the Java source template use to generate the contract binding
// based on.
const tmplSourceJava = `
// This file is an automatically generated Java binding. Do not modify as any
// change will likely be lost upon the next re-generation!

package {{.Package}};

import org.ethereum.geth.*;
import org.ethereum.geth.internal.*;
```

```
{{range $contract := .Contracts}}
public class {{.Type}} {
// ABI is the input ABI used to generate the binding from.
public final static String ABI = "{{.InputABI}}";

{{if .InputBin}}
// BYTECODE is the compiled bytecode used for deploying new contracts.
public final static byte[] BYTECODE = "{{.InputBin}}".getBytes();

// deploy deploys a new Ethereum contract, binding an instance of {{.Type}} to it.
public static {{.Type}} deploy(TransactOpts auth, EthereumClient client{{range
.Constructor.Inputs}}, {{bindtype .Type}} {{.Name}}{{end}}) throws Exception {
Interfaces args = Geth.newInterfaces({{(len .Constructor.Inputs)}});
{{range $index, $element := .Constructor.Inputs}}
  args.set({{$index}}, Geth.newInterface()); args.get({{$index}}).set{{namedtype (bindtype .Type)
.Type}}({{.Name}});
{{end}}
return new {{.Type}}(Geth.deployContract(auth, ABI, BYTECODE, client, args));
}

// Internal constructor used by contract deployment.
private {{.Type}}(BoundContract deployment) {
this.Address  = deployment.getAddress();
this.Deployer = deployment.getDeployer();
this.Contract = deployment;
}
{{end}}

// Ethereum address where this contract is located at.
public final Address Address;

// Ethereum transaction in which this contract was deployed (if known!).
public final Transaction Deployer;

// Contract instance bound to a blockchain address.
private final BoundContract Contract;

// Creates a new instance of {{.Type}}, bound to a specific deployed contract.
public {{.Type}}(Address address, EthereumClient client) throws Exception {
this(Geth.bindContract(address, ABI, client));
}
```

```
{{range .Calls}}
{{if gt (len .Normalized.Outputs) 1}}
// {{capitalise .Normalized.Name}}Results is the output of a call to {{.Normalized.Name}}.
public class {{capitalise .Normalized.Name}}Results {
{{range $index, $item := .Normalized.Outputs}}public {{bindtype .Type}} {{if ne .Name
""}}{{.Name}}{{else}}Return{{$index}}{{end}};
{{end}}
}
{{end}}

// {{.Normalized.Name}} is a free data retrieval call binding the contract method 0x{{printf "%x"
.Original.Id}}.
//
// Solidity: {{.Original.String}}
public {{if gt (len .Normalized.Outputs) 1}}{{capitalise .Normalized.Name}}Results{{else}}{{range
.Normalized.Outputs}}{{bindtype .Type}}{{end}}{{end}} {{.Normalized.Name}}(CallOpts opts{{range
.Normalized.Inputs}}, {{bindtype .Type}} {{.Name}}{{end}}) throws Exception {
Interfaces args = Geth.newInterfaces({{(len .Normalized.Inputs)}});
{{range $index, $item := .Normalized.Inputs}}args.set({{$index}}, Geth.newInterface());
args.get({{$index}}).set{{namedtype (bindtype .Type) .Type}}({{.Name}});
{{end}}

Interfaces results = Geth.newInterfaces({{(len .Normalized.Outputs)}});
{{range $index, $item := .Normalized.Outputs}}Interface result{{$index}} = Geth.newInterface();
result{{$index}}.setDefault{{namedtype (bindtype .Type) .Type}}(); results.set({{$index}},
result{{$index}});
{{end}}

if (opts == null) {
opts = Geth.newCallOpts();
}
this.Contract.call(opts, results, "{{.Original.Name}}", args);
{{if gt (len .Normalized.Outputs) 1}}
{{capitalise .Normalized.Name}}Results result = new {{capitalise .Normalized.Name}}Results();
{{range $index, $item := .Normalized.Outputs}}result.{{if ne .Name
""}}{{.Name}}{{else}}Return{{$index}}{{end}} = results.get({{$index}}).get{{namedtype (bindtype
.Type) .Type}}();
{{end}}
return result;
{{else}}{{range .Normalized.Outputs}}return results.get(0).get{{namedtype (bindtype .Type)
.Type}}();{{end}}
{{end}}
```

```
}
{{end}}

{{range .Transacts}}
// {{.Normalized.Name}} is a paid mutator transaction binding the contract method 0x{{printf "%x" .Original.Id}}.
//
// Solidity: {{.Original.String}}
public Transaction {{.Normalized.Name}}(TransactOpts opts{{range .Normalized.Inputs}}, {{bindtype .Type}} {{.Name}}{{end}}) throws Exception {
Interfaces args = Geth.newInterfaces({{(len .Normalized.Inputs)}});
{{range $index, $item := .Normalized.Inputs}}args.set({{$index}}, Geth.newInterface());
args.get({{$index}}).set{{namedtype (bindtype .Type) .Type}}({{.Name}});
{{end}}

return this.Contract.transact(opts, "{{.Original.Name}}", args);
}
{{end}}
}
{{end}}
`


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

package bind

import (
"context"
"fmt"
"time"

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

// WaitMined waits for tx to be mined on the blockchain.
// It stops waiting when the context is canceled.
func WaitMined(ctx context.Context, b DeployBackend, tx *types.Transaction) (*types.Receipt, error) {
queryTicker := time.NewTicker(time.Second)
```

```
defer queryTicker.Stop()

logger := log.New("hash", tx.Hash())
for {
receipt, err := b.TransactionReceipt(ctx, tx.Hash())
if receipt != nil {
return receipt, nil
}
if err != nil {
logger.Trace("Receipt retrieval failed", "err", err)
} else {
logger.Trace("Transaction not yet mined")
}
// Wait for the next round.
select {
case <-ctx.Done():
return nil, ctx.Err()
case <-queryTicker.C:
}
}
}

// WaitDeployed waits for a contract deployment transaction and returns the on-chain
// contract address when it is mined. It stops waiting when ctx is canceled.
func WaitDeployed(ctx context.Context, b DeployBackend, tx *types.Transaction)
(common.Address, error) {
if tx.To() != nil {
return common.Address{}, fmt.Errorf("tx is not contract creation")
}
receipt, err := WaitMined(ctx, b, tx)
if err != nil {
return common.Address{}, err
}
if receipt.ContractAddress == (common.Address{}) {
return common.Address{}, fmt.Errorf("zero address")
}
// Check that code has indeed been deployed at the address.
// This matters on pre-Homestead chains: OOG in the constructor
// could leave an empty account behind.
code, err := b.CodeAt(ctx, receipt.ContractAddress, nil)
if err == nil && len(code) == 0 {
err = ErrNoCodeAfterDeploy
```

```
}
return receipt.ContractAddress, err
}

11:F:\git\coin\ethereum\go-ethereum\accounts\abi\bind\util_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package bind_test

import (
"context"
"math/big"
"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/core"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
)

var testKey, _ =
crypto.HexToECDSA("b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f
291")

var waitDeployedTests = map[string]struct {
code        string
gas         *big.Int
wantAddress common.Address
wantErr     error
}{
"successful deploy": {
code:        `6060604052600a8060106000396000f360606040526008565b00`,
gas:         big.NewInt(3000000),
wantAddress: common.HexToAddress("0x3a220f351252089d385b29beca14e27f204c296a"),
},
"empty code": {
code:        ``,
gas:         big.NewInt(300000),
wantErr:     bind.ErrNoCodeAfterDeploy,
```

```
wantAddress: common.HexToAddress("0x3a220f351252089d385b29beca14e27f204c296a"),
},
}

func TestWaitDeployed(t *testing.T) {
for name, test := range waitDeployedTests {
backend := backends.NewSimulatedBackend(core.GenesisAlloc{
crypto.PubkeyToAddress(testKey.PublicKey): {Balance: big.NewInt(10000000000)},
})

// Create the transaction.
tx := types.NewContractCreation(0, big.NewInt(0), test.gas, big.NewInt(1),
common.FromHex(test.code))
tx, _ = types.SignTx(tx, types.HomesteadSigner{}, testKey)

// Wait for it to get mined in the background.
var (
err     error
address common.Address
mined   = make(chan struct{})
ctx     = context.Background()
)
go func() {
address, err = bind.WaitDeployed(ctx, backend, tx)
close(mined)
}()

// Send and mine the transaction.
backend.SendTransaction(ctx, tx)
backend.Commit()

select {
case <-mined:
if err != test.wantErr {
t.Errorf("test %q: error mismatch: got %q, want %q", name, err, test.wantErr)
}
if address != test.wantAddress {
t.Errorf("test %q: unexpected contract address %s", name, address.Hex())
}
case <-time.After(2 * time.Second):
t.Errorf("test %q: timeout", name)
}
```

```
}
}
```

12:F:\git\coin\ethereum\go-ethereum\accounts\abi\doc.go

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

// Package abi implements the Ethereum ABI (Application Binary
// Interface).
//
// The Ethereum ABI is strongly typed, known at compile time
// and static. This ABI will handle basic type casting; unsigned
// to signed and visa versa. It does not handle slice casting such
// as unsigned slice to signed slice. Bit size type casting is also
// handled. ints with a bit size of 32 will be properly cast to int256,
// etc.
package abi
```

13:F:\git\coin\ethereum\go-ethereum\accounts\abi\error.go

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

package abi

import (
"errors"
"fmt"
"reflect"
)

var (
errBadBool = errors.New("abi: improperly encoded boolean value")
)

// formatSliceString formats the reflection kind with the given slice size
// and returns a formatted string representation.
func formatSliceString(kind reflect.Kind, sliceSize int) string {
if sliceSize == -1 {
return fmt.Sprintf("[]%v", kind)
}
return fmt.Sprintf("[%d]%v", sliceSize, kind)
}

// sliceTypeCheck checks that the given slice can by assigned to the reflection
```

```
// type in t.
func sliceTypeCheck(t Type, val reflect.Value) error {
if val.Kind() != reflect.Slice && val.Kind() != reflect.Array {
return typeErr(formatSliceString(t.Kind, t.SliceSize), val.Type())
}
if t.IsArray && val.Len() != t.SliceSize {
return typeErr(formatSliceString(t.Elem.Kind, t.SliceSize),
formatSliceString(val.Type().Elem().Kind(), val.Len()))
}

if t.Elem.IsSlice {
if val.Len() > 0 {
return sliceTypeCheck(*t.Elem, val.Index(0))
}
} else if t.Elem.IsArray {
return sliceTypeCheck(*t.Elem, val.Index(0))
}

if elemKind := val.Type().Elem().Kind(); elemKind != t.Elem.Kind {
return typeErr(formatSliceString(t.Elem.Kind, t.SliceSize), val.Type())
}
return nil
}

// typeCheck checks that the given reflection value can be assigned to the reflection
// type in t.
func typeCheck(t Type, value reflect.Value) error {
if t.IsSlice || t.IsArray {
return sliceTypeCheck(t, value)
}

// Check base type validity. Element types will be checked later on.
if t.Kind != value.Kind() {
return typeErr(t.Kind, value.Kind())
}
return nil
}

// varErr returns a formatted error.
func varErr(expected, got reflect.Kind) error {
return typeErr(expected, got)
}
```

```
// typeErr returns a formatted type casting error.
func typeErr(expected, got interface{}) error {
return fmt.Errorf("abi: cannot use %v as type %v as argument", got, expected)
}

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

package abi

import (
"fmt"
"strings"

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

// Event is an event potentially triggered by the EVM's LOG mechanism. The Event
// holds type information (inputs) about the yielded output. Anonymous events
// don't get the signature canonical representation as the first LOG topic.
type Event struct {
Name      string
Anonymous bool
Inputs    []Argument
}

// Id returns the canonical representation of the event's signature used by the
// abi definition to identify event names and types.
func (e Event) Id() common.Hash {
types := make([]string, len(e.Inputs))
i := 0
for _, input := range e.Inputs {
types[i] = input.Type.String()
i++
}
return common.BytesToHash(crypto.Keccak256([]byte(fmt.Sprintf("%v(%v)", e.Name,
strings.Join(types, ","))))
}

15:F:\git\coin\ethereum\go-ethereum\accounts\abi\event_test.go
```

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

package abi

import (
"strings"
"testing"

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

func TestEventId(t *testing.T) {
var table = []struct {
definition   string
expectations map[string]common.Hash
}{
{
definition: `[
{ "type" : "event", "name" : "balance", "inputs": [{ "name" : "in", "type": "uint" }] },
{ "type" : "event", "name" : "check", "inputs": [{ "name" : "t", "type": "address" }, { "name": "b",
"type": "uint256" }] }
]`,
expectations: map[string]common.Hash{
"balance": crypto.Keccak256Hash([]byte("balance(uint256)")),
"check":   crypto.Keccak256Hash([]byte("check(address,uint256)")),
},
},
}

for _, test := range table {
abi, err := JSON(strings.NewReader(test.definition))
if err != nil {
t.Fatal(err)
}

for name, event := range abi.Events {
if event.Id() != test.expectations[name] {
t.Errorf("expected id to be %x, got %x", test.expectations[name], event.Id())
}
}
}
```

```
}

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

package abi

import (
"fmt"
"reflect"
"strings"

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

// Callable method given a `Name` and whether the method is a constant.
// If the method is `Const` no transaction needs to be created for this
// particular Method call. It can easily be simulated using a local VM.
// For example a `Balance()` method only needs to retrieve something
// from the storage and therefor requires no Tx to be send to the
// network. A method such as `Transact` does require a Tx and thus will
// be flagged `true`.
// Input specifies the required input parameters for this gives method.
type Method struct {
Name    string
Const   bool
Inputs  []Argument
Outputs []Argument
}

func (method Method) pack(args ...interface{}) ([]byte, error) {
// Make sure arguments match up and pack them
if len(args) != len(method.Inputs) {
return nil, fmt.Errorf("argument count mismatch: %d for %d", len(args), len(method.Inputs))
}
// variable input is the output appended at the end of packed
// output. This is used for strings and bytes types input.
var variableInput []byte

var ret []byte
for i, a := range args {
input := method.Inputs[i]
```

```
// pack the input
packed, err := input.Type.pack(reflect.ValueOf(a))
if err != nil {
return nil, fmt.Errorf("`%s` %v", method.Name, err)
}

// check for a slice type (string, bytes, slice)
if input.Type.requiresLengthPrefix() {
// calculate the offset
offset := len(method.Inputs)*32 + len(variableInput)
// set the offset
ret = append(ret, packNum(reflect.ValueOf(offset))...)
// Append the packed output to the variable input. The variable input
// will be appended at the end of the input.
variableInput = append(variableInput, packed...)
} else {
// append the packed value to the input
ret = append(ret, packed...)
}
}
// append the variable input at the end of the packed input
ret = append(ret, variableInput...)

return ret, nil
}

// Sig returns the methods string signature according to the ABI spec.
//
// Example
//
//     function foo(uint32 a, int b)    =    "foo(uint32,int256)"
//
// Please note that "int" is substitute for its canonical representation "int256"
func (m Method) Sig() string {
types := make([]string, len(m.Inputs))
i := 0
for _, input := range m.Inputs {
types[i] = input.Type.String()
i++
}
return fmt.Sprintf("%v(%v)", m.Name, strings.Join(types, ","))
}
```

```
func (m Method) String() string {
inputs := make([]string, len(m.Inputs))
for i, input := range m.Inputs {
inputs[i] = fmt.Sprintf("%v %v", input.Name, input.Type)
}
outputs := make([]string, len(m.Outputs))
for i, output := range m.Outputs {
if len(output.Name) > 0 {
outputs[i] = fmt.Sprintf("%v ", output.Name)
}
outputs[i] += output.Type.String()
}
constant := ""
if m.Const {
constant = "constant "
}
return fmt.Sprintf("function %v(%v) %sreturns(%v)", m.Name, strings.Join(inputs, ", "), constant,
strings.Join(outputs, ", "))
}

func (m Method) Id() []byte {
return crypto.Keccak256([]byte(m.Sig()))[:4]
}

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

package abi

import (
"math/big"
"reflect"

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

var (
big_t     = reflect.TypeOf(big.Int{})
ubig_t    = reflect.TypeOf(big.Int{})
byte_t    = reflect.TypeOf(byte(0))
```

```
byte_ts   = reflect.TypeOf([]byte(nil))
uint_t    = reflect.TypeOf(uint(0))
uint8_t   = reflect.TypeOf(uint8(0))
uint16_t  = reflect.TypeOf(uint16(0))
uint32_t  = reflect.TypeOf(uint32(0))
uint64_t  = reflect.TypeOf(uint64(0))
int_t     = reflect.TypeOf(int(0))
int8_t    = reflect.TypeOf(int8(0))
int16_t   = reflect.TypeOf(int16(0))
int32_t   = reflect.TypeOf(int32(0))
int64_t   = reflect.TypeOf(int64(0))
hash_t    = reflect.TypeOf(common.Hash{})
address_t = reflect.TypeOf(common.Address{})

uint_ts   = reflect.TypeOf([]uint(nil))
uint8_ts  = reflect.TypeOf([]uint8(nil))
uint16_ts = reflect.TypeOf([]uint16(nil))
uint32_ts = reflect.TypeOf([]uint32(nil))
uint64_ts = reflect.TypeOf([]uint64(nil))
ubig_ts   = reflect.TypeOf([]*big.Int(nil))

int_ts   = reflect.TypeOf([]int(nil))
int8_ts  = reflect.TypeOf([]int8(nil))
int16_ts = reflect.TypeOf([]int16(nil))
int32_ts = reflect.TypeOf([]int32(nil))
int64_ts = reflect.TypeOf([]int64(nil))
big_ts   = reflect.TypeOf([]*big.Int(nil))
)

// U256 converts a big Int into a 256bit EVM number.
func U256(n *big.Int) []byte {
return math.PaddedBigBytes(math.U256(n), 32)
}

// checks whether the given reflect value is signed. This also works for slices with a number type
func isSigned(v reflect.Value) bool {
switch v.Type() {
case int_ts, int8_ts, int16_ts, int32_ts, int64_ts, int_t, int8_t, int16_t, int32_t, int64_t:
return true
}
return false
}
```

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

package abi

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

func TestNumberTypes(t *testing.T) {
ubytes := make([]byte, 32)
ubytes[31] = 1

unsigned := U256(big.NewInt(1))
if !bytes.Equal(unsigned, ubytes) {
t.Errorf("expected %x got %x", ubytes, unsigned)
}
}

func TestSigned(t *testing.T) {
if isSigned(reflect.ValueOf(uint(10))) {
t.Error("signed")
}

if !isSigned(reflect.ValueOf(int(10))) {
t.Error("not signed")
}
}

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

package abi

import (
"math/big"
"reflect"

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

// packBytesSlice packs the given bytes as [L, V] as the canonical representation
// bytes slice
func packBytesSlice(bytes []byte, l int) []byte {
len := packNum(reflect.ValueOf(l))
return append(len, common.RightPadBytes(bytes, (l+31)/32*32)...)
}

// packElement packs the given reflect value according to the abi specification in
// t.
func packElement(t Type, reflectValue reflect.Value) []byte {
switch t.T {
case IntTy, UintTy:
return packNum(reflectValue)
case StringTy:
return packBytesSlice([]byte(reflectValue.String()), reflectValue.Len())
case AddressTy:
if reflectValue.Kind() == reflect.Array {
reflectValue = mustArrayToByteSlice(reflectValue)
}

return common.LeftPadBytes(reflectValue.Bytes(), 32)
case BoolTy:
if reflectValue.Bool() {
return math.PaddedBigBytes(common.Big1, 32)
} else {
return math.PaddedBigBytes(common.Big0, 32)
}
case BytesTy:
if reflectValue.Kind() == reflect.Array {
reflectValue = mustArrayToByteSlice(reflectValue)
}
return packBytesSlice(reflectValue.Bytes(), reflectValue.Len())
case FixedBytesTy, FunctionTy:
if reflectValue.Kind() == reflect.Array {
reflectValue = mustArrayToByteSlice(reflectValue)
}
return common.RightPadBytes(reflectValue.Bytes(), 32)
}
```

```
panic("abi: fatal error")
}

// packNum packs the given number (using the reflect value) and will cast it to appropriate number representation
func packNum(value reflect.Value) []byte {
switch kind := value.Kind(); kind {
case reflect.Uint, reflect.Uint8, reflect.Uint16, reflect.Uint32, reflect.Uint64:
return U256(new(big.Int).SetUint64(value.Uint()))
case reflect.Int, reflect.Int8, reflect.Int16, reflect.Int32, reflect.Int64:
return U256(big.NewInt(value.Int()))
case reflect.Ptr:
return U256(value.Interface().(*big.Int))
}
return nil
}

20:F:\git\coin\ethereum\go-ethereum\accounts\abi\pack_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package abi

import (
"bytes"
"math"
"math/big"
"reflect"
"strings"
"testing"

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

func TestPack(t *testing.T) {
for i, test := range []struct {
typ string

input  interface{}
output []byte
}{
{
"uint8",
```

```
uint8(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint8[]",
[]uint8{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint16",
uint16(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint16[]",
[]uint16{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint32",
uint32(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint32[]",
[]uint32{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint64",
uint64(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
```

```
},
{
"uint64[]",
[]uint64{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint256",
big.NewInt(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"uint256[]",
[]*big.Int{big.NewInt(1), big.NewInt(2)},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int8",
int8(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int8[]",
[]int8{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int16",
int16(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int16[]",
```

```
[]int16{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int32",
int32(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int32[]",
[]int32{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int64",
int64(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int64[]",
[]int64{1, 2},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int256",
big.NewInt(2),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"),
},
{
"int256[]",
[]*big.Int{big.NewInt(1), big.NewInt(2)},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000
```

```
00000000000000000000000000000000000000000000002"),
},
{
"bytes1",
[1]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes2",
[2]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes3",
[3]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes4",
[4]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes5",
[5]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes6",
[6]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes7",
[7]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
```

```
000"),
},
{
"bytes8",
[8]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes9",
[9]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes10",
[10]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes11",
[11]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes12",
[12]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes13",
[13]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes14",
[14]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
```

```
000"),
},
{
"bytes15",
[15]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes16",
[16]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes17",
[17]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes18",
[18]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes19",
[19]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes20",
[20]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes21",
[21]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
```

```
000"),
},
{
"bytes22",
[22]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes23",
[23]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes24",
[24]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes24",
[24]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes25",
[25]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes26",
[26]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes27",
[27]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
```

```
000"),
},
{
"bytes28",
[28]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes29",
[29]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes30",
[30]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes31",
[31]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"bytes32",
[32]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"address[]",
[]common.Address{{1}, {2}},
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000000
0020000000000000000000000000100000000000000000000000000000000000000000000000000
0000000000000200000000000000000000000000000000"),
},
{
"bytes32[]",
[]common.Hash{{1}, {2}},
```

```
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000000
0020100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000"),
},
{
"function",
[24]byte{1},
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000
000"),
},
{
"string",
"foobar",
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000000
006666f6f6261720000000000000000000000000000000000000000000000000000"),
},
} {
typ, err := NewType(test.typ)
if err != nil {
t.Fatal("unexpected parse error:", err)
}

output, err := typ.pack(reflect.ValueOf(test.input))
if err != nil {
t.Fatal("unexpected pack error:", err)
}

if !bytes.Equal(output, test.output) {
t.Errorf("%d failed. Expected bytes: '%x' Got: '%x'", i, test.output, output)
}
}
}

func TestMethodPack(t *testing.T) {
abi, err := JSON(strings.NewReader(jsondata2))
if err != nil {
t.Fatal(err)
}

sig := abi.Methods["slice"].Id()
sig = append(sig, common.LeftPadBytes([]byte{1}, 32)...)
sig = append(sig, common.LeftPadBytes([]byte{2}, 32)...)
```

```
packed, err := abi.Pack("slice", []uint32{1, 2})
if err != nil {
t.Error(err)
}

if !bytes.Equal(packed, sig) {
t.Errorf("expected %x got %x", sig, packed)
}

var addrA, addrB = common.Address{1}, common.Address{2}
sig = abi.Methods["sliceAddress"].Id()
sig = append(sig, common.LeftPadBytes([]byte{32}, 32)...)
sig = append(sig, common.LeftPadBytes([]byte{2}, 32)...)
sig = append(sig, common.LeftPadBytes(addrA[:], 32)...)
sig = append(sig, common.LeftPadBytes(addrB[:], 32)...)

packed, err = abi.Pack("sliceAddress", []common.Address{addrA, addrB})
if err != nil {
t.Fatal(err)
}
if !bytes.Equal(packed, sig) {
t.Errorf("expected %x got %x", sig, packed)
}

var addrC, addrD = common.Address{3}, common.Address{4}
sig = abi.Methods["sliceMultiAddress"].Id()
sig = append(sig, common.LeftPadBytes([]byte{64}, 32)...)
sig = append(sig, common.LeftPadBytes([]byte{160}, 32)...)
sig = append(sig, common.LeftPadBytes([]byte{2}, 32)...)
sig = append(sig, common.LeftPadBytes(addrA[:], 32)...)
sig = append(sig, common.LeftPadBytes(addrB[:], 32)...)
sig = append(sig, common.LeftPadBytes([]byte{2}, 32)...)
sig = append(sig, common.LeftPadBytes(addrC[:], 32)...)
sig = append(sig, common.LeftPadBytes(addrD[:], 32)...)

packed, err = abi.Pack("sliceMultiAddress", []common.Address{addrA, addrB},
[]common.Address{addrC, addrD})
if err != nil {
t.Fatal(err)
}
if !bytes.Equal(packed, sig) {
```

```
t.Errorf("expected %x got %x", sig, packed)
}

sig = abi.Methods["slice256"].Id()
sig = append(sig, common.LeftPadBytes([]byte{1}, 32)...)
sig = append(sig, common.LeftPadBytes([]byte{2}, 32)...)

packed, err = abi.Pack("slice256", []*big.Int{big.NewInt(1), big.NewInt(2)})
if err != nil {
t.Error(err)
}

if !bytes.Equal(packed, sig) {
t.Errorf("expected %x got %x", sig, packed)
}
}

func TestPackNumber(t *testing.T) {
tests := []struct {
value  reflect.Value
packed []byte
}{
// Protocol limits
{reflect.ValueOf(0),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000000")},
{reflect.ValueOf(1),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001")},
{reflect.ValueOf(-1), common.Hex2Bytes("ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff")},

// Type corner cases
{reflect.ValueOf(uint8(math.MaxUint8)),
common.Hex2Bytes("00000000000000000000000000000000000000000000000000000000000000ff")},
{reflect.ValueOf(uint16(math.MaxUint16)),
common.Hex2Bytes("000000000000000000000000000000000000000000000000000000000000ffff")},
{reflect.ValueOf(uint32(math.MaxUint32)),
common.Hex2Bytes("00000000000000000000000000000000000000000000000000000000ffffffff")},
{reflect.ValueOf(uint64(math.MaxUint64)),
```

```
common.Hex2Bytes("0000000000000000000000000000000000000000000000000fffffffffffffff")},

{reflect.ValueOf(int8(math.MaxInt8)),
common.Hex2Bytes("00000000000000000000000000000000000000000000000000000000000007f")},
{reflect.ValueOf(int16(math.MaxInt16)),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000007fff")},
{reflect.ValueOf(int32(math.MaxInt32)),
common.Hex2Bytes("000000000000000000000000000000000000000000000000000000007fffffff")},
{reflect.ValueOf(int64(math.MaxInt64)),
common.Hex2Bytes("00000000000000000000000000000000000000000000000007fffffffffffff")},

{reflect.ValueOf(int8(math.MinInt8)),
common.Hex2Bytes("ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80")},
{reflect.ValueOf(int16(math.MinInt16)),
common.Hex2Bytes("ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8000")},
{reflect.ValueOf(int32(math.MinInt32)),
common.Hex2Bytes("ffffffffffffffffffffffffffffffffffffffffffffffffffffffff80000000")},
{reflect.ValueOf(int64(math.MinInt64)),
common.Hex2Bytes("ffffffffffffffffffffffffffffffffffffffffffffffff8000000000000000")},
}
for i, tt := range tests {
packed := packNum(tt.value)
if !bytes.Equal(packed, tt.packed) {
t.Errorf("test %d: pack mismatch: have %x, want %x", i, packed, tt.packed)
}
}
if packed := packNum(reflect.ValueOf("string")); packed != nil {
t.Errorf("expected 'string' to pack to nil. got %x instead", packed)
}
}

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

package abi

import (
"fmt"
"reflect"
```

```
)

// indirect recursively dereferences the value until it either gets the value
// or finds a big.Int
func indirect(v reflect.Value) reflect.Value {
if v.Kind() == reflect.Ptr && v.Elem().Type() != big_t {
return indirect(v.Elem())
}
return v
}

// reflectIntKind returns the reflect using the given size and
// unsignedness.
func reflectIntKindAndType(unsigned bool, size int) (reflect.Kind, reflect.Type) {
switch size {
case 8:
if unsigned {
return reflect.Uint8, uint8_t
}
return reflect.Int8, int8_t
case 16:
if unsigned {
return reflect.Uint16, uint16_t
}
return reflect.Int16, int16_t
case 32:
if unsigned {
return reflect.Uint32, uint32_t
}
return reflect.Int32, int32_t
case 64:
if unsigned {
return reflect.Uint64, uint64_t
}
return reflect.Int64, int64_t
}
return reflect.Ptr, big_t
}

// mustArrayToBytesSlice creates a new byte slice with the exact same size as value
// and copies the bytes in value to the new slice.
func mustArrayToByteSlice(value reflect.Value) reflect.Value {
```

```
slice := reflect.MakeSlice(reflect.TypeOf([]byte{}), value.Len(), value.Len())
reflect.Copy(slice, value)
return slice
}

// set attempts to assign src to dst by either setting, copying or otherwise.
//
// set is a bit more lenient when it comes to assignment and doesn't force an as
// strict ruleset as bare `reflect` does.
func set(dst, src reflect.Value, output Argument) error {
dstType := dst.Type()
srcType := src.Type()

switch {
case dstType.AssignableTo(src.Type()):
dst.Set(src)
case dstType.Kind() == reflect.Array && srcType.Kind() == reflect.Slice:
if dst.Len() < output.Type.SliceSize {
return fmt.Errorf("abi: cannot unmarshal src (len=%d) in to dst (len=%d)", output.Type.SliceSize,
dst.Len())
}
reflect.Copy(dst, src)
case dstType.Kind() == reflect.Interface:
dst.Set(src)
case dstType.Kind() == reflect.Ptr:
return set(dst.Elem(), src, output)
default:
return fmt.Errorf("abi: cannot unmarshal %v in to %v", src.Type(), dst.Type())
}
return nil
}

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

package abi

import (
"fmt"
"reflect"
"regexp"
"strconv"
```

```
)

const (
IntTy byte = iota
UintTy
BoolTy
StringTy
SliceTy
AddressTy
FixedBytesTy
BytesTy
HashTy
FixedPointTy
FunctionTy
)

// Type is the reflection of the supported argument type
type Type struct {
IsSlice, IsArray bool
SliceSize        int

Elem *Type

Kind reflect.Kind
Type reflect.Type
Size int
T    byte // Our own type checking

stringKind string // holds the unparsed string for deriving signatures
}

var (
// fullTypeRegex parses the abi types
//
// Types can be in the format of:
//
// Input  = Type [ "[" [ Number ] "]" ] Name .
// Type   = [ "u" ] "int" [ Number ] [ x ] [ Number ].
//
// Examples:
//
//      string     int       uint       fixed
```

```
//      string32   int8      uint8      uint[]
//      address    int256    uint256    fixed128x128[2]
fullTypeRegex = regexp.MustCompile(`([a-zA-Z0-9]+)(\[([0-9]*)\])?`)
// typeRegex parses the abi sub types
typeRegex = regexp.MustCompile("([a-zA-Z]+)(([0-9]+)(x([0-9]+))?)?")
)

// NewType creates a new reflection type of abi type given in t.
func NewType(t string) (typ Type, err error) {
res := fullTypeRegex.FindAllStringSubmatch(t, -1)[0]
// check if type is slice and parse type.
switch {
case res[3] != "":
// err is ignored. Already checked for number through the regexp
typ.SliceSize, _ = strconv.Atoi(res[3])
typ.IsArray = true
case res[2] != "":
typ.IsSlice, typ.SliceSize = true, -1
case res[0] == "":
return Type{}, fmt.Errorf("abi: type parse error: %s", t)
}
if typ.IsArray || typ.IsSlice {
sliceType, err := NewType(res[1])
if err != nil {
return Type{}, err
}
typ.Elem = &sliceType
typ.stringKind = sliceType.stringKind + t[len(res[1]):]
// Although we know that this is an array, we cannot return
// as we don't know the type of the element, however, if it
// is still an array, then don't determine the type.
if typ.Elem.IsArray || typ.Elem.IsSlice {
return typ, nil
}
}

// parse the type and size of the abi-type.
parsedType := typeRegex.FindAllStringSubmatch(res[1], -1)[0]
// varSize is the size of the variable
var varSize int
if len(parsedType[3]) > 0 {
var err error
```

```
varSize, err = strconv.Atoi(parsedType[2])
if err != nil {
return Type{}, fmt.Errorf("abi: error parsing variable size: %v", err)
}
}
// varType is the parsed abi type
varType := parsedType[1]
// substitute canonical integer
if varSize == 0 && (varType == "int" || varType == "uint") {
varSize = 256
t += "256"
}

// only set stringKind if not array or slice, as for those,
// the correct string type has been set
if !(typ.IsArray || typ.IsSlice) {
typ.stringKind = t
}

switch varType {
case "int":
typ.Kind, typ.Type = reflectIntKindAndType(false, varSize)
typ.Size = varSize
typ.T = IntTy
case "uint":
typ.Kind, typ.Type = reflectIntKindAndType(true, varSize)
typ.Size = varSize
typ.T = UintTy
case "bool":
typ.Kind = reflect.Bool
typ.T = BoolTy
case "address":
typ.Kind = reflect.Array
typ.Type = address_t
typ.Size = 20
typ.T = AddressTy
case "string":
typ.Kind = reflect.String
typ.Size = -1
typ.T = StringTy
case "bytes":
sliceType, _ := NewType("uint8")
```

```
typ.Elem = &sliceType
if varSize == 0 {
typ.IsSlice = true
typ.T = BytesTy
typ.SliceSize = -1
} else {
typ.IsArray = true
typ.T = FixedBytesTy
typ.SliceSize = varSize
}
case "function":
sliceType, _ := NewType("uint8")
typ.Elem = &sliceType
typ.IsArray = true
typ.T = FunctionTy
typ.SliceSize = 24
default:
return Type{}, fmt.Errorf("unsupported arg type: %s", t)
}

return
}

// String implements Stringer
func (t Type) String() (out string) {
return t.stringKind
}

func (t Type) pack(v reflect.Value) ([]byte, error) {
// dereference pointer first if it's a pointer
v = indirect(v)

if err := typeCheck(t, v); err != nil {
return nil, err
}

if (t.IsSlice || t.IsArray) && t.T != BytesTy && t.T != FixedBytesTy && t.T != FunctionTy {
var packed []byte

for i := 0; i < v.Len(); i++ {
val, err := t.Elem.pack(v.Index(i))
if err != nil {
```

```
return nil, err
}
packed = append(packed, val...)
}
if t.IsSlice {
return packBytesSlice(packed, v.Len()), nil
} else if t.IsArray {
return packed, nil
}
}

return packElement(t, v), nil
}

// requireLengthPrefix returns whether the type requires any sort of length
// prefixing.
func (t Type) requiresLengthPrefix() bool {
return t.T != FixedBytesTy && (t.T == StringTy || t.T == BytesTy || t.IsSlice)
}

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

package abi

import (
"math/big"
"reflect"
"testing"

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

// typeWithoutStringer is a alias for the Type type which simply doesn't implement
// the stringer interface to allow printing type details in the tests below.
type typeWithoutStringer Type

// Tests that all allowed types get recognized by the type parser.
func TestTypeRegexp(t *testing.T) {
tests := []struct {
blob string
kind Type
```

}{
{"bool", Type{Kind: reflect.Bool, T: BoolTy, stringKind: "bool"}},
{"bool[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Bool, T: BoolTy, Elem: &Type{Kind: reflect.Bool, T: BoolTy, stringKind: "bool"}, stringKind: "bool[]"}},
{"bool[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Bool, T: BoolTy, Elem: &Type{Kind: reflect.Bool, T: BoolTy, stringKind: "bool"}, stringKind: "bool[2]"}},
{"int8", Type{Kind: reflect.Int8, Type: int8_t, Size: 8, T: IntTy, stringKind: "int8"}},
{"int16", Type{Kind: reflect.Int16, Type: int16_t, Size: 16, T: IntTy, stringKind: "int16"}},
{"int32", Type{Kind: reflect.Int32, Type: int32_t, Size: 32, T: IntTy, stringKind: "int32"}},
{"int64", Type{Kind: reflect.Int64, Type: int64_t, Size: 64, T: IntTy, stringKind: "int64"}},
{"int256", Type{Kind: reflect.Ptr, Type: big_t, Size: 256, T: IntTy, stringKind: "int256"}},
{"int8[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Int8, Type: int8_t, Size: 8, T: IntTy, Elem: &Type{Kind: reflect.Int8, Type: int8_t, Size: 8, T: IntTy, stringKind: "int8"}, stringKind: "int8[]"}},
{"int8[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Int8, Type: int8_t, Size: 8, T: IntTy, Elem: &Type{Kind: reflect.Int8, Type: int8_t, Size: 8, T: IntTy, stringKind: "int8"}, stringKind: "int8[2]"}},
{"int16[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Int16, Type: int16_t, Size: 16, T: IntTy, Elem: &Type{Kind: reflect.Int16, Type: int16_t, Size: 16, T: IntTy, stringKind: "int16"}, stringKind: "int16[]"}},
{"int16[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Int16, Type: int16_t, Size: 16, T: IntTy, Elem: &Type{Kind: reflect.Int16, Type: int16_t, Size: 16, T: IntTy, stringKind: "int16"}, stringKind: "int16[2]"}},
{"int32[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Int32, Type: int32_t, Size: 32, T: IntTy, Elem: &Type{Kind: reflect.Int32, Type: int32_t, Size: 32, T: IntTy, stringKind: "int32"}, stringKind: "int32[]"}},
{"int32[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Int32, Type: int32_t, Size: 32, T: IntTy, Elem: &Type{Kind: reflect.Int32, Type: int32_t, Size: 32, T: IntTy, stringKind: "int32"}, stringKind: "int32[2]"}},
{"int64[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Int64, Type: int64_t, Size: 64, T: IntTy, Elem: &Type{Kind: reflect.Int64, Type: int64_t, Size: 64, T: IntTy, stringKind: "int64"}, stringKind: "int64[]"}},
{"int64[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Int64, Type: int64_t, Size: 64, T: IntTy, Elem: &Type{Kind: reflect.Int64, Type: int64_t, Size: 64, T: IntTy, stringKind: "int64"}, stringKind: "int64[2]"}},
{"int256[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Ptr, Type: big_t, Size: 256, T: IntTy, Elem: &Type{Kind: reflect.Ptr, Type: big_t, Size: 256, T: IntTy, stringKind: "int256"}, stringKind: "int256[]"}},
{"int256[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Ptr, Type: big_t, Size: 256, T: IntTy, Elem: &Type{Kind: reflect.Ptr, Type: big_t, Size: 256, T: IntTy, stringKind: "int256"}, stringKind: "int256[2]"}},
{"uint8", Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}},
{"uint16", Type{Kind: reflect.Uint16, Type: uint16_t, Size: 16, T: UintTy, stringKind: "uint16"}},
{"uint32", Type{Kind: reflect.Uint32, Type: uint32_t, Size: 32, T: UintTy, stringKind: "uint32"}},

{"uint64", Type{Kind: reflect.Uint64, Type: uint64_t, Size: 64, T: UintTy, stringKind: "uint64"}},
{"uint256", Type{Kind: reflect.Ptr, Type: big_t, Size: 256, T: UintTy, stringKind: "uint256"}},
{"uint8[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, Elem: &Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}, stringKind: "uint8[]"}},
{"uint8[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, Elem: &Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}, stringKind: "uint8[2]"}},
{"uint16[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Uint16, Type: uint16_t, Size: 16, T: UintTy, Elem: &Type{Kind: reflect.Uint16, Type: uint16_t, Size: 16, T: UintTy, stringKind: "uint16"}, stringKind: "uint16[]"}},
{"uint16[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Uint16, Type: uint16_t, Size: 16, T: UintTy, Elem: &Type{Kind: reflect.Uint16, Type: uint16_t, Size: 16, T: UintTy, stringKind: "uint16"}, stringKind: "uint16[2]"}},
{"uint32[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Uint32, Type: uint32_t, Size: 32, T: UintTy, Elem: &Type{Kind: reflect.Uint32, Type: uint32_t, Size: 32, T: UintTy, stringKind: "uint32"}, stringKind: "uint32[]"}},
{"uint32[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Uint32, Type: uint32_t, Size: 32, T: UintTy, Elem: &Type{Kind: reflect.Uint32, Type: uint32_t, Size: 32, T: UintTy, stringKind: "uint32"}, stringKind: "uint32[2]"}},
{"uint64[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Uint64, Type: uint64_t, Size: 64, T: UintTy, Elem: &Type{Kind: reflect.Uint64, Type: uint64_t, Size: 64, T: UintTy, stringKind: "uint64"}, stringKind: "uint64[]"}},
{"uint64[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Uint64, Type: uint64_t, Size: 64, T: UintTy, Elem: &Type{Kind: reflect.Uint64, Type: uint64_t, Size: 64, T: UintTy, stringKind: "uint64"}, stringKind: "uint64[2]"}},
{"uint256[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Ptr, Type: big_t, Size: 256, T: UintTy, Elem: &Type{Kind: reflect.Ptr, Type: big_t, Size: 256, T: UintTy, stringKind: "uint256"}, stringKind: "uint256[]"}},
{"uint256[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Ptr, Type: big_t, Size: 256, T: UintTy, Elem: &Type{Kind: reflect.Ptr, Type: big_t, Size: 256, T: UintTy, stringKind: "uint256"}, stringKind: "uint256[2]"}},
{"bytes32", Type{IsArray: true, SliceSize: 32, Elem: &Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}, T: FixedBytesTy, stringKind: "bytes32"}},
{"bytes[]", Type{IsSlice: true, SliceSize: -1, Elem: &Type{IsSlice: true, SliceSize: -1, Elem: &Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}, T: BytesTy, stringKind: "bytes"}, stringKind: "bytes[]"}},
{"bytes[2]", Type{IsArray: true, SliceSize: 2, Elem: &Type{IsSlice: true, SliceSize: -1, Elem: &Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}, T: BytesTy, stringKind: "bytes"}, stringKind: "bytes[2]"}},
{"bytes32[]", Type{IsSlice: true, SliceSize: -1, Elem: &Type{IsArray: true, SliceSize: 32, Elem: &Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}, T: FixedBytesTy,

```
stringKind: "bytes32"}, stringKind: "bytes32[]"}},
{"bytes32[2]", Type{IsArray: true, SliceSize: 2, Elem: &Type{IsArray: true, SliceSize: 32, Elem:
&Type{Kind: reflect.Uint8, Type: uint8_t, Size: 8, T: UintTy, stringKind: "uint8"}, T: FixedBytesTy,
stringKind: "bytes32"}, stringKind: "bytes32[2]"}},
{"string", Type{Kind: reflect.String, Size: -1, T: StringTy, stringKind: "string"}},
{"string[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.String, T: StringTy, Size: -1, Elem:
&Type{Kind: reflect.String, T: StringTy, Size: -1, stringKind: "string"}, stringKind: "string[]"}},
{"string[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.String, T: StringTy, Size: -1, Elem:
&Type{Kind: reflect.String, T: StringTy, Size: -1, stringKind: "string"}, stringKind: "string[2]"}},
{"address", Type{Kind: reflect.Array, Type: address_t, Size: 20, T: AddressTy, stringKind:
"address"}},
{"address[]", Type{IsSlice: true, SliceSize: -1, Kind: reflect.Array, Type: address_t, T: AddressTy,
Size: 20, Elem: &Type{Kind: reflect.Array, Type: address_t, Size: 20, T: AddressTy, stringKind:
"address"}, stringKind: "address[]"}},
{"address[2]", Type{IsArray: true, SliceSize: 2, Kind: reflect.Array, Type: address_t, T: AddressTy,
Size: 20, Elem: &Type{Kind: reflect.Array, Type: address_t, Size: 20, T: AddressTy, stringKind:
"address"}, stringKind: "address[2]"}},

// TODO when fixed types are implemented properly
// {"fixed", Type{}},
// {"fixed128x128", Type{}},
// {"fixed[]", Type{}},
// {"fixed[2]", Type{}},
// {"fixed128x128[]", Type{}},
// {"fixed128x128[2]", Type{}},
}
for i, tt := range tests {
typ, err := NewType(tt.blob)
if err != nil {
t.Errorf("type %d: failed to parse type string: %v", i, err)
}
if !reflect.DeepEqual(typ, tt.kind) {
t.Errorf("type %d: parsed type mismatch:\n  have %+v\n  want %+v", i, typeWithoutStringer(typ),
typeWithoutStringer(tt.kind))
}
}
}

func TestTypeCheck(t *testing.T) {
for i, test := range []struct {
typ   string
input interface{}
```

```
err   string
}{
{"uint", big.NewInt(1), ""},
{"int", big.NewInt(1), ""},
{"uint30", big.NewInt(1), ""},
{"uint30", uint8(1), "abi: cannot use uint8 as type ptr as argument"},
{"uint16", uint16(1), ""},
{"uint16", uint8(1), "abi: cannot use uint8 as type uint16 as argument"},
{"uint16[]", []uint16{1, 2, 3}, ""},
{"uint16[]", [3]uint16{1, 2, 3}, ""},
{"uint16[]", []uint32{1, 2, 3}, "abi: cannot use []uint32 as type []uint16 as argument"},
{"uint16[3]", [3]uint32{1, 2, 3}, "abi: cannot use [3]uint32 as type [3]uint16 as argument"},
{"uint16[3]", [4]uint16{1, 2, 3}, "abi: cannot use [4]uint16 as type [3]uint16 as argument"},
{"uint16[3]", []uint16{1, 2, 3}, ""},
{"uint16[3]", []uint16{1, 2, 3, 4}, "abi: cannot use [4]uint16 as type [3]uint16 as argument"},
{"address[]", []common.Address{{1}}, ""},
{"address[1]", []common.Address{{1}}, ""},
{"address[1]", [1]common.Address{{1}}, ""},
{"address[2]", [1]common.Address{{1}}, "abi: cannot use [1]array as type [2]array as argument"},
{"bytes32", [32]byte{}, ""},
{"bytes32", [33]byte{}, "abi: cannot use [33]uint8 as type [32]uint8 as argument"},
{"bytes32", common.Hash{1}, ""},
{"bytes31", [31]byte{}, ""},
{"bytes31", [32]byte{}, "abi: cannot use [32]uint8 as type [31]uint8 as argument"},
{"bytes", []byte{0, 1}, ""},
{"bytes", [2]byte{0, 1}, ""},
{"bytes", common.Hash{1}, ""},
{"string", "hello world", ""},
{"bytes32[]", [][32]byte{{}}, ""},
{"function", [24]byte{}, ""},
} {
typ, err := NewType(test.typ)
if err != nil {
t.Fatal("unexpected parse error:", err)
}

err = typeCheck(typ, reflect.ValueOf(test.input))
if err != nil && len(test.err) == 0 {
t.Errorf("%d failed. Expected no err but got: %v", i, err)
continue
}
if err == nil && len(test.err) != 0 {
```

```
t.Errorf("%d failed. Expected err: %v but got none", i, test.err)
continue
}

if err != nil && len(test.err) != 0 && err.Error() != test.err {
t.Errorf("%d failed. Expected err: '%v' got err: '%v'", i, test.err, err)
}
}
}
```

24:F:\git\coin\ethereum\go-ethereum\accounts\abi\unpack.go

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

package abi

import (
"encoding/binary"
"fmt"
"math/big"
"reflect"

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

// toGoSliceType parses the input and casts it to the proper slice defined by the ABI
// argument in T.
func toGoSlice(i int, t Argument, output []byte) (interface{}, error) {
index := i * 32
// The slice must, at very least be large enough for the index+32 which is exactly the size required
// for the [offset in output, size of offset].
if index+32 > len(output) {
return nil, fmt.Errorf("abi: cannot marshal in to go slice: insufficient size output %d require %d",
len(output), index+32)
}
elem := t.Type.Elem

// first we need to create a slice of the type
var refSlice reflect.Value
switch elem.T {
case IntTy, UintTy, BoolTy:
// create a new reference slice matching the element type
switch t.Type.Kind {
```

```
case reflect.Bool:
refSlice = reflect.ValueOf([]bool(nil))
case reflect.Uint8:
refSlice = reflect.ValueOf([]uint8(nil))
case reflect.Uint16:
refSlice = reflect.ValueOf([]uint16(nil))
case reflect.Uint32:
refSlice = reflect.ValueOf([]uint32(nil))
case reflect.Uint64:
refSlice = reflect.ValueOf([]uint64(nil))
case reflect.Int8:
refSlice = reflect.ValueOf([]int8(nil))
case reflect.Int16:
refSlice = reflect.ValueOf([]int16(nil))
case reflect.Int32:
refSlice = reflect.ValueOf([]int32(nil))
case reflect.Int64:
refSlice = reflect.ValueOf([]int64(nil))
default:
refSlice = reflect.ValueOf([]*big.Int(nil))
}
case AddressTy: // address must be of slice Address
refSlice = reflect.ValueOf([]common.Address(nil))
case HashTy: // hash must be of slice hash
refSlice = reflect.ValueOf([]common.Hash(nil))
case FixedBytesTy:
refSlice = reflect.ValueOf([][]byte(nil))
default: // no other types are supported
return nil, fmt.Errorf("abi: unsupported slice type %v", elem.T)
}

var slice []byte
var size int
var offset int
if t.Type.IsSlice {
// get the offset which determines the start of this array ...
offset = int(binary.BigEndian.Uint64(output[index+24 : index+32]))
if offset+32 > len(output) {
return nil, fmt.Errorf("abi: cannot marshal in to go slice: offset %d would go over slice boundary
(len=%d)", len(output), offset+32)
}
```

```
slice = output[offset:]
// ... starting with the size of the array in elements ...
size = int(binary.BigEndian.Uint64(slice[24:32]))
slice = slice[32:]
// ... and make sure that we've at the very least the amount of bytes
// available in the buffer.
if size*32 > len(slice) {
return nil, fmt.Errorf("abi: cannot marshal in to go slice: insufficient size output %d require %d",
len(output), offset+32+size*32)
}

// reslice to match the required size
slice = slice[:size*32]
} else if t.Type.IsArray {
//get the number of elements in the array
size = t.Type.SliceSize

//check to make sure array size matches up
if index+32*size > len(output) {
return nil, fmt.Errorf("abi: cannot marshal in to go array: offset %d would go over slice boundary
(len=%d)", len(output), index+32*size)
}
//slice is there for a fixed amount of times
slice = output[index : index+size*32]
}

for i := 0; i < size; i++ {
var (
inter       interface{}             // interface type
returnOutput = slice[i*32 : i*32+32] // the return output
err         error
)
// set inter to the correct type (cast)
switch elem.T {
case IntTy, UintTy:
inter = readInteger(t.Type.Kind, returnOutput)
case BoolTy:
inter, err = readBool(returnOutput)
if err != nil {
return nil, err
}
case AddressTy:
```

```
inter = common.BytesToAddress(returnOutput)
case HashTy:
inter = common.BytesToHash(returnOutput)
case FixedBytesTy:
inter = returnOutput
}
// append the item to our reflect slice
refSlice = reflect.Append(refSlice, reflect.ValueOf(inter))
}

// return the interface
return refSlice.Interface(), nil
}

func readInteger(kind reflect.Kind, b []byte) interface{} {
switch kind {
case reflect.Uint8:
return uint8(b[len(b)-1])
case reflect.Uint16:
return binary.BigEndian.Uint16(b[len(b)-2:])
case reflect.Uint32:
return binary.BigEndian.Uint32(b[len(b)-4:])
case reflect.Uint64:
return binary.BigEndian.Uint64(b[len(b)-8:])
case reflect.Int8:
return int8(b[len(b)-1])
case reflect.Int16:
return int16(binary.BigEndian.Uint16(b[len(b)-2:]))
case reflect.Int32:
return int32(binary.BigEndian.Uint32(b[len(b)-4:]))
case reflect.Int64:
return int64(binary.BigEndian.Uint64(b[len(b)-8:]))
default:
return new(big.Int).SetBytes(b)
}
}

func readBool(word []byte) (bool, error) {
if len(word) != 32 {
return false, fmt.Errorf("abi: fatal error: incorrect word length")
}
```

```
for i, b := range word {
if b != 0 && i != 31 {
return false, errBadBool
}
}
switch word[31] {
case 0:
return false, nil
case 1:
return true, nil
default:
return false, errBadBool
}

}

// toGoType parses the input and casts it to the proper type defined by the ABI
// argument in T.
func toGoType(i int, t Argument, output []byte) (interface{}, error) {
// we need to treat slices differently
if (t.Type.IsSlice || t.Type.IsArray) && t.Type.T != BytesTy && t.Type.T != StringTy && t.Type.T !=
FixedBytesTy && t.Type.T != FunctionTy {
return toGoSlice(i, t, output)
}

index := i * 32
if index+32 > len(output) {
return nil, fmt.Errorf("abi: cannot marshal in to go type: length insufficient %d require %d",
len(output), index+32)
}

// Parse the given index output and check whether we need to read
// a different offset and length based on the type (i.e. string, bytes)
var returnOutput []byte
switch t.Type.T {
case StringTy, BytesTy: // variable arrays are written at the end of the return bytes
// parse offset from which we should start reading
offset := int(binary.BigEndian.Uint64(output[index+24 : index+32]))
if offset+32 > len(output) {
return nil, fmt.Errorf("abi: cannot marshal in to go type: length insufficient %d require %d",
len(output), offset+32)
}
```

```
// parse the size up until we should be reading
size := int(binary.BigEndian.Uint64(output[offset+24 : offset+32]))
if offset+32+size > len(output) {
return nil, fmt.Errorf("abi: cannot marshal in to go type: length insufficient %d require %d",
len(output), offset+32+size)
}

// get the bytes for this return value
returnOutput = output[offset+32 : offset+32+size]
default:
returnOutput = output[index : index+32]
}

// convert the bytes to whatever is specified by the ABI.
switch t.Type.T {
case IntTy, UintTy:
return readInteger(t.Type.Kind, returnOutput), nil
case BoolTy:
return readBool(returnOutput)
case AddressTy:
return common.BytesToAddress(returnOutput), nil
case HashTy:
return common.BytesToHash(returnOutput), nil
case BytesTy, FixedBytesTy, FunctionTy:
return returnOutput, nil
case StringTy:
return string(returnOutput), nil
}
return nil, fmt.Errorf("abi: unknown type %v", t.Type.T)
}

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

package abi

import (
"bytes"
"fmt"
"math/big"
"reflect"
"strings"
```

```
"testing"

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

func TestSimpleMethodUnpack(t *testing.T) {
for i, test := range []struct {
def              string      // definition of the **output** ABI params
marshalledOutput []byte      // evm return data
expectedOut      interface{} // the expected output
outVar           string      // the output variable (e.g. uint32, *big.Int, etc)
err              string      // empty or error if expected
}{
{
`[ { "type": "bool" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
bool(true),
"bool",
"",
},
{
`[ { "type": "uint32" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
uint32(1),
"uint32",
"",
},
{
`[ { "type": "uint32" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
nil,
"uint16",
"abi: cannot unmarshal uint32 in to uint16",
},
{
`[ { "type": "uint17" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
nil,
```

```
"uint16",
"abi: cannot unmarshal *big.Int in to uint16",
},
{
`[ { "type": "uint17" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
big.NewInt(1),
"*big.Int",
"",
},

{
`[ { "type": "int32" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
int32(1),
"int32",
"",
},
{
`[ { "type": "int32" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
nil,
"int16",
"abi: cannot unmarshal int32 in to int16",
},
{
`[ { "type": "int17" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
nil,
"int16",
"abi: cannot unmarshal *big.Int in to int16",
},
{
`[ { "type": "int17" } ]`,
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
big.NewInt(1),
"*big.Int",
```

```
"",
},

{
`[ { "type": "address" } ]`,
common.Hex2Bytes("0000000000000000000000001000000000000000000000000000000000000000"),
common.Address{1},
"address",
"",
},
{
`[ { "type": "bytes32" } ]`,
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000"),
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000"),
"bytes",
"",
},
{
`[ { "type": "bytes32" } ]`,
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000"),
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000"),
"hash",
"",
},
{
`[ { "type": "bytes32" } ]`,
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000"),
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000"),
"interface",
"",
},
{
`[ { "type": "function" } ]`,
common.Hex2Bytes("0100000000000000000000000000000000000000000000000000000000000000"),
```

```
[24]byte{1},
"function",
"",
},
} {
abiDefinition := fmt.Sprintf(`[{ "name" : "method", "outputs": %s}]`, test.def)
abi, err := JSON(strings.NewReader(abiDefinition))
if err != nil {
t.Errorf("%d failed. %v", i, err)
continue
}

var outvar interface{}
switch test.outVar {
case "bool":
var v bool
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "uint8":
var v uint8
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "uint16":
var v uint16
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "uint32":
var v uint32
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "uint64":
var v uint64
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "int8":
var v int8
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "int16":
var v int16
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
```

```
case "int32":
var v int32
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "int64":
var v int64
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "*big.Int":
var v *big.Int
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "address":
var v common.Address
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "bytes":
var v []byte
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "hash":
var v common.Hash
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v.Bytes()[:]
case "function":
var v [24]byte
err = abi.Unpack(&v, "method", test.marshalledOutput)
outvar = v
case "interface":
err = abi.Unpack(&outvar, "method", test.marshalledOutput)
default:
t.Errorf("unsupported type '%v' please add it to the switch statement in this test", test.outVar)
continue
}

if err != nil && len(test.err) == 0 {
t.Errorf("%d failed. Expected no err but got: %v", i, err)
continue
}
if err == nil && len(test.err) != 0 {
t.Errorf("%d failed. Expected err: %v but got none", i, test.err)
continue
```

```
}
if err != nil && len(test.err) != 0 && err.Error() != test.err {
t.Errorf("%d failed. Expected err: '%v' got err: '%v'", i, test.err, err)
continue
}

if err == nil {
if !reflect.DeepEqual(test.expectedOut, outvar) {
t.Errorf("%d failed. Output error: expected %v, got %v", i, test.expectedOut, outvar)
}
}
}
}

func TestUnpackSetInterfaceSlice(t *testing.T) {
var (
var1 = new(uint8)
var2 = new(uint8)
)
out := []interface{}{var1, var2}
abi, err := JSON(strings.NewReader(`[{"type":"function", "name":"ints", "outputs":[{"type":"uint8"},
{"type":"uint8"}]}]`))
if err != nil {
t.Fatal(err)
}
marshalledReturn :=
append(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002")...)
err = abi.Unpack(&out, "ints", marshalledReturn)
if err != nil {
t.Fatal(err)
}
if *var1 != 1 {
t.Error("expected var1 to be 1, got", *var1)
}
if *var2 != 2 {
t.Error("expected var2 to be 2, got", *var2)
}

out = []interface{}{var1}
```

```
err = abi.Unpack(&out, "ints", marshalledReturn)

expErr := "abi: cannot marshal in to slices of unequal size (require: 2, got: 1)"
if err == nil || err.Error() != expErr {
t.Error("expected err:", expErr, "Got:", err)
}
}

func TestUnpackSetInterfaceArrayOutput(t *testing.T) {
var (
var1 = new([1]uint32)
var2 = new([1]uint32)
)
out := []interface{}{var1, var2}
abi, err := JSON(strings.NewReader(`[{"type":"function", "name":"ints",
"outputs":[{"type":"uint32[1]"}, {"type":"uint32[1]"}]}]`))
if err != nil {
t.Fatal(err)
}
marshalledReturn :=
append(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"),
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002")...)
err = abi.Unpack(&out, "ints", marshalledReturn)
if err != nil {
t.Fatal(err)
}

if *var1 != [1]uint32{1} {
t.Error("expected var1 to be [1], got", *var1)
}
if *var2 != [1]uint32{2} {
t.Error("expected var2 to be [2], got", *var2)
}
}

func TestMultiReturnWithStruct(t *testing.T) {
const definition = `[
{ "name" : "multi", "constant" : false, "outputs": [ { "name": "Int", "type": "uint256" }, { "name":
"String", "type": "string" } ] }]`
```

```
abi, err := JSON(strings.NewReader(definition))
if err != nil {
t.Fatal(err)
}

// using buff to make the code readable
buff := new(bytes.Buffer)
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000040"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000005"))
stringOut := "hello"
buff.Write(common.RightPadBytes([]byte(stringOut), 32))

var inter struct {
Int    *big.Int
String string
}
err = abi.Unpack(&inter, "multi", buff.Bytes())
if err != nil {
t.Error(err)
}

if inter.Int == nil || inter.Int.Cmp(big.NewInt(1)) != 0 {
t.Error("expected Int to be 1 got", inter.Int)
}

if inter.String != stringOut {
t.Error("expected String to be", stringOut, "got", inter.String)
}

var reversed struct {
String string
Int    *big.Int
}

err = abi.Unpack(&reversed, "multi", buff.Bytes())
if err != nil {
t.Error(err)
}
```

```
if reversed.Int == nil || reversed.Int.Cmp(big.NewInt(1)) != 0 {
t.Error("expected Int to be 1 got", reversed.Int)
}

if reversed.String != stringOut {
t.Error("expected String to be", stringOut, "got", reversed.String)
}
}

func TestMultiReturnWithSlice(t *testing.T) {
const definition = `[
{ "name" : "multi", "constant" : false, "outputs": [ { "name": "Int", "type": "uint256" }, { "name":
"String", "type": "string" } ] }]`

abi, err := JSON(strings.NewReader(definition))
if err != nil {
t.Fatal(err)
}

// using buff to make the code readable
buff := new(bytes.Buffer)
buff.Write(common.Hex2Bytes("00000000000000000000000000000000000000000000000000000000
000000000001"))
buff.Write(common.Hex2Bytes("00000000000000000000000000000000000000000000000000000000
000000000040"))
buff.Write(common.Hex2Bytes("00000000000000000000000000000000000000000000000000000000
000000000005"))
stringOut := "hello"
buff.Write(common.RightPadBytes([]byte(stringOut), 32))

var inter []interface{}
err = abi.Unpack(&inter, "multi", buff.Bytes())
if err != nil {
t.Error(err)
}

if len(inter) != 2 {
t.Fatal("expected 2 results got", len(inter))
}

if num, ok := inter[0].(*big.Int); !ok || num.Cmp(big.NewInt(1)) != 0 {
```

```
t.Error("expected index 0 to be 1 got", num)
}

if str, ok := inter[1].(string); !ok || str != stringOut {
t.Error("expected index 1 to be", stringOut, "got", str)
}
}

func TestMarshalArrays(t *testing.T) {
const definition = `[
{ "name" : "bytes32", "constant" : false, "outputs": [ { "type": "bytes32" } ] },
{ "name" : "bytes10", "constant" : false, "outputs": [ { "type": "bytes10" } ] }
]`

abi, err := JSON(strings.NewReader(definition))
if err != nil {
t.Fatal(err)
}

output := common.LeftPadBytes([]byte{1}, 32)

var bytes10 [10]byte
err = abi.Unpack(&bytes10, "bytes32", output)
if err == nil || err.Error() != "abi: cannot unmarshal src (len=32) in to dst (len=10)" {
t.Error("expected error or bytes32 not be assignable to bytes10:", err)
}

var bytes32 [32]byte
err = abi.Unpack(&bytes32, "bytes32", output)
if err != nil {
t.Error("didn't expect error:", err)
}
if !bytes.Equal(bytes32[:], output) {
t.Error("expected bytes32[31] to be 1 got", bytes32[31])
}

type (
B10 [10]byte
B32 [32]byte
)

var b10 B10
```

```
err = abi.Unpack(&b10, "bytes32", output)
if err == nil || err.Error() != "abi: cannot unmarshal src (len=32) in to dst (len=10)" {
t.Error("expected error or bytes32 not be assignable to bytes10:", err)
}

var b32 B32
err = abi.Unpack(&b32, "bytes32", output)
if err != nil {
t.Error("didn't expect error:", err)
}
if !bytes.Equal(b32[:], output) {
t.Error("expected bytes32[31] to be 1 got", bytes32[31])
}

output[10] = 1
var shortAssignLong [32]byte
err = abi.Unpack(&shortAssignLong, "bytes10", output)
if err != nil {
t.Error("didn't expect error:", err)
}
if !bytes.Equal(output, shortAssignLong[:]) {
t.Errorf("expected %x to be %x", shortAssignLong, output)
}
}

func TestUnmarshal(t *testing.T) {
const definition = `[
{ "name" : "int", "constant" : false, "outputs": [ { "type": "uint256" } ] },
{ "name" : "bool", "constant" : false, "outputs": [ { "type": "bool" } ] },
{ "name" : "bytes", "constant" : false, "outputs": [ { "type": "bytes" } ] },
{ "name" : "fixed", "constant" : false, "outputs": [ { "type": "bytes32" } ] },
{ "name" : "multi", "constant" : false, "outputs": [ { "type": "bytes" }, { "type": "bytes" } ] },
{ "name" : "intArraySingle", "constant" : false, "outputs": [ { "type": "uint256[3]" } ] },
{ "name" : "addressSliceSingle", "constant" : false, "outputs": [ { "type": "address[]" } ] },
{ "name" : "addressSliceDouble", "constant" : false, "outputs": [ { "name": "a", "type": "address[]" },
{ "name": "b", "type": "address[]" } ] },
{ "name" : "mixedBytes", "constant" : true, "outputs": [ { "name": "a", "type": "bytes" }, { "name": "b",
"type": "bytes32" } ] }]`

abi, err := JSON(strings.NewReader(definition))
if err != nil {
t.Fatal(err)
```

```
}
buff := new(bytes.Buffer)

// marshal int
var Int *big.Int
err = abi.Unpack(&Int, "int",
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"))
if err != nil {
t.Error(err)
}

if Int == nil || Int.Cmp(big.NewInt(1)) != 0 {
t.Error("expected Int to be 1 got", Int)
}

// marshal bool
var Bool bool
err = abi.Unpack(&Bool, "bool",
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"))
if err != nil {
t.Error(err)
}

if !Bool {
t.Error("expected Bool to be true")
}

// marshal dynamic bytes max length 32
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020"))
bytesOut := common.RightPadBytes([]byte("hello"), 32)
buff.Write(bytesOut)

var Bytes []byte
err = abi.Unpack(&Bytes, "bytes", buff.Bytes())
if err != nil {
t.Error(err)
}
```

```
if !bytes.Equal(Bytes, bytesOut) {
t.Errorf("expected %x got %x", bytesOut, Bytes)
}

// marshall dynamic bytes max length 64
buff.Reset()
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000040"))
bytesOut = common.RightPadBytes([]byte("hello"), 64)
buff.Write(bytesOut)

err = abi.Unpack(&Bytes, "bytes", buff.Bytes())
if err != nil {
t.Error(err)
}

if !bytes.Equal(Bytes, bytesOut) {
t.Errorf("expected %x got %x", bytesOut, Bytes)
}

// marshall dynamic bytes max length 63
buff.Reset()
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020"))
buff.Write(common.Hex2Bytes("000000000000000000000000000000000000000000000000000000000000003f"))
bytesOut = common.RightPadBytes([]byte("hello"), 63)
buff.Write(bytesOut)

err = abi.Unpack(&Bytes, "bytes", buff.Bytes())
if err != nil {
t.Error(err)
}

if !bytes.Equal(Bytes, bytesOut) {
t.Errorf("expected %x got %x", bytesOut, Bytes)
}

// marshal dynamic bytes output empty
```

```
err = abi.Unpack(&Bytes, "bytes", nil)
if err == nil {
t.Error("expected error")
}

// marshal dynamic bytes length 5
buff.Reset()
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000005"))
buff.Write(common.RightPadBytes([]byte("hello"), 32))

err = abi.Unpack(&Bytes, "bytes", buff.Bytes())
if err != nil {
t.Error(err)
}

if !bytes.Equal(Bytes, []byte("hello")) {
t.Errorf("expected %x got %x", bytesOut, Bytes)
}

// marshal dynamic bytes length 5
buff.Reset()
buff.Write(common.RightPadBytes([]byte("hello"), 32))

var hash common.Hash
err = abi.Unpack(&hash, "fixed", buff.Bytes())
if err != nil {
t.Error(err)
}

helloHash := common.BytesToHash(common.RightPadBytes([]byte("hello"), 32))
if hash != helloHash {
t.Errorf("Expected %x to equal %x", hash, helloHash)
}

// marshal error
buff.Reset()
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020"))
err = abi.Unpack(&Bytes, "bytes", buff.Bytes())
```

```
if err == nil {
t.Error("expected error")
}

err = abi.Unpack(&Bytes, "multi", make([]byte, 64))
if err == nil {
t.Error("expected error")
}

// marshal mixed bytes
buff.Reset()
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000040"))
fixed :=
common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001")
buff.Write(fixed)
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000020"))
bytesOut = common.RightPadBytes([]byte("hello"), 32)
buff.Write(bytesOut)

var out []interface{}
err = abi.Unpack(&out, "mixedBytes", buff.Bytes())
if err != nil {
t.Fatal("didn't expect error:", err)
}

if !bytes.Equal(bytesOut, out[0].([]byte)) {
t.Errorf("expected %x, got %x", bytesOut, out[0])
}

if !bytes.Equal(fixed, out[1].([]byte)) {
t.Errorf("expected %x, got %x", fixed, out[1])
}

buff.Reset()
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000
```

```
000000000003"))
// marshal int array
var intArray [3]*big.Int
err = abi.Unpack(&intArray, "intArraySingle", buff.Bytes())
if err != nil {
t.Error(err)
}
var testAgainstIntArray [3]*big.Int
testAgainstIntArray[0] = big.NewInt(1)
testAgainstIntArray[1] = big.NewInt(2)
testAgainstIntArray[2] = big.NewInt(3)

for i, Int := range intArray {
if Int.Cmp(testAgainstIntArray[i]) != 0 {
t.Errorf("expected %v, got %v", testAgainstIntArray[i], Int)
}
}
// marshal address slice
buff.Reset()
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000
000000000020")) // offset
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000
000000000001")) // size
buff.Write(common.Hex2Bytes("0000000000000000000000010000000000000000000000000000
000000000000"))

var outAddr []common.Address
err = abi.Unpack(&outAddr, "addressSliceSingle", buff.Bytes())
if err != nil {
t.Fatal("didn't expect error:", err)
}

if len(outAddr) != 1 {
t.Fatal("expected 1 item, got", len(outAddr))
}

if outAddr[0] != (common.Address{1}) {
t.Errorf("expected %x, got %x", common.Address{1}, outAddr[0])
}

// marshal multiple address slice
buff.Reset()
```

```
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000040")) // offset
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000080")) // offset
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000001")) // size
buff.Write(common.Hex2Bytes("0000000000000000000000010000000000000000000000000000000000000000"))
buff.Write(common.Hex2Bytes("0000000000000000000000000000000000000000000000000000000000000002")) // size
buff.Write(common.Hex2Bytes("0000000000000000000000020000000000000000000000000000000000000000"))
buff.Write(common.Hex2Bytes("0000000000000000000000030000000000000000000000000000000000000000"))

var outAddrStruct struct {
A []common.Address
B []common.Address
}
err = abi.Unpack(&outAddrStruct, "addressSliceDouble", buff.Bytes())
if err != nil {
t.Fatal("didn't expect error:", err)
}

if len(outAddrStruct.A) != 1 {
t.Fatal("expected 1 item, got", len(outAddrStruct.A))
}

if outAddrStruct.A[0] != (common.Address{1}) {
t.Errorf("expected %x, got %x", common.Address{1}, outAddrStruct.A[0])
}

if len(outAddrStruct.B) != 2 {
t.Fatal("expected 1 item, got", len(outAddrStruct.B))
}

if outAddrStruct.B[0] != (common.Address{2}) {
t.Errorf("expected %x, got %x", common.Address{2}, outAddrStruct.B[0])
}
if outAddrStruct.B[1] != (common.Address{3}) {
t.Errorf("expected %x, got %x", common.Address{3}, outAddrStruct.B[1])
}
```

```
// marshal invalid address slice
buff.Reset()
buff.Write(common.Hex2Bytes("00000000000000000000000000000000000000000000000000000000000000000000000000000100"))

err = abi.Unpack(&outAddr, "addressSliceSingle", buff.Bytes())
if err == nil {
t.Fatal("expected error:", err)
}
}

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

// Package accounts implements high level Ethereum account management.
package accounts

import (
"math/big"

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

// Account represents an Ethereum account located at a specific location defined
// by the optional URL field.
type Account struct {
Address common.Address `json:"address"` // Ethereum account address derived from the key
URL     URL           `json:"url"`     // Optional resource locator within a backend
}

// Wallet represents a software or hardware wallet that might contain one or more
// accounts (derived from the same seed).
type Wallet interface {
// URL retrieves the canonical path under which this wallet is reachable. It is
// user by upper layers to define a sorting order over all wallets from multiple
// backends.
URL() URL
```

```
// Status returns a textual status to aid the user in the current state of the
// wallet.
Status() string

// Open initializes access to a wallet instance. It is not meant to unlock or
// decrypt account keys, rather simply to establish a connection to hardware
// wallets and/or to access derivation seeds.
//
// The passphrase parameter may or may not be used by the implementation of a
// particular wallet instance. The reason there is no passwordless open method
// is to strive towards a uniform wallet handling, oblivious to the different
// backend providers.
//
// Please note, if you open a wallet, you must close it to release any allocated
// resources (especially important when working with hardware wallets).
Open(passphrase string) error

// Close releases any resources held by an open wallet instance.
Close() error

// Accounts retrieves the list of signing accounts the wallet is currently aware
// of. For hierarchical deterministic wallets, the list will not be exhaustive,
// rather only contain the accounts explicitly pinned during account derivation.
Accounts() []Account

// Contains returns whether an account is part of this particular wallet or not.
Contains(account Account) bool

// Derive attempts to explicitly derive a hierarchical deterministic account at
// the specified derivation path. If requested, the derived account will be added
// to the wallet's tracked account list.
Derive(path DerivationPath, pin bool) (Account, error)

// SelfDerive sets a base account derivation path from which the wallet attempts
// to discover non zero accounts and automatically add them to list of tracked
// accounts.
//
// Note, self derivaton will increment the last component of the specified path
// opposed to decending into a child path to allow discovering accounts starting
// from non zero components.
//
// You can disable automatic account discovery by calling SelfDerive with a nil
```

```
// chain state reader.
SelfDerive(base DerivationPath, chain ethereum.ChainStateReader)

// SignHash requests the wallet to sign the given hash.
//
// It looks up the account specified either solely via its address contained within,
// or optionally with the aid of any location metadata from the embedded URL field.
//
// If the wallet requires additional authentication to sign the request (e.g.
// a password to decrypt the account, or a PIN code o verify the transaction),
// an AuthNeededError instance will be returned, containing infos for the user
// about which fields or actions are needed. The user may retry by providing
// the needed details via SignHashWithPassphrase, or by other means (e.g. unlock
// the account in a keystore).
SignHash(account Account, hash []byte) ([]byte, error)

// SignTx requests the wallet to sign the given transaction.
//
// It looks up the account specified either solely via its address contained within,
// or optionally with the aid of any location metadata from the embedded URL field.
//
// If the wallet requires additional authentication to sign the request (e.g.
// a password to decrypt the account, or a PIN code o verify the transaction),
// an AuthNeededError instance will be returned, containing infos for the user
// about which fields or actions are needed. The user may retry by providing
// the needed details via SignTxWithPassphrase, or by other means (e.g. unlock
// the account in a keystore).
SignTx(account Account, tx *types.Transaction, chainID *big.Int) (*types.Transaction, error)

// SignHashWithPassphrase requests the wallet to sign the given hash with the
// given passphrase as extra authentication information.
//
// It looks up the account specified either solely via its address contained within,
// or optionally with the aid of any location metadata from the embedded URL field.
SignHashWithPassphrase(account Account, passphrase string, hash []byte) ([]byte, error)

// SignTxWithPassphrase requests the wallet to sign the given transaction, with the
// given passphrase as extra authentication information.
//
// It looks up the account specified either solely via its address contained within,
// or optionally with the aid of any location metadata from the embedded URL field.
SignTxWithPassphrase(account Account, passphrase string, tx *types.Transaction, chainID
```

```
*big.Int) (*types.Transaction, error)
}

// Backend is a "wallet provider" that may contain a batch of accounts they can
// sign transactions with and upon request, do so.
type Backend interface {
// Wallets retrieves the list of wallets the backend is currently aware of.
//
// The returned wallets are not opened by default. For software HD wallets this
// means that no base seeds are decrypted, and for hardware wallets that no actual
// connection is established.
//
// The resulting wallet list will be sorted alphabetically based on its internal
// URL assigned by the backend. Since wallets (especially hardware) may come and
// go, the same wallet might appear at a different positions in the list during
// subsequent retrievals.
Wallets() []Wallet

// Subscribe creates an async subscription to receive notifications when the
// backend detects the arrival or departure of a wallet.
Subscribe(sink chan<- WalletEvent) event.Subscription
}

// WalletEvent is an event fired by an account backend when a wallet arrival or
// departure is detected.
type WalletEvent struct {
Wallet Wallet // Wallet instance arrived or departed
Arrive bool   // Whether the wallet was added or removed
}

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

package accounts

import (
"errors"
"fmt"
)

// ErrUnknownAccount is returned for any requested operation for which no backend
// provides the specified account.
```

```
var ErrUnknownAccount = errors.New("unknown account")

// ErrUnknownWallet is returned for any requested operation for which no backend
// provides the specified wallet.
var ErrUnknownWallet = errors.New("unknown wallet")

// ErrNotSupported is returned when an operation is requested from an account
// backend that it does not support.
var ErrNotSupported = errors.New("not supported")

// ErrInvalidPassphrase is returned when a decryption operation receives a bad
// passphrase.
var ErrInvalidPassphrase = errors.New("invalid passphrase")

// ErrWalletAlreadyOpen is returned if a wallet is attempted to be opened the
// second time.
var ErrWalletAlreadyOpen = errors.New("wallet already open")

// ErrWalletClosed is returned if a wallet is attempted to be opened the
// secodn time.
var ErrWalletClosed = errors.New("wallet closed")

// AuthNeededError is returned by backends for signing requests where the user
// is required to provide further authentication before signing can succeed.
//
// This usually means either that a password needs to be supplied, or perhaps a
// one time PIN code displayed by some hardware device.
type AuthNeededError struct {
Needed string // Extra authentication the user needs to provide
}

// NewAuthNeededError creates a new authentication error with the extra details
// about the needed fields set.
func NewAuthNeededError(needed string) error {
return &AuthNeededError{
Needed: needed,
}
}

// Error implements the standard error interfacel.
func (err *AuthNeededError) Error() string {
return fmt.Sprintf("authentication needed: %s", err.Needed)
```

```
}

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

package accounts

import (
"errors"
"fmt"
"math"
"math/big"
"strings"
)

// DefaultRootDerivationPath is the root path to which custom derivation endpoints
// are appended. As such, the first account will be at m/44'/60'/0'/0, the second
// at m/44'/60'/0'/1, etc.
var DefaultRootDerivationPath = DerivationPath{0x80000000 + 44, 0x80000000 + 60,
0x80000000 + 0}

// DefaultBaseDerivationPath is the base path from which custom derivation endpoints
// are incremented. As such, the first account will be at m/44'/60'/0'/0, the second
// at m/44'/60'/0'/1, etc.
var DefaultBaseDerivationPath = DerivationPath{0x80000000 + 44, 0x80000000 + 60,
0x80000000 + 0, 0}

// DerivationPath represents the computer friendly version of a hierarchical
// deterministic wallet account derivaion path.
//
// The BIP-32 spec https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki
// defines derivation paths to be of the form:
//
//   m / purpose' / coin_type' / account' / change / address_index
//
// The BIP-44 spec https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki
// defines that the `purpose` be 44' (or 0x8000002C) for crypto currencies, and
// SLIP-44 https://github.com/satoshilabs/slips/blob/master/slip-0044.md assigns
// the `coin_type` 60' (or 0x8000003C) to Ethereum.
//
// The root path for Ethereum is m/44'/60'/0'/0 according to the specification
// from https://github.com/ethereum/EIPs/issues/84, albeit it's not set in stone
```

```
// yet whether accounts should increment the last component or the children of
// that. We will go with the simpler approach of incrementing the last component.
type DerivationPath []uint32

// ParseDerivationPath converts a user specified derivation path string to the
// internal binary representation.
//
// Full derivation paths need to start with the `m/` prefix, relative derivation
// paths (which will get appended to the default root path) must not have prefixes
// in front of the first element. Whitespace is ignored.
func ParseDerivationPath(path string) (DerivationPath, error) {
var result DerivationPath

// Handle absolute or relative paths
components := strings.Split(path, "/")
switch {
case len(components) == 0:
return nil, errors.New("empty derivation path")

case strings.TrimSpace(components[0]) == "":
return nil, errors.New("ambiguous path: use 'm/' prefix for absolute paths, or no leading '/' for
relative ones")

case strings.TrimSpace(components[0]) == "m":
components = components[1:]

default:
result = append(result, DefaultRootDerivationPath...)
}
// All remaining components are relative, append one by one
if len(components) == 0 {
return nil, errors.New("empty derivation path") // Empty relative paths
}
for _, component := range components {
// Ignore any user added whitespace
component = strings.TrimSpace(component)
var value uint32

// Handle hardened paths
if strings.HasSuffix(component, "'") {
value = 0x80000000
component = strings.TrimSpace(strings.TrimSuffix(component, "'"))
```

```
}
// Handle the non hardened component
bigval, ok := new(big.Int).SetString(component, 0)
if !ok {
return nil, fmt.Errorf("invalid component: %s", component)
}
max := math.MaxUint32 - value
if bigval.Sign() < 0 || bigval.Cmp(big.NewInt(int64(max))) > 0 {
if value == 0 {
return nil, fmt.Errorf("component %v out of allowed range [0, %d]", bigval, max)
}
return nil, fmt.Errorf("component %v out of allowed hardened range [0, %d]", bigval, max)
}
value += uint32(bigval.Uint64())

// Append and repeat
result = append(result, value)
}
return result, nil
}

// String implements the stringer interface, converting a binary derivation path
// to its canonical representation.
func (path DerivationPath) String() string {
result := "m"
for _, component := range path {
var hardened bool
if component >= 0x80000000 {
component -= 0x80000000
hardened = true
}
result = fmt.Sprintf("%s/%d", result, component)
if hardened {
result += "'"
}
}
return result
}

29:F:\git\coin\ethereum\go-ethereum\accounts\hd_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.
```

```
package accounts

import (
"reflect"
"testing"
)

// Tests that HD derivation paths can be correctly parsed into our internal binary
// representation.
func TestHDPathParsing(t *testing.T) {
tests := []struct {
input  string
output DerivationPath
}{
// Plain absolute derivation paths
{"m/44'/60'/0'/0", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0}},
{"m/44'/60'/0'/128", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 128}},
{"m/44'/60'/0'/0'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0,
0x80000000 + 0}},
{"m/44'/60'/0'/128'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0,
0x80000000 + 128}},
{"m/2147483692/2147483708/2147483648/0", DerivationPath{0x80000000 + 44, 0x80000000 +
60, 0x80000000 + 0, 0}},
{"m/2147483692/2147483708/2147483648/2147483648", DerivationPath{0x80000000 + 44,
0x80000000 + 60, 0x80000000 + 0, 0x80000000 + 0}},

// Plain relative derivation paths
{"0", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0}},
{"128", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 128}},
{"0'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0x80000000 + 0}},
{"128'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0x80000000 +
128}},
{"2147483648", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0,
0x80000000 + 0}},

// Hexadecimal absolute derivation paths
{"m/0x2C'/0x3c'/0x00'/0x00", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 +
0, 0}},
{"m/0x2C'/0x3c'/0x00'/0x80", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 +
0, 128}},
{"m/0x2C'/0x3c'/0x00'/0x00'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 +
0, 0x80000000 + 0}},
```

{"m/0x2C'/0x3c'/0x00'/0x80'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0x80000000 + 128}},
{"m/0x8000002C/0x8000003c/0x80000000/0x00", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0}},
{"m/0x8000002C/0x8000003c/0x80000000/0x80000000", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0x80000000 + 0}},

// Hexadecimal relative derivation paths
{"0x00", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0}},
{"0x80", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 128}},
{"0x00'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0x80000000 + 0}},
{"0x80'", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0x80000000 + 128}},
{"0x80000000", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0x80000000 + 0}},

// Weird inputs just to ensure they work
{"m /   44'\n/\n   60\n\n\t'   /\n0 ' /\t\t0", DerivationPath{0x80000000 + 44, 0x80000000 + 60, 0x80000000 + 0, 0}},

// Invaid derivation paths
{"", nil},   // Empty relative derivation path
{"m", nil},   // Empty absolute derivation path
{"m/", nil},   // Missing last derivation component
{"/44'/60'/0'/0", nil}, // Absolute path without m prefix, might be user error
{"m/2147483648'", nil}, // Overflows 32 bit integer
{"m/-1'", nil},   // Cannot contain negative number
}
for i, tt := range tests {
if path, err := ParseDerivationPath(tt.input); !reflect.DeepEqual(path, tt.output) {
t.Errorf("test %d: parse mismatch: have %v (%v), want %v", i, path, err, tt.output)
} else if path == nil && err == nil {
t.Errorf("test %d: nil path and error: %v", i, err)
}
}
}

30:F:\git\coin\ethereum\go-ethereum\accounts\keystore\account_cache.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package keystore

```
import (
"bufio"
"encoding/json"
"fmt"
"io/ioutil"
"os"
"path/filepath"
"sort"
"strings"
"sync"
"time"

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

// Minimum amount of time between cache reloads. This limit applies if the platform does
// not support change notifications. It also applies if the keystore directory does not
// exist yet, the code will attempt to create a watcher at most this often.
const minReloadInterval = 2 * time.Second

type accountsByURL []accounts.Account

func (s accountsByURL) Len() int           { return len(s) }
func (s accountsByURL) Less(i, j int) bool { return s[i].URL.Cmp(s[j].URL) < 0 }
func (s accountsByURL) Swap(i, j int)      { s[i], s[j] = s[j], s[i] }

// AmbiguousAddrError is returned when attempting to unlock
// an address for which more than one file exists.
type AmbiguousAddrError struct {
Addr    common.Address
Matches []accounts.Account
}

func (err *AmbiguousAddrError) Error() string {
files := ""
for i, a := range err.Matches {
files += a.URL.Path
if i < len(err.Matches)-1 {
files += ", "
}
```

```
}
return fmt.Sprintf("multiple keys match address (%s)", files)
}

// accountCache is a live index of all accounts in the keystore.
type accountCache struct {
keydir   string
watcher  *watcher
mu       sync.Mutex
all      accountsByURL
byAddr   map[common.Address][]accounts.Account
throttle *time.Timer
notify   chan struct{}
}

func newAccountCache(keydir string) (*accountCache, chan struct{}) {
ac := &accountCache{
keydir: keydir,
byAddr: make(map[common.Address][]accounts.Account),
notify: make(chan struct{}, 1),
}
ac.watcher = newWatcher(ac)
return ac, ac.notify
}

func (ac *accountCache) accounts() []accounts.Account {
ac.maybeReload()
ac.mu.Lock()
defer ac.mu.Unlock()
cpy := make([]accounts.Account, len(ac.all))
copy(cpy, ac.all)
return cpy
}

func (ac *accountCache) hasAddress(addr common.Address) bool {
ac.maybeReload()
ac.mu.Lock()
defer ac.mu.Unlock()
return len(ac.byAddr[addr]) > 0
}

func (ac *accountCache) add(newAccount accounts.Account) {
```

```
ac.mu.Lock()
defer ac.mu.Unlock()

i := sort.Search(len(ac.all), func(i int) bool { return ac.all[i].URL.Cmp(newAccount.URL) >= 0 })
if i < len(ac.all) && ac.all[i] == newAccount {
return
}
// newAccount is not in the cache.
ac.all = append(ac.all, accounts.Account{})
copy(ac.all[i+1:], ac.all[i:])
ac.all[i] = newAccount
ac.byAddr[newAccount.Address] = append(ac.byAddr[newAccount.Address], newAccount)
}

// note: removed needs to be unique here (i.e. both File and Address must be set).
func (ac *accountCache) delete(removed accounts.Account) {
ac.mu.Lock()
defer ac.mu.Unlock()

ac.all = removeAccount(ac.all, removed)
if ba := removeAccount(ac.byAddr[removed.Address], removed); len(ba) == 0 {
delete(ac.byAddr, removed.Address)
} else {
ac.byAddr[removed.Address] = ba
}
}

func removeAccount(slice []accounts.Account, elem accounts.Account) []accounts.Account {
for i := range slice {
if slice[i] == elem {
return append(slice[:i], slice[i+1:]...)
}
}
return slice
}

// find returns the cached account for address if there is a unique match.
// The exact matching rules are explained by the documentation of accounts.Account.
// Callers must hold ac.mu.
func (ac *accountCache) find(a accounts.Account) (accounts.Account, error) {
// Limit search to address candidates if possible.
matches := ac.all
```

```
if (a.Address != common.Address{}) {
matches = ac.byAddr[a.Address]
}
if a.URL.Path != "" {
// If only the basename is specified, complete the path.
if !strings.ContainsRune(a.URL.Path, filepath.Separator) {
a.URL.Path = filepath.Join(ac.keydir, a.URL.Path)
}
for i := range matches {
if matches[i].URL == a.URL {
return matches[i], nil
}
}
if (a.Address == common.Address{}) {
return accounts.Account{}, ErrNoMatch
}
}
switch len(matches) {
case 1:
return matches[0], nil
case 0:
return accounts.Account{}, ErrNoMatch
default:
err := &AmbiguousAddrError{Addr: a.Address, Matches: make([]accounts.Account, len(matches))}
copy(err.Matches, matches)
return accounts.Account{}, err
}
}

func (ac *accountCache) maybeReload() {
ac.mu.Lock()
defer ac.mu.Unlock()

if ac.watcher.running {
return // A watcher is running and will keep the cache up-to-date.
}
if ac.throttle == nil {
ac.throttle = time.NewTimer(0)
} else {
select {
case <-ac.throttle.C:
default:
```

```
return // The cache was reloaded recently.
}
}
ac.watcher.start()
ac.reload()
ac.throttle.Reset(minReloadInterval)
}

func (ac *accountCache) close() {
ac.mu.Lock()
ac.watcher.close()
if ac.throttle != nil {
ac.throttle.Stop()
}
if ac.notify != nil {
close(ac.notify)
ac.notify = nil
}
ac.mu.Unlock()
}

// reload caches addresses of existing accounts.
// Callers must hold ac.mu.
func (ac *accountCache) reload() {
accounts, err := ac.scan()
if err != nil {
log.Debug("Failed to reload keystore contents", "err", err)
}
ac.all = accounts
sort.Sort(ac.all)
for k := range ac.byAddr {
delete(ac.byAddr, k)
}
for _, a := range accounts {
ac.byAddr[a.Address] = append(ac.byAddr[a.Address], a)
}
select {
case ac.notify <- struct{}{}:
default:
}
log.Debug("Reloaded keystore contents", "accounts", len(ac.all))
}
```

```
func (ac *accountCache) scan() ([]accounts.Account, error) {
files, err := ioutil.ReadDir(ac.keydir)
if err != nil {
return nil, err
}

var (
buf     = new(bufio.Reader)
addrs   []accounts.Account
keyJSON struct {
Address string `json:"address"`
}
)
for _, fi := range files {
path := filepath.Join(ac.keydir, fi.Name())
if skipKeyFile(fi) {
log.Trace("Ignoring file on account scan", "path", path)
continue
}
logger := log.New("path", path)

fd, err := os.Open(path)
if err != nil {
logger.Trace("Failed to open keystore file", "err", err)
continue
}
buf.Reset(fd)
// Parse the address.
keyJSON.Address = ""
err = json.NewDecoder(buf).Decode(&keyJSON)
addr := common.HexToAddress(keyJSON.Address)
switch {
case err != nil:
logger.Debug("Failed to decode keystore key", "err", err)
case (addr == common.Address{}):
logger.Debug("Failed to decode keystore key", "err", "missing or zero address")
default:
addrs = append(addrs, accounts.Account{Address: addr, URL: accounts.URL{Scheme:
KeyStoreScheme, Path: path}})
}
fd.Close()
```

```
}
return addrs, err
}

func skipKeyFile(fi os.FileInfo) bool {
// Skip editor backups and UNIX-style hidden files.
if strings.HasSuffix(fi.Name(), "~") || strings.HasPrefix(fi.Name(), ".") {
return true
}
// Skip misc special files, directories (yes, symlinks too).
if fi.IsDir() || fi.Mode()&os.ModeType != 0 {
return true
}
return false
}

31:F:\git\coin\ethereum\go-ethereum\accounts\keystore\account_cache_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package keystore

import (
"fmt"
"math/rand"
"os"
"path/filepath"
"reflect"
"sort"
"testing"
"time"

"github.com/cespare/cp"
"github.com/davecgh/go-spew/spew"
"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/common"
)

var (
cachetestDir, _   = filepath.Abs(filepath.Join("testdata", "keystore"))
cachetestAccounts = []accounts.Account{
{
Address: common.HexToAddress("7ef5a6135f1fd6a02593eedc869c6d41d934aef8"),
```

```
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(cachetestDir, "UTC--2016-
03-22T12-57-55.920751759Z--7ef5a6135f1fd6a02593eedc869c6d41d934aef8")},
},
{
Address: common.HexToAddress("f466859ead1932d743d622cb74fc058882e8648a"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(cachetestDir, "aaa")},
},
{
Address: common.HexToAddress("289d485d9771714cce91d3393d764e1311907acc"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(cachetestDir, "zzz")},
},
}
)

func TestWatchNewFile(t *testing.T) {
t.Parallel()

dir, ks := tmpKeyStore(t, false)
defer os.RemoveAll(dir)

// Ensure the watcher is started before adding any files.
ks.Accounts()
time.Sleep(200 * time.Millisecond)

// Move in the files.
wantAccounts := make([]accounts.Account, len(cachetestAccounts))
for i := range cachetestAccounts {
wantAccounts[i] = accounts.Account{
Address: cachetestAccounts[i].Address,
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(dir,
filepath.Base(cachetestAccounts[i].URL.Path))},
}
if err := cp.CopyFile(wantAccounts[i].URL.Path, cachetestAccounts[i].URL.Path); err != nil {
t.Fatal(err)
}
}

// ks should see the accounts.
var list []accounts.Account
for d := 200 * time.Millisecond; d < 5*time.Second; d *= 2 {
list = ks.Accounts()
if reflect.DeepEqual(list, wantAccounts) {
```

```
// ks should have also received change notifications
select {
case <-ks.changes:
default:
t.Fatalf("wasn't notified of new accounts")
}
return
}
time.Sleep(d)
}
t.Errorf("got %s, want %s", spew.Sdump(list), spew.Sdump(wantAccounts))
}

func TestWatchNoDir(t *testing.T) {
t.Parallel()

// Create ks but not the directory that it watches.
rand.Seed(time.Now().UnixNano())
dir := filepath.Join(os.TempDir(), fmt.Sprintf("eth-keystore-watch-test-%d-%d", os.Getpid(),
rand.Int()))
ks := NewKeyStore(dir, LightScryptN, LightScryptP)

list := ks.Accounts()
if len(list) > 0 {
t.Error("initial account list not empty:", list)
}
time.Sleep(100 * time.Millisecond)

// Create the directory and copy a key file into it.
os.MkdirAll(dir, 0700)
defer os.RemoveAll(dir)
file := filepath.Join(dir, "aaa")
if err := cp.CopyFile(file, cachetestAccounts[0].URL.Path); err != nil {
t.Fatal(err)
}

// ks should see the account.
wantAccounts := []accounts.Account{cachetestAccounts[0]}
wantAccounts[0].URL = accounts.URL{Scheme: KeyStoreScheme, Path: file}
for d := 200 * time.Millisecond; d < 8*time.Second; d *= 2 {
list = ks.Accounts()
if reflect.DeepEqual(list, wantAccounts) {
```

```
// ks should have also received change notifications
select {
case <-ks.changes:
default:
t.Fatalf("wasn't notified of new accounts")
}
return
}
time.Sleep(d)
}
t.Errorf("\ngot  %v\nwant %v", list, wantAccounts)
}

func TestCacheInitialReload(t *testing.T) {
cache, _ := newAccountCache(cachetestDir)
accounts := cache.accounts()
if !reflect.DeepEqual(accounts, cachetestAccounts) {
t.Fatalf("got initial accounts: %swant %s", spew.Sdump(accounts),
spew.Sdump(cachetestAccounts))
}
}

func TestCacheAddDeleteOrder(t *testing.T) {
cache, _ := newAccountCache("testdata/no-such-dir")
cache.watcher.running = true // prevent unexpected reloads

accs := []accounts.Account{
{
Address: common.HexToAddress("095e7baea6a6c7c4c2dfeb977efac326af552d87"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: "-309830980"},
},
{
Address: common.HexToAddress("2cac1adea150210703ba75ed097ddfe24e14f213"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: "ggg"},
},
{
Address: common.HexToAddress("8bda78331c916a08481428e4b07c96d3e916d165"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: "zzzzzz-the-very-last-one.keyXXX"},
},
{
Address: common.HexToAddress("d49ff4eeb0b2686ed89c0fc0f2b6ea533ddbbd5e"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: "SOMETHING.key"},
```

```
},
{
Address: common.HexToAddress("7ef5a6135f1fd6a02593eedc869c6d41d934aef8"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: "UTC--2016-03-22T12-57-
55.920751759Z--7ef5a6135f1fd6a02593eedc869c6d41d934aef8"},
},
{
Address: common.HexToAddress("f466859ead1932d743d622cb74fc058882e8648a"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: "aaa"},
},
{
Address: common.HexToAddress("289d485d9771714cce91d3393d764e1311907acc"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: "zzz"},
},
}
for _, a := range accs {
cache.add(a)
}
// Add some of them twice to check that they don't get reinserted.
cache.add(accs[0])
cache.add(accs[2])

// Check that the account list is sorted by filename.
wantAccounts := make([]accounts.Account, len(accs))
copy(wantAccounts, accs)
sort.Sort(accountsByURL(wantAccounts))
list := cache.accounts()
if !reflect.DeepEqual(list, wantAccounts) {
t.Fatalf("got accounts: %s\nwant %s", spew.Sdump(accs), spew.Sdump(wantAccounts))
}
for _, a := range accs {
if !cache.hasAddress(a.Address) {
t.Errorf("expected hasAccount(%x) to return true", a.Address)
}
}
if cache.hasAddress(common.HexToAddress("fd9bd350f08ee3c0c19b85a8e16114a11a60aa4e"))
{
t.Errorf("expected hasAccount(%x) to return false",
common.HexToAddress("fd9bd350f08ee3c0c19b85a8e16114a11a60aa4e"))
}

// Delete a few keys from the cache.
```

```
for i := 0; i < len(accs); i += 2 {
cache.delete(wantAccounts[i])
}
cache.delete(accounts.Account{Address:
common.HexToAddress("fd9bd350f08ee3c0c19b85a8e16114a11a60aa4e"), URL:
accounts.URL{Scheme: KeyStoreScheme, Path: "something"}})

// Check content again after deletion.
wantAccountsAfterDelete := []accounts.Account{
wantAccounts[1],
wantAccounts[3],
wantAccounts[5],
}
list = cache.accounts()
if !reflect.DeepEqual(list, wantAccountsAfterDelete) {
t.Fatalf("got accounts after delete: %s\nwant %s", spew.Sdump(list),
spew.Sdump(wantAccountsAfterDelete))
}
for _, a := range wantAccountsAfterDelete {
if !cache.hasAddress(a.Address) {
t.Errorf("expected hasAccount(%x) to return true", a.Address)
}
}
if cache.hasAddress(wantAccounts[0].Address) {
t.Errorf("expected hasAccount(%x) to return false", wantAccounts[0].Address)
}
}

func TestCacheFind(t *testing.T) {
dir := filepath.Join("testdata", "dir")
cache, _ := newAccountCache(dir)
cache.watcher.running = true // prevent unexpected reloads

accs := []accounts.Account{
{
Address: common.HexToAddress("095e7baea6a6c7c4c2dfeb977efac326af552d87"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(dir, "a.key")},
},
{
Address: common.HexToAddress("2cac1adea150210703ba75ed097ddfe24e14f213"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(dir, "b.key")},
},
```

```
{
Address: common.HexToAddress("d49ff4eeb0b2686ed89c0fc0f2b6ea533ddbbd5e"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(dir, "c.key")},
},
{
Address: common.HexToAddress("d49ff4eeb0b2686ed89c0fc0f2b6ea533ddbbd5e"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(dir, "c2.key")},
},
}
for _, a := range accs {
cache.add(a)
}

nomatchAccount := accounts.Account{
Address: common.HexToAddress("f466859ead1932d743d622cb74fc058882e8648a"),
URL:     accounts.URL{Scheme: KeyStoreScheme, Path: filepath.Join(dir, "something")},
}
tests := []struct {
Query      accounts.Account
WantResult accounts.Account
WantError  error
}{
// by address
{Query: accounts.Account{Address: accs[0].Address}, WantResult: accs[0]},
// by file
{Query: accounts.Account{URL: accs[0].URL}, WantResult: accs[0]},
// by basename
{Query: accounts.Account{URL: accounts.URL{Scheme: KeyStoreScheme, Path:
filepath.Base(accs[0].URL.Path)}}, WantResult: accs[0]},
// by file and address
{Query: accs[0], WantResult: accs[0]},
// ambiguous address, tie resolved by file
{Query: accs[2], WantResult: accs[2]},
// ambiguous address error
{
Query: accounts.Account{Address: accs[2].Address},
WantError: &AmbiguousAddrError{
Addr:    accs[2].Address,
Matches: []accounts.Account{accs[2], accs[3]},
},
},
// no match error
```

```
{Query: nomatchAccount, WantError: ErrNoMatch},
{Query: accounts.Account{URL: nomatchAccount.URL}, WantError: ErrNoMatch},
{Query: accounts.Account{URL: accounts.URL{Scheme: KeyStoreScheme, Path:
filepath.Base(nomatchAccount.URL.Path)}}, WantError: ErrNoMatch},
{Query: accounts.Account{Address: nomatchAccount.Address}, WantError: ErrNoMatch},
}
for i, test := range tests {
a, err := cache.find(test.Query)
if !reflect.DeepEqual(err, test.WantError) {
t.Errorf("test %d: error mismatch for query %v\ngot %q\nwant %q", i, test.Query, err,
test.WantError)
continue
}
if a != test.WantResult {
t.Errorf("test %d: result mismatch for query %v\ngot %v\nwant %v", i, test.Query, a,
test.WantResult)
continue
}
}
}
```

32:F:\git\coin\ethereum\go-ethereum\accounts\keystore\key.go

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

package keystore

import (
"bytes"
"crypto/ecdsa"
"encoding/hex"
"encoding/json"
"fmt"
"io"
"io/ioutil"
"os"
"path/filepath"
"strings"
"time"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/crypto"
```

```
"github.com/pborman/uuid"
)

const (
version = 3
)

type Key struct {
Id uuid.UUID // Version 4 "random" for unique id not derived from key data
// to simplify lookups we also store the address
Address common.Address
// we only store privkey as pubkey/address can be derived from it
// privkey in this struct is always in plaintext
PrivateKey *ecdsa.PrivateKey
}

type keyStore interface {
// Loads and decrypts the key from disk.
GetKey(addr common.Address, filename string, auth string) (*Key, error)
// Writes and encrypts the key.
StoreKey(filename string, k *Key, auth string) error
// Joins filename with the key directory unless it is already absolute.
JoinPath(filename string) string
}

type plainKeyJSON struct {
Address    string `json:"address"`
PrivateKey string `json:"privatekey"`
Id         string `json:"id"`
Version    int    `json:"version"`
}

type encryptedKeyJSONV3 struct {
Address string     `json:"address"`
Crypto  cryptoJSON `json:"crypto"`
Id      string     `json:"id"`
Version int        `json:"version"`
}

type encryptedKeyJSONV1 struct {
Address string     `json:"address"`
Crypto  cryptoJSON `json:"crypto"`
```

```
Id      string     `json:"id"`
Version string     `json:"version"`
}

type cryptoJSON struct {
Cipher       string                `json:"cipher"`
CipherText   string                `json:"ciphertext"`
CipherParams cipherparamsJSON       `json:"cipherparams"`
KDF          string                `json:"kdf"`
KDFParams    map[string]interface{} `json:"kdfparams"`
MAC          string                `json:"mac"`
}

type cipherparamsJSON struct {
IV string `json:"iv"`
}

type scryptParamsJSON struct {
N     int    `json:"n"`
R     int    `json:"r"`
P     int    `json:"p"`
DkLen int    `json:"dklen"`
Salt  string `json:"salt"`
}

func (k *Key) MarshalJSON() (j []byte, err error) {
jStruct := plainKeyJSON{
hex.EncodeToString(k.Address[:]),
hex.EncodeToString(crypto.FromECDSA(k.PrivateKey)),
k.Id.String(),
version,
}
j, err = json.Marshal(jStruct)
return j, err
}

func (k *Key) UnmarshalJSON(j []byte) (err error) {
keyJSON := new(plainKeyJSON)
err = json.Unmarshal(j, &keyJSON)
if err != nil {
return err
}
```

```
u := new(uuid.UUID)
*u = uuid.Parse(keyJSON.Id)
k.Id = *u
addr, err := hex.DecodeString(keyJSON.Address)
if err != nil {
return err
}
privkey, err := crypto.HexToECDSA(keyJSON.PrivateKey)
if err != nil {
return err
}

k.Address = common.BytesToAddress(addr)
k.PrivateKey = privkey

return nil
}

func newKeyFromECDSA(privateKeyECDSA *ecdsa.PrivateKey) *Key {
id := uuid.NewRandom()
key := &Key{
Id:         id,
Address:    crypto.PubkeyToAddress(privateKeyECDSA.PublicKey),
PrivateKey: privateKeyECDSA,
}
return key
}

// NewKeyForDirectICAP generates a key whose address fits into < 155 bits so it can fit
// into the Direct ICAP spec. for simplicity and easier compatibility with other libs, we
// retry until the first byte is 0.
func NewKeyForDirectICAP(rand io.Reader) *Key {
randBytes := make([]byte, 64)
_, err := rand.Read(randBytes)
if err != nil {
panic("key generation: could not read from random source: " + err.Error())
}
reader := bytes.NewReader(randBytes)
privateKeyECDSA, err := ecdsa.GenerateKey(crypto.S256(), reader)
if err != nil {
panic("key generation: ecdsa.GenerateKey failed: " + err.Error())
```

```
}
key := newKeyFromECDSA(privateKeyECDSA)
if !strings.HasPrefix(key.Address.Hex(), "0x00") {
return NewKeyForDirectICAP(rand)
}
return key
}

func newKey(rand io.Reader) (*Key, error) {
privateKeyECDSA, err := ecdsa.GenerateKey(crypto.S256(), rand)
if err != nil {
return nil, err
}
return newKeyFromECDSA(privateKeyECDSA), nil
}

func storeNewKey(ks keyStore, rand io.Reader, auth string) (*Key, accounts.Account, error) {
key, err := newKey(rand)
if err != nil {
return nil, accounts.Account{}, err
}
a := accounts.Account{Address: key.Address, URL: accounts.URL{Scheme: KeyStoreScheme,
Path: ks.JoinPath(keyFileName(key.Address))}}
if err := ks.StoreKey(a.URL.Path, key, auth); err != nil {
zeroKey(key.PrivateKey)
return nil, a, err
}
return key, a, err
}

func writeKeyFile(file string, content []byte) error {
// Create the keystore directory with appropriate permissions
// in case it is not present yet.
const dirPerm = 0700
if err := os.MkdirAll(filepath.Dir(file), dirPerm); err != nil {
return err
}
// Atomic write: create a temporary hidden file first
// then move it into place. TempFile assigns mode 0600.
f, err := ioutil.TempFile(filepath.Dir(file), "."+filepath.Base(file)+".tmp")
if err != nil {
return err
```

```
}
if _, err := f.Write(content); err != nil {
f.Close()
os.Remove(f.Name())
return err
}
f.Close()
return os.Rename(f.Name(), file)
}

// keyFileName implements the naming convention for keyfiles:
// UTC--<created_at UTC ISO8601>-<address hex>
func keyFileName(keyAddr common.Address) string {
ts := time.Now().UTC()
return fmt.Sprintf("UTC--%s--%s", toISO8601(ts), hex.EncodeToString(keyAddr[:]))
}

func toISO8601(t time.Time) string {
var tz string
name, offset := t.Zone()
if name == "UTC" {
tz = "Z"
} else {
tz = fmt.Sprintf("%03d00", offset/3600)
}
return fmt.Sprintf("%04d-%02d-%02dT%02d-%02d-%02d.%09d%s", t.Year(), t.Month(), t.Day(),
t.Hour(), t.Minute(), t.Second(), t.Nanosecond(), tz)
}

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

// Package keystore implements encrypted storage of secp256k1 private keys.
//
// Keys are stored as encrypted JSON files according to the Web3 Secret Storage specification.
// See https://github.com/ethereum/wiki/wiki/Web3-Secret-Storage-Definition for more information.
package keystore

import (
"crypto/ecdsa"
crand "crypto/rand"
"errors"
```

```
"fmt"
"math/big"
"os"
"path/filepath"
"reflect"
"runtime"
"sync"
"time"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/event"
)

var (
ErrLocked  = accounts.NewAuthNeededError("password or unlock")
ErrNoMatch = errors.New("no key for given address or file")
ErrDecrypt = errors.New("could not decrypt key with given passphrase")
)

// KeyStoreType is the reflect type of a keystore backend.
var KeyStoreType = reflect.TypeOf(&KeyStore{})

// KeyStoreScheme is the protocol scheme prefixing account and wallet URLs.
var KeyStoreScheme = "keystore"

// Maximum time between wallet refreshes (if filesystem notifications don't work).
const walletRefreshCycle = 3 * time.Second

// KeyStore manages a key storage directory on disk.
type KeyStore struct {
storage  keyStore                     // Storage backend, might be cleartext or encrypted
cache    *accountCache                // In-memory account cache over the filesystem storage
changes  chan struct{}                // Channel receiving change notifications from the cache
unlocked map[common.Address]*unlocked // Currently unlocked account (decrypted private keys)

wallets     []accounts.Wallet       // Wallet wrappers around the individual key files
updateFeed  event.Feed             // Event feed to notify wallet additions/removals
updateScope event.SubscriptionScope // Subscription scope tracking current live listeners
updating    bool                   // Whether the event notification loop is running
```

```
mu sync.RWMutex
}

type unlocked struct {
*Key
abort chan struct{}
}

// NewKeyStore creates a keystore for the given directory.
func NewKeyStore(keydir string, scryptN, scryptP int) *KeyStore {
keydir, _ = filepath.Abs(keydir)
ks := &KeyStore{storage: &keyStorePassphrase{keydir, scryptN, scryptP}}
ks.init(keydir)
return ks
}

// NewPlaintextKeyStore creates a keystore for the given directory.
// Deprecated: Use NewKeyStore.
func NewPlaintextKeyStore(keydir string) *KeyStore {
keydir, _ = filepath.Abs(keydir)
ks := &KeyStore{storage: &keyStorePlain{keydir}}
ks.init(keydir)
return ks
}

func (ks *KeyStore) init(keydir string) {
// Lock the mutex since the account cache might call back with events
ks.mu.Lock()
defer ks.mu.Unlock()

// Initialize the set of unlocked keys and the account cache
ks.unlocked = make(map[common.Address]*unlocked)
ks.cache, ks.changes = newAccountCache(keydir)

// TODO: In order for this finalizer to work, there must be no references
// to ks. addressCache doesn't keep a reference but unlocked keys do,
// so the finalizer will not trigger until all timed unlocks have expired.
runtime.SetFinalizer(ks, func(m *KeyStore) {
m.cache.close()
})
// Create the initial list of wallets from the cache
```

```
accs := ks.cache.accounts()
ks.wallets = make([]accounts.Wallet, len(accs))
for i := 0; i < len(accs); i++ {
ks.wallets[i] = &keystoreWallet{account: accs[i], keystore: ks}
}
}

// Wallets implements accounts.Backend, returning all single-key wallets from the
// keystore directory.
func (ks *KeyStore) Wallets() []accounts.Wallet {
// Make sure the list of wallets is in sync with the account cache
ks.refreshWallets()

ks.mu.RLock()
defer ks.mu.RUnlock()

cpy := make([]accounts.Wallet, len(ks.wallets))
copy(cpy, ks.wallets)
return cpy
}

// refreshWallets retrieves the current account list and based on that does any
// necessary wallet refreshes.
func (ks *KeyStore) refreshWallets() {
// Retrieve the current list of accounts
ks.mu.Lock()
accs := ks.cache.accounts()

// Transform the current list of wallets into the new one
wallets := make([]accounts.Wallet, 0, len(accs))
events := []accounts.WalletEvent{}

for _, account := range accs {
// Drop wallets while they were in front of the next account
for len(ks.wallets) > 0 && ks.wallets[0].URL().Cmp(account.URL) < 0 {
events = append(events, accounts.WalletEvent{Wallet: ks.wallets[0], Arrive: false})
ks.wallets = ks.wallets[1:]
}
// If there are no more wallets or the account is before the next, wrap new wallet
if len(ks.wallets) == 0 || ks.wallets[0].URL().Cmp(account.URL) > 0 {
wallet := &keystoreWallet{account: account, keystore: ks}
```

```
events = append(events, accounts.WalletEvent{Wallet: wallet, Arrive: true})
wallets = append(wallets, wallet)
continue
}
// If the account is the same as the first wallet, keep it
if ks.wallets[0].Accounts()[0] == account {
wallets = append(wallets, ks.wallets[0])
ks.wallets = ks.wallets[1:]
continue
}
}
// Drop any leftover wallets and set the new batch
for _, wallet := range ks.wallets {
events = append(events, accounts.WalletEvent{Wallet: wallet, Arrive: false})
}
ks.wallets = wallets
ks.mu.Unlock()

// Fire all wallet events and return
for _, event := range events {
ks.updateFeed.Send(event)
}
}

// Subscribe implements accounts.Backend, creating an async subscription to
// receive notifications on the addition or removal of keystore wallets.
func (ks *KeyStore) Subscribe(sink chan<- accounts.WalletEvent) event.Subscription {
// We need the mutex to reliably start/stop the update loop
ks.mu.Lock()
defer ks.mu.Unlock()

// Subscribe the caller and track the subscriber count
sub := ks.updateScope.Track(ks.updateFeed.Subscribe(sink))

// Subscribers require an active notification loop, start it
if !ks.updating {
ks.updating = true
go ks.updater()
}
return sub
}
```

```
// updater is responsible for maintaining an up-to-date list of wallets stored in
// the keystore, and for firing wallet addition/removal events. It listens for
// account change events from the underlying account cache, and also periodically
// forces a manual refresh (only triggers for systems where the filesystem notifier
// is not running).
func (ks *KeyStore) updater() {
for {
// Wait for an account update or a refresh timeout
select {
case <-ks.changes:
case <-time.After(walletRefreshCycle):
}
// Run the wallet refresher
ks.refreshWallets()

// If all our subscribers left, stop the updater
ks.mu.Lock()
if ks.updateScope.Count() == 0 {
ks.updating = false
ks.mu.Unlock()
return
}
ks.mu.Unlock()
}
}

// HasAddress reports whether a key with the given address is present.
func (ks *KeyStore) HasAddress(addr common.Address) bool {
return ks.cache.hasAddress(addr)
}

// Accounts returns all key files present in the directory.
func (ks *KeyStore) Accounts() []accounts.Account {
return ks.cache.accounts()
}

// Delete deletes the key matched by account if the passphrase is correct.
// If the account contains no filename, the address must match a unique key.
func (ks *KeyStore) Delete(a accounts.Account, passphrase string) error {
// Decrypting the key isn't really necessary, but we do
// it anyway to check the password and zero out the key
// immediately afterwards.
```

```
a, key, err := ks.getDecryptedKey(a, passphrase)
if key != nil {
zeroKey(key.PrivateKey)
}
if err != nil {
return err
}
// The order is crucial here. The key is dropped from the
// cache after the file is gone so that a reload happening in
// between won't insert it into the cache again.
err = os.Remove(a.URL.Path)
if err == nil {
ks.cache.delete(a)
ks.refreshWallets()
}
return err
}

// SignHash calculates a ECDSA signature for the given hash. The produced
// signature is in the [R || S || V] format where V is 0 or 1.
func (ks *KeyStore) SignHash(a accounts.Account, hash []byte) ([]byte, error) {
// Look up the key to sign with and abort if it cannot be found
ks.mu.RLock()
defer ks.mu.RUnlock()

unlockedKey, found := ks.unlocked[a.Address]
if !found {
return nil, ErrLocked
}
// Sign the hash using plain ECDSA operations
return crypto.Sign(hash, unlockedKey.PrivateKey)
}

// SignTx signs the given transaction with the requested account.
func (ks *KeyStore) SignTx(a accounts.Account, tx *types.Transaction, chainID *big.Int)
(*types.Transaction, error) {
// Look up the key to sign with and abort if it cannot be found
ks.mu.RLock()
defer ks.mu.RUnlock()

unlockedKey, found := ks.unlocked[a.Address]
if !found {
```

```
return nil, ErrLocked
}
// Depending on the presence of the chain ID, sign with EIP155 or homestead
if chainID != nil {
return types.SignTx(tx, types.NewEIP155Signer(chainID), unlockedKey.PrivateKey)
}
return types.SignTx(tx, types.HomesteadSigner{}, unlockedKey.PrivateKey)
}

// SignHashWithPassphrase signs hash if the private key matching the given address
// can be decrypted with the given passphrase. The produced signature is in the
// [R || S || V] format where V is 0 or 1.
func (ks *KeyStore) SignHashWithPassphrase(a accounts.Account, passphrase string, hash
[]byte) (signature []byte, err error) {
_, key, err := ks.getDecryptedKey(a, passphrase)
if err != nil {
return nil, err
}
defer zeroKey(key.PrivateKey)
return crypto.Sign(hash, key.PrivateKey)
}

// SignTxWithPassphrase signs the transaction if the private key matching the
// given address can be decrypted with the given passphrase.
func (ks *KeyStore) SignTxWithPassphrase(a accounts.Account, passphrase string, tx
*types.Transaction, chainID *big.Int) (*types.Transaction, error) {
_, key, err := ks.getDecryptedKey(a, passphrase)
if err != nil {
return nil, err
}
defer zeroKey(key.PrivateKey)

// Depending on the presence of the chain ID, sign with EIP155 or homestead
if chainID != nil {
return types.SignTx(tx, types.NewEIP155Signer(chainID), key.PrivateKey)
}
return types.SignTx(tx, types.HomesteadSigner{}, key.PrivateKey)
}

// Unlock unlocks the given account indefinitely.
func (ks *KeyStore) Unlock(a accounts.Account, passphrase string) error {
return ks.TimedUnlock(a, passphrase, 0)
```

```
}

// Lock removes the private key with the given address from memory.
func (ks *KeyStore) Lock(addr common.Address) error {
ks.mu.Lock()
if unl, found := ks.unlocked[addr]; found {
ks.mu.Unlock()
ks.expire(addr, unl, time.Duration(0)*time.Nanosecond)
} else {
ks.mu.Unlock()
}
return nil
}

// TimedUnlock unlocks the given account with the passphrase. The account
// stays unlocked for the duration of timeout. A timeout of 0 unlocks the account
// until the program exits. The account must match a unique key file.
//
// If the account address is already unlocked for a duration, TimedUnlock extends or
// shortens the active unlock timeout. If the address was previously unlocked
// indefinitely the timeout is not altered.
func (ks *KeyStore) TimedUnlock(a accounts.Account, passphrase string, timeout time.Duration)
error {
a, key, err := ks.getDecryptedKey(a, passphrase)
if err != nil {
return err
}

ks.mu.Lock()
defer ks.mu.Unlock()
u, found := ks.unlocked[a.Address]
if found {
if u.abort == nil {
// The address was unlocked indefinitely, so unlocking
// it with a timeout would be confusing.
zeroKey(key.PrivateKey)
return nil
}
// Terminate the expire goroutine and replace it below.
close(u.abort)
}
if timeout > 0 {
```

```
u = &unlocked{Key: key, abort: make(chan struct{})}
go ks.expire(a.Address, u, timeout)
} else {
u = &unlocked{Key: key}
}
ks.unlocked[a.Address] = u
return nil
}

// Find resolves the given account into a unique entry in the keystore.
func (ks *KeyStore) Find(a accounts.Account) (accounts.Account, error) {
ks.cache.maybeReload()
ks.cache.mu.Lock()
a, err := ks.cache.find(a)
ks.cache.mu.Unlock()
return a, err
}

func (ks *KeyStore) getDecryptedKey(a accounts.Account, auth string) (accounts.Account, *Key,
error) {
a, err := ks.Find(a)
if err != nil {
return a, nil, err
}
key, err := ks.storage.GetKey(a.Address, a.URL.Path, auth)
return a, key, err
}

func (ks *KeyStore) expire(addr common.Address, u *unlocked, timeout time.Duration) {
t := time.NewTimer(timeout)
defer t.Stop()
select {
case <-u.abort:
// just quit
case <-t.C:
ks.mu.Lock()
// only drop if it's still the same key instance that dropLater
// was launched with. we can check that using pointer equality
// because the map stores a new pointer every time the key is
// unlocked.
if ks.unlocked[addr] == u {
zeroKey(u.PrivateKey)
```

```
delete(ks.unlocked, addr)
}
ks.mu.Unlock()
}
}

// NewAccount generates a new key and stores it into the key directory,
// encrypting it with the passphrase.
func (ks *KeyStore) NewAccount(passphrase string) (accounts.Account, error) {
_, account, err := storeNewKey(ks.storage, crand.Reader, passphrase)
if err != nil {
return accounts.Account{}, err
}
// Add the account to the cache immediately rather
// than waiting for file system notifications to pick it up.
ks.cache.add(account)
ks.refreshWallets()
return account, nil
}

// Export exports as a JSON key, encrypted with newPassphrase.
func (ks *KeyStore) Export(a accounts.Account, passphrase, newPassphrase string) (keyJSON
[]byte, err error) {
_, key, err := ks.getDecryptedKey(a, passphrase)
if err != nil {
return nil, err
}
var N, P int
if store, ok := ks.storage.(*keyStorePassphrase); ok {
N, P = store.scryptN, store.scryptP
} else {
N, P = StandardScryptN, StandardScryptP
}
return EncryptKey(key, newPassphrase, N, P)
}

// Import stores the given encrypted JSON key into the key directory.
func (ks *KeyStore) Import(keyJSON []byte, passphrase, newPassphrase string)
(accounts.Account, error) {
key, err := DecryptKey(keyJSON, passphrase)
if key != nil && key.PrivateKey != nil {
defer zeroKey(key.PrivateKey)
```

```
}
if err != nil {
return accounts.Account{}, err
}
return ks.importKey(key, newPassphrase)
}

// ImportECDSA stores the given key into the key directory, encrypting it with the passphrase.
func (ks *KeyStore) ImportECDSA(priv *ecdsa.PrivateKey, passphrase string) (accounts.Account,
error) {
key := newKeyFromECDSA(priv)
if ks.cache.hasAddress(key.Address) {
return accounts.Account{}, fmt.Errorf("account already exists")
}
return ks.importKey(key, passphrase)
}

func (ks *KeyStore) importKey(key *Key, passphrase string) (accounts.Account, error) {
a := accounts.Account{Address: key.Address, URL: accounts.URL{Scheme: KeyStoreScheme,
Path: ks.storage.JoinPath(keyFileName(key.Address))}}
if err := ks.storage.StoreKey(a.URL.Path, key, passphrase); err != nil {
return accounts.Account{}, err
}
ks.cache.add(a)
ks.refreshWallets()
return a, nil
}

// Update changes the passphrase of an existing account.
func (ks *KeyStore) Update(a accounts.Account, passphrase, newPassphrase string) error {
a, key, err := ks.getDecryptedKey(a, passphrase)
if err != nil {
return err
}
return ks.storage.StoreKey(a.URL.Path, key, newPassphrase)
}

// ImportPreSaleKey decrypts the given Ethereum presale wallet and stores
// a key file in the key directory. The key file is encrypted with the same passphrase.
func (ks *KeyStore) ImportPreSaleKey(keyJSON []byte, passphrase string) (accounts.Account,
error) {
a, _, err := importPreSaleKey(ks.storage, keyJSON, passphrase)
```

```
if err != nil {
return a, err
}
ks.cache.add(a)
ks.refreshWallets()
return a, nil
}

// zeroKey zeroes a private key in memory.
func zeroKey(k *ecdsa.PrivateKey) {
b := k.D.Bits()
for i := range b {
b[i] = 0
}
}

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

/*

This key store behaves as KeyStorePlain with the difference that
the private key is encrypted and on disk uses another JSON encoding.

The crypto is documented at https://github.com/ethereum/wiki/wiki/Web3-Secret-Storage-Definition

*/

package keystore

import (
"bytes"
"crypto/aes"
"crypto/sha256"
"encoding/hex"
"encoding/json"
"fmt"
"io/ioutil"
"path/filepath"

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

```
"github.com/ethereum/go-ethereum/common/math"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/crypto/randentropy"
"github.com/pborman/uuid"
"golang.org/x/crypto/pbkdf2"
"golang.org/x/crypto/scrypt"
)

const (
keyHeaderKDF = "scrypt"

// StandardScryptN is the N parameter of Scrypt encryption algorithm, using 256MB
// memory and taking approximately 1s CPU time on a modern processor.
StandardScryptN = 1 << 18

// StandardScryptP is the P parameter of Scrypt encryption algorithm, using 256MB
// memory and taking approximately 1s CPU time on a modern processor.
StandardScryptP = 1

// LightScryptN is the N parameter of Scrypt encryption algorithm, using 4MB
// memory and taking approximately 100ms CPU time on a modern processor.
LightScryptN = 1 << 12

// LightScryptP is the P parameter of Scrypt encryption algorithm, using 4MB
// memory and taking approximately 100ms CPU time on a modern processor.
LightScryptP = 6

scryptR     = 8
scryptDKLen = 32
)

type keyStorePassphrase struct {
keysDirPath string
scryptN     int
scryptP     int
}

func (ks keyStorePassphrase) GetKey(addr common.Address, filename, auth string) (*Key, error) {
// Load the key from the keystore and decrypt its contents
keyjson, err := ioutil.ReadFile(filename)
if err != nil {
return nil, err
```

```
}
key, err := DecryptKey(keyjson, auth)
if err != nil {
return nil, err
}
// Make sure we're really operating on the requested key (no swap attacks)
if key.Address != addr {
return nil, fmt.Errorf("key content mismatch: have account %x, want %x", key.Address, addr)
}
return key, nil
}

func (ks keyStorePassphrase) StoreKey(filename string, key *Key, auth string) error {
keyjson, err := EncryptKey(key, auth, ks.scryptN, ks.scryptP)
if err != nil {
return err
}
return writeKeyFile(filename, keyjson)
}

func (ks keyStorePassphrase) JoinPath(filename string) string {
if filepath.IsAbs(filename) {
return filename
} else {
return filepath.Join(ks.keysDirPath, filename)
}
}

// EncryptKey encrypts a key using the specified scrypt parameters into a json
// blob that can be decrypted later on.
func EncryptKey(key *Key, auth string, scryptN, scryptP int) ([]byte, error) {
authArray := []byte(auth)
salt := randentropy.GetEntropyCSPRNG(32)
derivedKey, err := scrypt.Key(authArray, salt, scryptN, scryptR, scryptP, scryptDKLen)
if err != nil {
return nil, err
}
encryptKey := derivedKey[:16]
keyBytes := math.PaddedBigBytes(key.PrivateKey.D, 32)

iv := randentropy.GetEntropyCSPRNG(aes.BlockSize) // 16
cipherText, err := aesCTRXOR(encryptKey, keyBytes, iv)
```

```
if err != nil {
return nil, err
}
mac := crypto.Keccak256(derivedKey[16:32], cipherText)

scryptParamsJSON := make(map[string]interface{}, 5)
scryptParamsJSON["n"] = scryptN
scryptParamsJSON["r"] = scryptR
scryptParamsJSON["p"] = scryptP
scryptParamsJSON["dklen"] = scryptDKLen
scryptParamsJSON["salt"] = hex.EncodeToString(salt)

cipherParamsJSON := cipherparamsJSON{
IV: hex.EncodeToString(iv),
}

cryptoStruct := cryptoJSON{
Cipher:       "aes-128-ctr",
CipherText:   hex.EncodeToString(cipherText),
CipherParams: cipherParamsJSON,
KDF:          "scrypt",
KDFParams:    scryptParamsJSON,
MAC:          hex.EncodeToString(mac),
}
encryptedKeyJSONV3 := encryptedKeyJSONV3{
hex.EncodeToString(key.Address[:]),
cryptoStruct,
key.Id.String(),
version,
}
return json.Marshal(encryptedKeyJSONV3)
}

// DecryptKey decrypts a key from a json blob, returning the private key itself.
func DecryptKey(keyjson []byte, auth string) (*Key, error) {
// Parse the json into a simple map to fetch the key version
m := make(map[string]interface{})
if err := json.Unmarshal(keyjson, &m); err != nil {
return nil, err
}
// Depending on the version try to parse one way or another
var (
```

```
keyBytes, keyId []byte
err            error
)
if version, ok := m["version"].(string); ok && version == "1" {
k := new(encryptedKeyJSONV1)
if err := json.Unmarshal(keyjson, k); err != nil {
return nil, err
}
keyBytes, keyId, err = decryptKeyV1(k, auth)
} else {
k := new(encryptedKeyJSONV3)
if err := json.Unmarshal(keyjson, k); err != nil {
return nil, err
}
keyBytes, keyId, err = decryptKeyV3(k, auth)
}
// Handle any decryption errors and return the key
if err != nil {
return nil, err
}
key := crypto.ToECDSAUnsafe(keyBytes)

return &Key{
Id:         uuid.UUID(keyId),
Address:    crypto.PubkeyToAddress(key.PublicKey),
PrivateKey: key,
}, nil
}

func decryptKeyV3(keyProtected *encryptedKeyJSONV3, auth string) (keyBytes []byte, keyId
[]byte, err error) {
if keyProtected.Version != version {
return nil, nil, fmt.Errorf("Version not supported: %v", keyProtected.Version)
}

if keyProtected.Crypto.Cipher != "aes-128-ctr" {
return nil, nil, fmt.Errorf("Cipher not supported: %v", keyProtected.Crypto.Cipher)
}

keyId = uuid.Parse(keyProtected.Id)
mac, err := hex.DecodeString(keyProtected.Crypto.MAC)
if err != nil {
```

```
return nil, nil, err
}

iv, err := hex.DecodeString(keyProtected.Crypto.CipherParams.IV)
if err != nil {
return nil, nil, err
}

cipherText, err := hex.DecodeString(keyProtected.Crypto.CipherText)
if err != nil {
return nil, nil, err
}

derivedKey, err := getKDFKey(keyProtected.Crypto, auth)
if err != nil {
return nil, nil, err
}

calculatedMAC := crypto.Keccak256(derivedKey[16:32], cipherText)
if !bytes.Equal(calculatedMAC, mac) {
return nil, nil, ErrDecrypt
}

plainText, err := aesCTRXOR(derivedKey[:16], cipherText, iv)
if err != nil {
return nil, nil, err
}
return plainText, keyId, err
}

func decryptKeyV1(keyProtected *encryptedKeyJSONV1, auth string) (keyBytes []byte, keyId
[]byte, err error) {
keyId = uuid.Parse(keyProtected.Id)
mac, err := hex.DecodeString(keyProtected.Crypto.MAC)
if err != nil {
return nil, nil, err
}

iv, err := hex.DecodeString(keyProtected.Crypto.CipherParams.IV)
if err != nil {
return nil, nil, err
}
```

```
cipherText, err := hex.DecodeString(keyProtected.Crypto.CipherText)
if err != nil {
return nil, nil, err
}

derivedKey, err := getKDFKey(keyProtected.Crypto, auth)
if err != nil {
return nil, nil, err
}

calculatedMAC := crypto.Keccak256(derivedKey[16:32], cipherText)
if !bytes.Equal(calculatedMAC, mac) {
return nil, nil, ErrDecrypt
}

plainText, err := aesCBCDecrypt(crypto.Keccak256(derivedKey[:16])[:16], cipherText, iv)
if err != nil {
return nil, nil, err
}
return plainText, keyId, err
}

func getKDFKey(cryptoJSON cryptoJSON, auth string) ([]byte, error) {
authArray := []byte(auth)
salt, err := hex.DecodeString(cryptoJSON.KDFParams["salt"].(string))
if err != nil {
return nil, err
}
dkLen := ensureInt(cryptoJSON.KDFParams["dklen"])

if cryptoJSON.KDF == "scrypt" {
n := ensureInt(cryptoJSON.KDFParams["n"])
r := ensureInt(cryptoJSON.KDFParams["r"])
p := ensureInt(cryptoJSON.KDFParams["p"])
return scrypt.Key(authArray, salt, n, r, p, dkLen)

} else if cryptoJSON.KDF == "pbkdf2" {
c := ensureInt(cryptoJSON.KDFParams["c"])
prf := cryptoJSON.KDFParams["prf"].(string)
if prf != "hmac-sha256" {
return nil, fmt.Errorf("Unsupported PBKDF2 PRF: %s", prf)
```

```
}
key := pbkdf2.Key(authArray, salt, c, dkLen, sha256.New)
return key, nil
}

return nil, fmt.Errorf("Unsupported KDF: %s", cryptoJSON.KDF)
}

// TODO: can we do without this when unmarshalling dynamic JSON?
// why do integers in KDF params end up as float64 and not int after
// unmarshal?
func ensureInt(x interface{}) int {
res, ok := x.(int)
if !ok {
res = int(x.(float64))
}
return res
}

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

package keystore

import (
"io/ioutil"
"testing"

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

const (
veryLightScryptN = 2
veryLightScryptP = 1
)

// Tests that a json key file can be decrypted and encrypted in multiple rounds.
func TestKeyEncryptDecrypt(t *testing.T) {
keyjson, err := ioutil.ReadFile("testdata/very-light-scrypt.json")
if err != nil {
t.Fatal(err)
}
```

```
password := ""
address := common.HexToAddress("45dea0fb0bba44f4fcf290bba71fd57d7117cbb8")

// Do a few rounds of decryption and encryption
for i := 0; i < 3; i++ {
// Try a bad password first
if _, err := DecryptKey(keyjson, password+"bad"); err == nil {
t.Errorf("test %d: json key decrypted with bad password", i)
}
// Decrypt with the correct password
key, err := DecryptKey(keyjson, password)
if err != nil {
t.Fatalf("test %d: json key failed to decrypt: %v", i, err)
}
if key.Address != address {
t.Errorf("test %d: key address mismatch: have %x, want %x", i, key.Address, address)
}
// Recrypt with a new password and start over
password += "new data appended"
if keyjson, err = EncryptKey(key, password, veryLightScryptN, veryLightScryptP); err != nil {
t.Errorf("test %d: failed to recrypt key %v", i, err)
}
}
}

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

package keystore

import (
"encoding/json"
"fmt"
"os"
"path/filepath"

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

type keyStorePlain struct {
keysDirPath string
}
```

```
func (ks keyStorePlain) GetKey(addr common.Address, filename, auth string) (*Key, error) {
fd, err := os.Open(filename)
if err != nil {
return nil, err
}
defer fd.Close()
key := new(Key)
if err := json.NewDecoder(fd).Decode(key); err != nil {
return nil, err
}
if key.Address != addr {
return nil, fmt.Errorf("key content mismatch: have address %x, want %x", key.Address, addr)
}
return key, nil
}

func (ks keyStorePlain) StoreKey(filename string, key *Key, auth string) error {
content, err := json.Marshal(key)
if err != nil {
return err
}
return writeKeyFile(filename, content)
}

func (ks keyStorePlain) JoinPath(filename string) string {
if filepath.IsAbs(filename) {
return filename
} else {
return filepath.Join(ks.keysDirPath, filename)
}
}
```

37:F:\git\coin\ethereum\go-ethereum\accounts\keystore\keystore_plain_test.go

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

package keystore

import (
"crypto/rand"
"encoding/hex"
"fmt"
```

```
"io/ioutil"
"os"
"reflect"
"strings"
"testing"

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

func tmpKeyStoreIface(t *testing.T, encrypted bool) (dir string, ks keyStore) {
d, err := ioutil.TempDir("", "geth-keystore-test")
if err != nil {
t.Fatal(err)
}
if encrypted {
ks = &keyStorePassphrase{d, veryLightScryptN, veryLightScryptP}
} else {
ks = &keyStorePlain{d}
}
return d, ks
}

func TestKeyStorePlain(t *testing.T) {
dir, ks := tmpKeyStoreIface(t, false)
defer os.RemoveAll(dir)

pass := "" // not used but required by API
k1, account, err := storeNewKey(ks, rand.Reader, pass)
if err != nil {
t.Fatal(err)
}
k2, err := ks.GetKey(k1.Address, account.URL.Path, pass)
if err != nil {
t.Fatal(err)
}
if !reflect.DeepEqual(k1.Address, k2.Address) {
t.Fatal(err)
}
if !reflect.DeepEqual(k1.PrivateKey, k2.PrivateKey) {
t.Fatal(err)
}
```

```
}

func TestKeyStorePassphrase(t *testing.T) {
dir, ks := tmpKeyStoreIface(t, true)
defer os.RemoveAll(dir)

pass := "foo"
k1, account, err := storeNewKey(ks, rand.Reader, pass)
if err != nil {
t.Fatal(err)
}
k2, err := ks.GetKey(k1.Address, account.URL.Path, pass)
if err != nil {
t.Fatal(err)
}
if !reflect.DeepEqual(k1.Address, k2.Address) {
t.Fatal(err)
}
if !reflect.DeepEqual(k1.PrivateKey, k2.PrivateKey) {
t.Fatal(err)
}
}

func TestKeyStorePassphraseDecryptionFail(t *testing.T) {
dir, ks := tmpKeyStoreIface(t, true)
defer os.RemoveAll(dir)

pass := "foo"
k1, account, err := storeNewKey(ks, rand.Reader, pass)
if err != nil {
t.Fatal(err)
}
if _, err = ks.GetKey(k1.Address, account.URL.Path, "bar"); err != ErrDecrypt {
t.Fatalf("wrong error for invalid passphrase\ngot %q\nwant %q", err, ErrDecrypt)
}
}

func TestImportPreSaleKey(t *testing.T) {
dir, ks := tmpKeyStoreIface(t, true)
defer os.RemoveAll(dir)

// file content of a presale key file generated with:
```

```
// python pyethsaletool.py genwallet
// with password "foo"
fileContent := "{\"encseed\":
\"26d87f5f2bf9835f9a47eefae571bc09f9107bb13d54ff12a4ec095d01f83897494cf34f7bed2ed3412
6ecba9db7b62de56c9d7cd136520a0427bfb11b8954ba7ac39b90d4650d3448e31185affcd74226a
68f1e94b1108e6e0a4a91cdd83eba\", \"ethaddr\":
\"d4584b5f6229b7be90727b0fc8c6b91bb427821f\", \"email\": \"gustav.simonsson@gmail.com\",
\"btcaddr\": \"1EVknXyFC68kKNLkh6YnKzW41svSRoaAcx\"}"
pass := "foo"
account, _, err := importPreSaleKey(ks, []byte(fileContent), pass)
if err != nil {
t.Fatal(err)
}
if account.Address != common.HexToAddress("d4584b5f6229b7be90727b0fc8c6b91bb427821f")
{
t.Errorf("imported account has wrong address %x", account.Address)
}
if !strings.HasPrefix(account.URL.Path, dir) {
t.Errorf("imported account file not in keystore directory: %q", account.URL)
}
}

// Test and utils for the key store tests in the Ethereum JSON tests;
// testdataKeyStoreTests/basic_tests.json
type KeyStoreTestV3 struct {
Json     encryptedKeyJSONV3
Password string
Priv     string
}

type KeyStoreTestV1 struct {
Json     encryptedKeyJSONV1
Password string
Priv     string
}

func TestV3_PBKDF2_1(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("testdata/v3_test_vector.json", t)
testDecryptV3(tests["wikipage_test_vector_pbkdf2"], t)
}
```

```
func TestV3_PBKDF2_2(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("../../tests/files/KeyStoreTests/basic_tests.json", t)
testDecryptV3(tests["test1"], t)
}

func TestV3_PBKDF2_3(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("../../tests/files/KeyStoreTests/basic_tests.json", t)
testDecryptV3(tests["python_generated_test_with_odd_iv"], t)
}

func TestV3_PBKDF2_4(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("../../tests/files/KeyStoreTests/basic_tests.json", t)
testDecryptV3(tests["evilnonce"], t)
}

func TestV3_Scrypt_1(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("testdata/v3_test_vector.json", t)
testDecryptV3(tests["wikipage_test_vector_scrypt"], t)
}

func TestV3_Scrypt_2(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("../../tests/files/KeyStoreTests/basic_tests.json", t)
testDecryptV3(tests["test2"], t)
}

func TestV1_1(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV1("testdata/v1_test_vector.json", t)
testDecryptV1(tests["test1"], t)
}

func TestV1_2(t *testing.T) {
t.Parallel()
ks := &keyStorePassphrase{"testdata/v1", LightScryptN, LightScryptP}
addr := common.HexToAddress("cb61d5a9c4896fb9658090b597ef0e7be6f7b67e")
file :=
"testdata/v1/cb61d5a9c4896fb9658090b597ef0e7be6f7b67e/cb61d5a9c4896fb9658090b597ef0e
```

```
7be6f7b67e"
k, err := ks.GetKey(addr, file, "g")
if err != nil {
t.Fatal(err)
}
privHex := hex.EncodeToString(crypto.FromECDSA(k.PrivateKey))
expectedHex := "d1b1178d3529626a1a93e073f65028370d14c7eb0936eb42abef05db6f37ad7d"
if privHex != expectedHex {
t.Fatal(fmt.Errorf("Unexpected privkey: %v, expected %v", privHex, expectedHex))
}
}

func testDecryptV3(test KeyStoreTestV3, t *testing.T) {
privBytes, _, err := decryptKeyV3(&test.Json, test.Password)
if err != nil {
t.Fatal(err)
}
privHex := hex.EncodeToString(privBytes)
if test.Priv != privHex {
t.Fatal(fmt.Errorf("Decrypted bytes not equal to test, expected %v have %v", test.Priv, privHex))
}
}

func testDecryptV1(test KeyStoreTestV1, t *testing.T) {
privBytes, _, err := decryptKeyV1(&test.Json, test.Password)
if err != nil {
t.Fatal(err)
}
privHex := hex.EncodeToString(privBytes)
if test.Priv != privHex {
t.Fatal(fmt.Errorf("Decrypted bytes not equal to test, expected %v have %v", test.Priv, privHex))
}
}

func loadKeyStoreTestV3(file string, t *testing.T) map[string]KeyStoreTestV3 {
tests := make(map[string]KeyStoreTestV3)
err := common.LoadJSON(file, &tests)
if err != nil {
t.Fatal(err)
}
return tests
}
```

```
func loadKeyStoreTestV1(file string, t *testing.T) map[string]KeyStoreTestV1 {
tests := make(map[string]KeyStoreTestV1)
err := common.LoadJSON(file, &tests)
if err != nil {
t.Fatal(err)
}
return tests
}

func TestKeyForDirectICAP(t *testing.T) {
t.Parallel()
key := NewKeyForDirectICAP(rand.Reader)
if !strings.HasPrefix(key.Address.Hex(), "0x00") {
t.Errorf("Expected first address byte to be zero, have: %s", key.Address.Hex())
}
}

func TestV3_31_Byte_Key(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("testdata/v3_test_vector.json", t)
testDecryptV3(tests["31_byte_key"], t)
}

func TestV3_30_Byte_Key(t *testing.T) {
t.Parallel()
tests := loadKeyStoreTestV3("testdata/v3_test_vector.json", t)
testDecryptV3(tests["30_byte_key"], t)
}

38:F:\git\coin\ethereum\go-ethereum\accounts\keystore\keystore_test.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package keystore

import (
"io/ioutil"
"math/rand"
"os"
"runtime"
"sort"
"strings"
```

```
"testing"
"time"

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

var testSigData = make([]byte, 32)

func TestKeyStore(t *testing.T) {
dir, ks := tmpKeyStore(t, true)
defer os.RemoveAll(dir)

a, err := ks.NewAccount("foo")
if err != nil {
t.Fatal(err)
}
if !strings.HasPrefix(a.URL.Path, dir) {
t.Errorf("account file %s doesn't have dir prefix", a.URL)
}
stat, err := os.Stat(a.URL.Path)
if err != nil {
t.Fatalf("account file %s doesn't exist (%v)", a.URL, err)
}
if runtime.GOOS != "windows" && stat.Mode() != 0600 {
t.Fatalf("account file has wrong mode: got %o, want %o", stat.Mode(), 0600)
}
if !ks.HasAddress(a.Address) {
t.Errorf("HasAccount(%x) should've returned true", a.Address)
}
if err := ks.Update(a, "foo", "bar"); err != nil {
t.Errorf("Update error: %v", err)
}
if err := ks.Delete(a, "bar"); err != nil {
t.Errorf("Delete error: %v", err)
}
if common.FileExist(a.URL.Path) {
t.Errorf("account file %s should be gone after Delete", a.URL)
}
if ks.HasAddress(a.Address) {
t.Errorf("HasAccount(%x) should've returned true after Delete", a.Address)
```

```
}
}

func TestSign(t *testing.T) {
dir, ks := tmpKeyStore(t, true)
defer os.RemoveAll(dir)

pass := "" // not used but required by API
a1, err := ks.NewAccount(pass)
if err != nil {
t.Fatal(err)
}
if err := ks.Unlock(a1, ""); err != nil {
t.Fatal(err)
}
if _, err := ks.SignHash(accounts.Account{Address: a1.Address}, testSigData); err != nil {
t.Fatal(err)
}
}

func TestSignWithPassphrase(t *testing.T) {
dir, ks := tmpKeyStore(t, true)
defer os.RemoveAll(dir)

pass := "passwd"
acc, err := ks.NewAccount(pass)
if err != nil {
t.Fatal(err)
}

if _, unlocked := ks.unlocked[acc.Address]; unlocked {
t.Fatal("expected account to be locked")
}

_, err = ks.SignHashWithPassphrase(acc, pass, testSigData)
if err != nil {
t.Fatal(err)
}

if _, unlocked := ks.unlocked[acc.Address]; unlocked {
t.Fatal("expected account to be locked")
}
```

```
if _, err = ks.SignHashWithPassphrase(acc, "invalid passwd", testSigData); err == nil {
t.Fatal("expected SignHashWithPassphrase to fail with invalid password")
}
}

func TestTimedUnlock(t *testing.T) {
dir, ks := tmpKeyStore(t, true)
defer os.RemoveAll(dir)

pass := "foo"
a1, err := ks.NewAccount(pass)
if err != nil {
t.Fatal(err)
}

// Signing without passphrase fails because account is locked
_, err = ks.SignHash(accounts.Account{Address: a1.Address}, testSigData)
if err != ErrLocked {
t.Fatal("Signing should've failed with ErrLocked before unlocking, got ", err)
}

// Signing with passphrase works
if err = ks.TimedUnlock(a1, pass, 100*time.Millisecond); err != nil {
t.Fatal(err)
}

// Signing without passphrase works because account is temp unlocked
_, err = ks.SignHash(accounts.Account{Address: a1.Address}, testSigData)
if err != nil {
t.Fatal("Signing shouldn't return an error after unlocking, got ", err)
}

// Signing fails again after automatic locking
time.Sleep(250 * time.Millisecond)
_, err = ks.SignHash(accounts.Account{Address: a1.Address}, testSigData)
if err != ErrLocked {
t.Fatal("Signing should've failed with ErrLocked timeout expired, got ", err)
}
}

func TestOverrideUnlock(t *testing.T) {
```

```
dir, ks := tmpKeyStore(t, false)
defer os.RemoveAll(dir)

pass := "foo"
a1, err := ks.NewAccount(pass)
if err != nil {
t.Fatal(err)
}

// Unlock indefinitely.
if err = ks.TimedUnlock(a1, pass, 5*time.Minute); err != nil {
t.Fatal(err)
}

// Signing without passphrase works because account is temp unlocked
_, err = ks.SignHash(accounts.Account{Address: a1.Address}, testSigData)
if err != nil {
t.Fatal("Signing shouldn't return an error after unlocking, got ", err)
}

// reset unlock to a shorter period, invalidates the previous unlock
if err = ks.TimedUnlock(a1, pass, 100*time.Millisecond); err != nil {
t.Fatal(err)
}

// Signing without passphrase still works because account is temp unlocked
_, err = ks.SignHash(accounts.Account{Address: a1.Address}, testSigData)
if err != nil {
t.Fatal("Signing shouldn't return an error after unlocking, got ", err)
}

// Signing fails again after automatic locking
time.Sleep(250 * time.Millisecond)
_, err = ks.SignHash(accounts.Account{Address: a1.Address}, testSigData)
if err != ErrLocked {
t.Fatal("Signing should've failed with ErrLocked timeout expired, got ", err)
}
}

// This test should fail under -race if signing races the expiration goroutine.
func TestSignRace(t *testing.T) {
dir, ks := tmpKeyStore(t, false)
```

```
defer os.RemoveAll(dir)

// Create a test account.
a1, err := ks.NewAccount("")
if err != nil {
t.Fatal("could not create the test account", err)
}

if err := ks.TimedUnlock(a1, "", 15*time.Millisecond); err != nil {
t.Fatal("could not unlock the test account", err)
}
end := time.Now().Add(500 * time.Millisecond)
for time.Now().Before(end) {
if _, err := ks.SignHash(accounts.Account{Address: a1.Address}, testSigData); err == ErrLocked {
return
} else if err != nil {
t.Errorf("Sign error: %v", err)
return
}
time.Sleep(1 * time.Millisecond)
}
t.Errorf("Account did not lock within the timeout")
}

// Tests that the wallet notifier loop starts and stops correctly based on the
// addition and removal of wallet event subscriptions.
func TestWalletNotifierLifecycle(t *testing.T) {
// Create a temporary kesytore to test with
dir, ks := tmpKeyStore(t, false)
defer os.RemoveAll(dir)

// Ensure that the notification updater is not running yet
time.Sleep(250 * time.Millisecond)
ks.mu.RLock()
updating := ks.updating
ks.mu.RUnlock()

if updating {
t.Errorf("wallet notifier running without subscribers")
}
// Subscribe to the wallet feed and ensure the updater boots up
updates := make(chan accounts.WalletEvent)
```

```
subs := make([]event.Subscription, 2)
for i := 0; i < len(subs); i++ {
// Create a new subscription
subs[i] = ks.Subscribe(updates)

// Ensure the notifier comes online
time.Sleep(250 * time.Millisecond)
ks.mu.RLock()
updating = ks.updating
ks.mu.RUnlock()

if !updating {
t.Errorf("sub %d: wallet notifier not running after subscription", i)
}
}
// Unsubscribe and ensure the updater terminates eventually
for i := 0; i < len(subs); i++ {
// Close an existing subscription
subs[i].Unsubscribe()

// Ensure the notifier shuts down at and only at the last close
for k := 0; k < int(walletRefreshCycle/(250*time.Millisecond))+2; k++ {
ks.mu.RLock()
updating = ks.updating
ks.mu.RUnlock()

if i < len(subs)-1 && !updating {
t.Fatalf("sub %d: event notifier stopped prematurely", i)
}
if i == len(subs)-1 && !updating {
return
}
time.Sleep(250 * time.Millisecond)
}
}
t.Errorf("wallet notifier didn't terminate after unsubscribe")
}

// Tests that wallet notifications and correctly fired when accounts are added
// or deleted from the keystore.
func TestWalletNotifications(t *testing.T) {
```

```
// Create a temporary kesytore to test with
dir, ks := tmpKeyStore(t, false)
defer os.RemoveAll(dir)

// Subscribe to the wallet feed
updates := make(chan accounts.WalletEvent, 1)
sub := ks.Subscribe(updates)
defer sub.Unsubscribe()

// Randomly add and remove account and make sure events and wallets are in sync
live := make(map[common.Address]accounts.Account)
for i := 0; i < 1024; i++ {
// Execute a creation or deletion and ensure event arrival
if create := len(live) == 0 || rand.Int()%4 > 0; create {
// Add a new account and ensure wallet notifications arrives
account, err := ks.NewAccount("")
if err != nil {
t.Fatalf("failed to create test account: %v", err)
}
select {
case event := <-updates:
if !event.Arrive {
t.Errorf("departure event on account creation")
}
if event.Wallet.Accounts()[0] != account {
t.Errorf("account mismatch on created wallet: have %v, want %v", event.Wallet.Accounts()[0],
account)
}
default:
t.Errorf("wallet arrival event not fired on account creation")
}
live[account.Address] = account
} else {
// Select a random account to delete (crude, but works)
var account accounts.Account
for _, a := range live {
account = a
break
}
// Remove an account and ensure wallet notifiaction arrives
if err := ks.Delete(account, ""); err != nil {
t.Fatalf("failed to delete test account: %v", err)
```

```
}
select {
case event := <-updates:
if event.Arrive {
t.Errorf("arrival event on account deletion")
}
if event.Wallet.Accounts()[0] != account {
t.Errorf("account mismatch on deleted wallet: have %v, want %v", event.Wallet.Accounts()[0],
account)
}
default:
t.Errorf("wallet departure event not fired on account creation")
}
delete(live, account.Address)
}
// Retrieve the list of wallets and ensure it matches with our required live set
liveList := make([]accounts.Account, 0, len(live))
for _, account := range live {
liveList = append(liveList, account)
}
sort.Sort(accountsByURL(liveList))

wallets := ks.Wallets()
if len(liveList) != len(wallets) {
t.Errorf("wallet list doesn't match required accounts: have %v, want %v", wallets, liveList)
} else {
for j, wallet := range wallets {
if accs := wallet.Accounts(); len(accs) != 1 {
t.Errorf("wallet %d: contains invalid number of accounts: have %d, want 1", j, len(accs))
} else if accs[0] != liveList[j] {
t.Errorf("wallet %d: account mismatch: have %v, want %v", j, accs[0], liveList[j])
}
}
}
}
}

func tmpKeyStore(t *testing.T, encrypted bool) (string, *KeyStore) {
d, err := ioutil.TempDir("", "eth-keystore-test")
if err != nil {
t.Fatal(err)
}
```

```
new := NewPlaintextKeyStore
if encrypted {
new = func(kd string) *KeyStore { return NewKeyStore(kd, veryLightScryptN, veryLightScryptP) }
}
return d, new(d)
}

39:F:\git\coin\ethereum\go-ethereum\accounts\keystore\keystore_wallet.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package keystore

import (
"math/big"

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

// keystoreWallet implements the accounts.Wallet interface for the original
// keystore.
type keystoreWallet struct {
account  accounts.Account // Single account contained in this wallet
keystore *KeyStore        // Keystore where the account originates from
}

// URL implements accounts.Wallet, returning the URL of the account within.
func (w *keystoreWallet) URL() accounts.URL {
return w.account.URL
}

// Status implements accounts.Wallet, always returning "open", since there is no
// concept of open/close for plain keystore accounts.
func (w *keystoreWallet) Status() string {
w.keystore.mu.RLock()
defer w.keystore.mu.RUnlock()

if _, ok := w.keystore.unlocked[w.account.Address]; ok {
return "Unlocked"
}
return "Locked"
```

```
}

// Open implements accounts.Wallet, but is a noop for plain wallets since there
// is no connection or decryption step necessary to access the list of accounts.
func (w *keystoreWallet) Open(passphrase string) error { return nil }

// Close implements accounts.Wallet, but is a noop for plain wallets since is no
// meaningful open operation.
func (w *keystoreWallet) Close() error { return nil }

// Accounts implements accounts.Wallet, returning an account list consisting of
// a single account that the plain kestore wallet contains.
func (w *keystoreWallet) Accounts() []accounts.Account {
return []accounts.Account{w.account}
}

// Contains implements accounts.Wallet, returning whether a particular account is
// or is not wrapped by this wallet instance.
func (w *keystoreWallet) Contains(account accounts.Account) bool {
return account.Address == w.account.Address && (account.URL == (accounts.URL{}) ||
account.URL == w.account.URL)
}

// Derive implements accounts.Wallet, but is a noop for plain wallets since there
// is no notion of hierarchical account derivation for plain keystore accounts.
func (w *keystoreWallet) Derive(path accounts.DerivationPath, pin bool) (accounts.Account, error)
{
return accounts.Account{}, accounts.ErrNotSupported
}

// SelfDerive implements accounts.Wallet, but is a noop for plain wallets since
// there is no notion of hierarchical account derivation for plain keystore accounts.
func (w *keystoreWallet) SelfDerive(base accounts.DerivationPath, chain
ethereum.ChainStateReader) {}

// SignHash implements accounts.Wallet, attempting to sign the given hash with
// the given account. If the wallet does not wrap this particular account, an
// error is returned to avoid account leakage (even though in theory we may be
// able to sign via our shared keystore backend).
func (w *keystoreWallet) SignHash(account accounts.Account, hash []byte) ([]byte, error) {
// Make sure the requested account is contained within
if account.Address != w.account.Address {
```

```
return nil, accounts.ErrUnknownAccount
}
if account.URL != (accounts.URL{}) && account.URL != w.account.URL {
return nil, accounts.ErrUnknownAccount
}
// Account seems valid, request the keystore to sign
return w.keystore.SignHash(account, hash)
}

// SignTx implements accounts.Wallet, attempting to sign the given transaction
// with the given account. If the wallet does not wrap this particular account,
// an error is returned to avoid account leakage (even though in theory we may
// be able to sign via our shared keystore backend).
func (w *keystoreWallet) SignTx(account accounts.Account, tx *types.Transaction, chainID
*big.Int) (*types.Transaction, error) {
// Make sure the requested account is contained within
if account.Address != w.account.Address {
return nil, accounts.ErrUnknownAccount
}
if account.URL != (accounts.URL{}) && account.URL != w.account.URL {
return nil, accounts.ErrUnknownAccount
}
// Account seems valid, request the keystore to sign
return w.keystore.SignTx(account, tx, chainID)
}

// SignHashWithPassphrase implements accounts.Wallet, attempting to sign the
// given hash with the given account using passphrase as extra authentication.
func (w *keystoreWallet) SignHashWithPassphrase(account accounts.Account, passphrase string,
hash []byte) ([]byte, error) {
// Make sure the requested account is contained within
if account.Address != w.account.Address {
return nil, accounts.ErrUnknownAccount
}
if account.URL != (accounts.URL{}) && account.URL != w.account.URL {
return nil, accounts.ErrUnknownAccount
}
// Account seems valid, request the keystore to sign
return w.keystore.SignHashWithPassphrase(account, passphrase, hash)
}

// SignTxWithPassphrase implements accounts.Wallet, attempting to sign the given
```

```
// transaction with the given account using passphrase as extra authentication.
func (w *keystoreWallet) SignTxWithPassphrase(account accounts.Account, passphrase string, tx
*types.Transaction, chainID *big.Int) (*types.Transaction, error) {
// Make sure the requested account is contained within
if account.Address != w.account.Address {
return nil, accounts.ErrUnknownAccount
}
if account.URL != (accounts.URL{}) && account.URL != w.account.URL {
return nil, accounts.ErrUnknownAccount
}
// Account seems valid, request the keystore to sign
return w.keystore.SignTxWithPassphrase(account, passphrase, tx, chainID)
}

40:F:\git\coin\ethereum\go-ethereum\accounts\keystore\presale.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package keystore

import (
"crypto/aes"
"crypto/cipher"
"crypto/sha256"
"encoding/hex"
"encoding/json"
"errors"
"fmt"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/crypto"
"github.com/pborman/uuid"
"golang.org/x/crypto/pbkdf2"
)

// creates a Key and stores that in the given KeyStore by decrypting a presale key JSON
func importPreSaleKey(keyStore keyStore, keyJSON []byte, password string) (accounts.Account,
*Key, error) {
key, err := decryptPreSaleKey(keyJSON, password)
if err != nil {
return accounts.Account{}, nil, err
}
key.Id = uuid.NewRandom()
```

```
a := accounts.Account{Address: key.Address, URL: accounts.URL{Scheme: KeyStoreScheme,
Path: keyStore.JoinPath(keyFileName(key.Address))}}
err = keyStore.StoreKey(a.URL.Path, key, password)
return a, key, err
}

func decryptPreSaleKey(fileContent []byte, password string) (key *Key, err error) {
preSaleKeyStruct := struct {
EncSeed string
EthAddr string
Email   string
BtcAddr string
}{}
err = json.Unmarshal(fileContent, &preSaleKeyStruct)
if err != nil {
return nil, err
}
encSeedBytes, err := hex.DecodeString(preSaleKeyStruct.EncSeed)
if err != nil {
return nil, errors.New("invalid hex in encSeed")
}
iv := encSeedBytes[:16]
cipherText := encSeedBytes[16:]
/*
See https://github.com/ethereum/pyethsaletool

pyethsaletool generates the encryption key from password by
2000 rounds of PBKDF2 with HMAC-SHA-256 using password as salt (:().
16 byte key length within PBKDF2 and resulting key is used as AES key
*/
passBytes := []byte(password)
derivedKey := pbkdf2.Key(passBytes, passBytes, 2000, 16, sha256.New)
plainText, err := aesCBCDecrypt(derivedKey, cipherText, iv)
if err != nil {
return nil, err
}
ethPriv := crypto.Keccak256(plainText)
ecKey := crypto.ToECDSAUnsafe(ethPriv)

key = &Key{
Id:         nil,
Address:    crypto.PubkeyToAddress(ecKey.PublicKey),
```

```
PrivateKey: ecKey,
}
derivedAddr := hex.EncodeToString(key.Address.Bytes()) // needed because .Hex() gives leading
"0x"
expectedAddr := preSaleKeyStruct.EthAddr
if derivedAddr != expectedAddr {
err = fmt.Errorf("decrypted addr '%s' not equal to expected addr '%s'", derivedAddr, expectedAddr)
}
return key, err
}

func aesCTRXOR(key, inText, iv []byte) ([]byte, error) {
// AES-128 is selected due to size of encryptKey.
aesBlock, err := aes.NewCipher(key)
if err != nil {
return nil, err
}
stream := cipher.NewCTR(aesBlock, iv)
outText := make([]byte, len(inText))
stream.XORKeyStream(outText, inText)
return outText, err
}

func aesCBCDecrypt(key, cipherText, iv []byte) ([]byte, error) {
aesBlock, err := aes.NewCipher(key)
if err != nil {
return nil, err
}
decrypter := cipher.NewCBCDecrypter(aesBlock, iv)
paddedPlaintext := make([]byte, len(cipherText))
decrypter.CryptBlocks(paddedPlaintext, cipherText)
plaintext := pkcs7Unpad(paddedPlaintext)
if plaintext == nil {
return nil, ErrDecrypt
}
return plaintext, err
}

// From https://leanpub.com/gocrypto/read#leanpub-auto-block-cipher-modes
func pkcs7Unpad(in []byte) []byte {
if len(in) == 0 {
return nil
```

```
}

padding := in[len(in)-1]
if int(padding) > len(in) || padding > aes.BlockSize {
return nil
} else if padding == 0 {
return nil
}

for i := len(in) - 1; i > len(in)-int(padding)-1; i-- {
if in[i] != padding {
return nil
}
}
return in[:len(in)-int(padding)]
}

41:F:\git\coin\ethereum\go-ethereum\accounts\keystore\watch.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// +build darwin,!ios freebsd linux,!arm64 netbsd solaris

package keystore

import (
"time"

"github.com/ethereum/go-ethereum/log"
"github.com/rjeczalik/notify"
)

type watcher struct {
ac       *accountCache
starting bool
running  bool
ev       chan notify.EventInfo
quit     chan struct{}
}

func newWatcher(ac *accountCache) *watcher {
return &watcher{
ac:   ac,
```

```
ev:   make(chan notify.EventInfo, 10),
quit: make(chan struct{}),
}
}

// starts the watcher loop in the background.
// Start a watcher in the background if that's not already in progress.
// The caller must hold w.ac.mu.
func (w *watcher) start() {
if w.starting || w.running {
return
}
w.starting = true
go w.loop()
}

func (w *watcher) close() {
close(w.quit)
}

func (w *watcher) loop() {
defer func() {
w.ac.mu.Lock()
w.running = false
w.starting = false
w.ac.mu.Unlock()
}()
logger := log.New("path", w.ac.keydir)

if err := notify.Watch(w.ac.keydir, w.ev, notify.All); err != nil {
logger.Trace("Failed to watch keystore folder", "err", err)
return
}
defer notify.Stop(w.ev)

logger.Trace("Started watching keystore folder")
defer logger.Trace("Stopped watching keystore folder")

w.ac.mu.Lock()
w.running = true
w.ac.mu.Unlock()
```

```
// Wait for file system events and reload.
// When an event occurs, the reload call is delayed a bit so that
// multiple events arriving quickly only cause a single reload.
var (
debounce          = time.NewTimer(0)
debounceDuration  = 500 * time.Millisecond
inCycle, hadEvent bool
)
defer debounce.Stop()
for {
select {
case <-w.quit:
return
case <-w.ev:
if !inCycle {
debounce.Reset(debounceDuration)
inCycle = true
} else {
hadEvent = true
}
case <-debounce.C:
w.ac.mu.Lock()
w.ac.reload()
w.ac.mu.Unlock()
if hadEvent {
debounce.Reset(debounceDuration)
inCycle, hadEvent = true, false
} else {
inCycle, hadEvent = false, false
}
}
}
}
```

42:F:\git\coin\ethereum\go-ethereum\accounts\keystore\watch_fallback.go

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

// +build ios linux,arm64 windows !darwin,!freebsd,!linux,!netbsd,!solaris

// This is the fallback implementation of directory watching.
// It is used on unsupported platforms.
```

```
package keystore

type watcher struct{ running bool }

func newWatcher(*accountCache) *watcher { return new(watcher) }
func (*watcher) start()               {}
func (*watcher) close()                {}

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

package accounts

import (
"reflect"
"sort"
"sync"

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

// Manager is an overarching account manager that can communicate with various
// backends for signing transactions.
type Manager struct {
backends map[reflect.Type][]Backend // Index of backends currently registered
updaters []event.Subscription       // Wallet update subscriptions for all backends
updates  chan WalletEvent           // Subscription sink for backend wallet changes
wallets  []Wallet                   // Cache of all wallets from all registered backends

feed event.Feed // Wallet feed notifying of arrivals/departures

quit chan chan error
lock sync.RWMutex
}

// NewManager creates a generic account manager to sign transaction via various
// supported backends.
func NewManager(backends ...Backend) *Manager {
// Subscribe to wallet notifications from all backends
updates := make(chan WalletEvent, 4*len(backends))

subs := make([]event.Subscription, len(backends))
```

```
for i, backend := range backends {
subs[i] = backend.Subscribe(updates)
}
// Retrieve the initial list of wallets from the backends and sort by URL
var wallets []Wallet
for _, backend := range backends {
wallets = merge(wallets, backend.Wallets()...)
}
// Assemble the account manager and return
am := &Manager{
backends: make(map[reflect.Type][]Backend),
updaters: subs,
updates:  updates,
wallets:  wallets,
quit:     make(chan chan error),
}
for _, backend := range backends {
kind := reflect.TypeOf(backend)
am.backends[kind] = append(am.backends[kind], backend)
}
go am.update()

return am
}

// Close terminates the account manager's internal notification processes.
func (am *Manager) Close() error {
errc := make(chan error)
am.quit <- errc
return <-errc
}

// update is the wallet event loop listening for notifications from the backends
// and updating the cache of wallets.
func (am *Manager) update() {
// Close all subscriptions when the manager terminates
defer func() {
am.lock.Lock()
for _, sub := range am.updaters {
sub.Unsubscribe()
}
am.updaters = nil
```

```
am.lock.Unlock()
}()

// Loop until termination
for {
select {
case event := <-am.updates:
// Wallet event arrived, update local cache
am.lock.Lock()
if event.Arrive {
am.wallets = merge(am.wallets, event.Wallet)
} else {
am.wallets = drop(am.wallets, event.Wallet)
}
am.lock.Unlock()

// Notify any listeners of the event
am.feed.Send(event)

case errc := <-am.quit:
// Manager terminating, return
errc <- nil
return
}
}
}

// Backends retrieves the backend(s) with the given type from the account manager.
func (am *Manager) Backends(kind reflect.Type) []Backend {
return am.backends[kind]
}

// Wallets returns all signer accounts registered under this account manager.
func (am *Manager) Wallets() []Wallet {
am.lock.RLock()
defer am.lock.RUnlock()

cpy := make([]Wallet, len(am.wallets))
copy(cpy, am.wallets)
return cpy
}
```

```
// Wallet retrieves the wallet associated with a particular URL.
func (am *Manager) Wallet(url string) (Wallet, error) {
am.lock.RLock()
defer am.lock.RUnlock()

parsed, err := parseURL(url)
if err != nil {
return nil, err
}
for _, wallet := range am.Wallets() {
if wallet.URL() == parsed {
return wallet, nil
}
}
return nil, ErrUnknownWallet
}

// Find attempts to locate the wallet corresponding to a specific account. Since
// accounts can be dynamically added to and removed from wallets, this method has
// a linear runtime in the number of wallets.
func (am *Manager) Find(account Account) (Wallet, error) {
am.lock.RLock()
defer am.lock.RUnlock()

for _, wallet := range am.wallets {
if wallet.Contains(account) {
return wallet, nil
}
}
return nil, ErrUnknownAccount
}

// Subscribe creates an async subscription to receive notifications when the
// manager detects the arrival or departure of a wallet from any of its backends.
func (am *Manager) Subscribe(sink chan<- WalletEvent) event.Subscription {
return am.feed.Subscribe(sink)
}

// merge is a sorted analogue of append for wallets, where the ordering of the
// origin list is preserved by inserting new wallets at the correct position.
//
// The original slice is assumed to be already sorted by URL.
```

```
func merge(slice []Wallet, wallets ...Wallet) []Wallet {
for _, wallet := range wallets {
n := sort.Search(len(slice), func(i int) bool { return slice[i].URL().Cmp(wallet.URL()) >= 0 })
if n == len(slice) {
slice = append(slice, wallet)
continue
}
slice = append(slice[:n], append([]Wallet{wallet}, slice[n:]...)...)
}
return slice
}

// drop is the couterpart of merge, which looks up wallets from within the sorted
// cache and removes the ones specified.
func drop(slice []Wallet, wallets ...Wallet) []Wallet {
for _, wallet := range wallets {
n := sort.Search(len(slice), func(i int) bool { return slice[i].URL().Cmp(wallet.URL()) >= 0 })
if n == len(slice) {
// Wallet not found, may happen during startup
continue
}
slice = append(slice[:n], slice[n+1:]...)
}
return slice
}

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

package accounts

import (
"encoding/json"
"errors"
"fmt"
"strings"
)

// URL represents the canonical identification URL of a wallet or account.
//
// It is a simplified version of url.URL, with the important limitations (which
// are considered features here) that it contains value-copyable components only,
```

```
// as well as that it doesn't do any URL encoding/decoding of special characters.
//
// The former is important to allow an account to be copied without leaving live
// references to the original version, whereas the latter is important to ensure
// one single canonical form opposed to many allowed ones by the RFC 3986 spec.
//
// As such, these URLs should not be used outside of the scope of an Ethereum
// wallet or account.
type URL struct {
Scheme string // Protocol scheme to identify a capable account backend
Path   string // Path for the backend to identify a unique entity
}

// parseURL converts a user supplied URL into the accounts specific structure.
func parseURL(url string) (URL, error) {
parts := strings.Split(url, "://")
if len(parts) != 2 || parts[0] == "" {
return URL{}, errors.New("protocol scheme missing")
}
return URL{
Scheme: parts[0],
Path:   parts[1],
}, nil
}

// String implements the stringer interface.
func (u URL) String() string {
if u.Scheme != "" {
return fmt.Sprintf("%s://%s", u.Scheme, u.Path)
}
return u.Path
}

// TerminalString implements the log.TerminalStringer interface.
func (u URL) TerminalString() string {
url := u.String()
if len(url) > 32 {
return url[:31] + "…"
}
return url
}
```

```
// MarshalJSON implements the json.Marshaller interface.
func (u URL) MarshalJSON() ([]byte, error) {
return json.Marshal(u.String())
}

// Cmp compares x and y and returns:
//
//   -1 if x <  y
//    0 if x == y
//   +1 if x >  y
//
func (u URL) Cmp(url URL) int {
if u.Scheme == url.Scheme {
return strings.Compare(u.Path, url.Path)
}
return strings.Compare(u.Scheme, url.Scheme)
}

45:F:\git\coin\ethereum\go-ethereum\accounts\usbwallet\ledger_hub.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// This file contains the implementation for interacting with the Ledger hardware
// wallets. The wire protocol spec can be found in the Ledger Blue GitHub repo:
// https://raw.githubusercontent.com/LedgerHQ/blue-app-eth/master/doc/ethapp.asc

package usbwallet

import (
"errors"
"runtime"
"sync"
"time"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/log"
"github.com/karalabe/hid"
)

// LedgerScheme is the protocol scheme prefixing account and wallet URLs.
var LedgerScheme = "ledger"
```

```
// ledgerDeviceIDs are the known device IDs that Ledger wallets use.
var ledgerDeviceIDs = []deviceID{
{Vendor: 0x2c97, Product: 0x0000}, // Ledger Blue
{Vendor: 0x2c97, Product: 0x0001}, // Ledger Nano S
}

// Maximum time between wallet refreshes (if USB hotplug notifications don't work).
const ledgerRefreshCycle = time.Second

// Minimum time between wallet refreshes to avoid USB trashing.
const ledgerRefreshThrottling = 500 * time.Millisecond

// LedgerHub is a accounts.Backend that can find and handle Ledger hardware wallets.
type LedgerHub struct {
refreshed   time.Time               // Time instance when the list of wallets was last refreshed
wallets     []accounts.Wallet       // List of Ledger devices currently tracking
updateFeed  event.Feed              // Event feed to notify wallet additions/removals
updateScope event.SubscriptionScope // Subscription scope tracking current live listeners
updating    bool                    // Whether the event notification loop is running

quit chan chan error

stateLock sync.RWMutex // Protects the internals of the hub from racey access

// TODO(karalabe): remove if hotplug lands on Windows
commsPend int        // Number of operations blocking enumeration
commsLock sync.Mutex // Lock protecting the pending counter and enumeration
}

// NewLedgerHub creates a new hardware wallet manager for Ledger devices.
func NewLedgerHub() (*LedgerHub, error) {
if !hid.Supported() {
return nil, errors.New("unsupported platform")
}
hub := &LedgerHub{
quit: make(chan chan error),
}
hub.refreshWallets()
return hub, nil
}

// Wallets implements accounts.Backend, returning all the currently tracked USB
```

```
// devices that appear to be Ledger hardware wallets.
func (hub *LedgerHub) Wallets() []accounts.Wallet {
// Make sure the list of wallets is up to date
hub.refreshWallets()

hub.stateLock.RLock()
defer hub.stateLock.RUnlock()

cpy := make([]accounts.Wallet, len(hub.wallets))
copy(cpy, hub.wallets)
return cpy
}

// refreshWallets scans the USB devices attached to the machine and updates the
// list of wallets based on the found devices.
func (hub *LedgerHub) refreshWallets() {
// Don't scan the USB like crazy it the user fetches wallets in a loop
hub.stateLock.RLock()
elapsed := time.Since(hub.refreshed)
hub.stateLock.RUnlock()

if elapsed < ledgerRefreshThrottling {
return
}
// Retrieve the current list of Ledger devices
var ledgers []hid.DeviceInfo

if runtime.GOOS == "linux" {
// hidapi on Linux opens the device during enumeration to retrieve some infos,
// breaking the Ledger protocol if that is waiting for user confirmation. This
// is a bug acknowledged at Ledger, but it won't be fixed on old devices so we
// need to prevent concurrent comms ourselves. The more elegant solution would
// be to ditch enumeration in favor of hutplug events, but that don't work yet
// on Windows so if we need to hack it anyway, this is more elegant for now.
hub.commsLock.Lock()
if hub.commsPend > 0 { // A confirmation is pending, don't refresh
hub.commsLock.Unlock()
return
}
}
for _, info := range hid.Enumerate(0, 0) { // Can't enumerate directly, one valid ID is the 0 wildcard
for _, id := range ledgerDeviceIDs {
```

```
if info.VendorID == id.Vendor && info.ProductID == id.Product {
ledgers = append(ledgers, info)
break
}
}
}
if runtime.GOOS == "linux" {
// See rationale before the enumeration why this is needed and only on Linux.
hub.commsLock.Unlock()
}
// Transform the current list of wallets into the new one
hub.stateLock.Lock()

wallets := make([]accounts.Wallet, 0, len(ledgers))
events := []accounts.WalletEvent{}

for _, ledger := range ledgers {
url := accounts.URL{Scheme: LedgerScheme, Path: ledger.Path}

// Drop wallets in front of the next device or those that failed for some reason
for len(hub.wallets) > 0 && (hub.wallets[0].URL().Cmp(url) < 0 ||
hub.wallets[0].(*ledgerWallet).failed()) {
events = append(events, accounts.WalletEvent{Wallet: hub.wallets[0], Arrive: false})
hub.wallets = hub.wallets[1:]
}
// If there are no more wallets or the device is before the next, wrap new wallet
if len(hub.wallets) == 0 || hub.wallets[0].URL().Cmp(url) > 0 {
wallet := &ledgerWallet{hub: hub, url: &url, info: ledger, log: log.New("url", url)}

events = append(events, accounts.WalletEvent{Wallet: wallet, Arrive: true})
wallets = append(wallets, wallet)
continue
}
// If the device is the same as the first wallet, keep it
if hub.wallets[0].URL().Cmp(url) == 0 {
wallets = append(wallets, hub.wallets[0])
hub.wallets = hub.wallets[1:]
continue
}
}
// Drop any leftover wallets and set the new batch
for _, wallet := range hub.wallets {
```

```
events = append(events, accounts.WalletEvent{Wallet: wallet, Arrive: false})
}
hub.refreshed = time.Now()
hub.wallets = wallets
hub.stateLock.Unlock()

// Fire all wallet events and return
for _, event := range events {
hub.updateFeed.Send(event)
}
}

// Subscribe implements accounts.Backend, creating an async subscription to
// receive notifications on the addition or removal of Ledger wallets.
func (hub *LedgerHub) Subscribe(sink chan<- accounts.WalletEvent) event.Subscription {
// We need the mutex to reliably start/stop the update loop
hub.stateLock.Lock()
defer hub.stateLock.Unlock()

// Subscribe the caller and track the subscriber count
sub := hub.updateScope.Track(hub.updateFeed.Subscribe(sink))

// Subscribers require an active notification loop, start it
if !hub.updating {
hub.updating = true
go hub.updater()
}
return sub
}

// updater is responsible for maintaining an up-to-date list of wallets stored in
// the keystore, and for firing wallet addition/removal events. It listens for
// account change events from the underlying account cache, and also periodically
// forces a manual refresh (only triggers for systems where the filesystem notifier
// is not running).
func (hub *LedgerHub) updater() {
for {
// Wait for a USB hotplug event (not supported yet) or a refresh timeout
select {
//case <-hub.changes: // reenable on hutplug implementation
case <-time.After(ledgerRefreshCycle):
}
```

```
// Run the wallet refresher
hub.refreshWallets()

// If all our subscribers left, stop the updater
hub.stateLock.Lock()
if hub.updateScope.Count() == 0 {
hub.updating = false
hub.stateLock.Unlock()
return
}
hub.stateLock.Unlock()
}
}

46:F:\git\coin\ethereum\go-ethereum\accounts\usbwallet\ledger_wallet.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

// This file contains the implementation for interacting with the Ledger hardware
// wallets. The wire protocol spec can be found in the Ledger Blue GitHub repo:
// https://raw.githubusercontent.com/LedgerHQ/blue-app-eth/master/doc/ethapp.asc

package usbwallet

import (
"context"
"encoding/binary"
"encoding/hex"
"errors"
"fmt"
"io"
"math/big"
"sync"
"time"

ethereum "github.com/ethereum/go-ethereum"
"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/common/hexutil"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/rlp"
"github.com/karalabe/hid"
```

```
)

// Maximum time between wallet health checks to detect USB unplugs.
const ledgerHeartbeatCycle = time.Second

// Minimum time to wait between self derivation attempts, even it the user is
// requesting accounts like crazy.
const ledgerSelfDeriveThrottling = time.Second

// ledgerOpcode is an enumeration encoding the supported Ledger opcodes.
type ledgerOpcode byte

// ledgerParam1 is an enumeration encoding the supported Ledger parameters for
// specific opcodes. The same parameter values may be reused between opcodes.
type ledgerParam1 byte

// ledgerParam2 is an enumeration encoding the supported Ledger parameters for
// specific opcodes. The same parameter values may be reused between opcodes.
type ledgerParam2 byte

const (
ledgerOpRetrieveAddress  ledgerOpcode = 0x02 // Returns the public key and Ethereum address for a given BIP 32 path
ledgerOpSignTransaction  ledgerOpcode = 0x04 // Signs an Ethereum transaction after having the user validate the parameters
ledgerOpGetConfiguration ledgerOpcode = 0x06 // Returns specific wallet application configuration

ledgerP1DirectlyFetchAddress    ledgerParam1 = 0x00 // Return address directly from the wallet
ledgerP1ConfirmFetchAddress     ledgerParam1 = 0x01 // Require a user confirmation before returning the address
ledgerP1InitTransactionData     ledgerParam1 = 0x00 // First transaction data block for signing
ledgerP1ContTransactionData     ledgerParam1 = 0x80 // Subsequent transaction data block for signing
ledgerP2DiscardAddressChainCode ledgerParam2 = 0x00 // Do not return the chain code along with the address
ledgerP2ReturnAddressChainCode  ledgerParam2 = 0x01 // Require a user confirmation before returning the address
)

// errReplyInvalidHeader is the error message returned by a Ledger data exchange
// if the device replies with a mismatching header. This usually means the device
```

```
// is in browser mode.
var errReplyInvalidHeader = errors.New("invalid reply header")

// errInvalidVersionReply is the error message returned by a Ledger version retrieval
// when a response does arrive, but it does not contain the expected data.
var errInvalidVersionReply = errors.New("invalid version reply")

// ledgerWallet represents a live USB Ledger hardware wallet.
type ledgerWallet struct {
hub *LedgerHub    // USB hub the device originates from (TODO(karalabe): remove if hotplug
lands on Windows)
url *accounts.URL // Textual URL uniquely identifying this wallet

info    hid.DeviceInfo // Known USB device infos about the wallet
device  *hid.Device    // USB device advertising itself as a Ledger wallet
failure error          // Any failure that would make the device unusable

version  [3]byte                            // Current version of the Ledger Ethereum app (zero if app is
offline)
browser  bool                               // Flag whether the Ledger is in browser mode (reply
channel mismatch)
accounts []accounts.Account                 // List of derive accounts pinned on the Ledger
paths    map[common.Address]accounts.DerivationPath // Known derivation paths for signing
operations

deriveNextPath accounts.DerivationPath   // Next derivation path for account auto-discovery
deriveNextAddr common.Address            // Next derived account address for auto-discovery
deriveChain    ethereum.ChainStateReader // Blockchain state reader to discover used account
with
deriveReq      chan chan struct{}        // Channel to request a self-derivation on
deriveQuit     chan chan error           // Channel to terminate the self-deriver with

healthQuit chan chan error

// Locking a hardware wallet is a bit special. Since hardware devices are lower
// performing, any communication with them might take a non negligible amount of
// time. Worse still, waiting for user confirmation can take arbitrarily long,
// but exclusive communication must be upheld during. Locking the entire wallet
// in the mean time however would stall any parts of the system that don't want
// to communicate, just read some state (e.g. list the accounts).
//
// As such, a hardware wallet needs two locks to function correctly. A state
```

```
// lock can be used to protect the wallet's software-side internal state, which
// must not be held exlusively during hardware communication. A communication
// lock can be used to achieve exclusive access to the device itself, this one
// however should allow "skipping" waiting for operations that might want to
// use the device, but can live without too (e.g. account self-derivation).
//
// Since we have two locks, it's important to know how to properly use them:
//   - Communication requires the `device` to not change, so obtaining the
//     commsLock should be done after having a stateLock.
//   - Communication must not disable read access to the wallet state, so it
//     must only ever hold a *read* lock to stateLock.
commsLock chan struct{} // Mutex (buf=1) for the USB comms without keeping the state locked
stateLock sync.RWMutex  // Protects read and write access to the wallet struct fields

log log.Logger // Contextual logger to tag the ledger with its id
}

// URL implements accounts.Wallet, returning the URL of the Ledger device.
func (w *ledgerWallet) URL() accounts.URL {
return *w.url // Immutable, no need for a lock
}

// Status implements accounts.Wallet, always whether the Ledger is opened, closed
// or whether the Ethereum app was not started on it.
func (w *ledgerWallet) Status() string {
w.stateLock.RLock() // No device communication, state lock is enough
defer w.stateLock.RUnlock()

if w.failure != nil {
return fmt.Sprintf("Failed: %v", w.failure)
}
if w.device == nil {
return "Closed"
}
if w.browser {
return "Ethereum app in browser mode"
}
if w.offline() {
return "Ethereum app offline"
}
return fmt.Sprintf("Ethereum app v%d.%d.%d online", w.version[0], w.version[1], w.version[2])
}
```

```
// offline returns whether the wallet and the Ethereum app is offline or not.
//
// The method assumes that the state lock is held!
func (w *ledgerWallet) offline() bool {
return w.version == [3]byte{0, 0, 0}
}

// failed returns if the USB device wrapped by the wallet failed for some reason.
// This is used by the device scanner to report failed wallets as departed.
//
// The method assumes that the state lock is *not* held!
func (w *ledgerWallet) failed() bool {
w.stateLock.RLock() // No device communication, state lock is enough
defer w.stateLock.RUnlock()

return w.failure != nil
}

// Open implements accounts.Wallet, attempting to open a USB connection to the
// Ledger hardware wallet. The Ledger does not require a user passphrase, so that
// parameter is silently discarded.
func (w *ledgerWallet) Open(passphrase string) error {
w.stateLock.Lock() // State lock is enough since there's no connection yet at this point
defer w.stateLock.Unlock()

// If the wallet was already opened, don't try to open again
if w.device != nil {
return accounts.ErrWalletAlreadyOpen
}
// Otherwise iterate over all USB devices and find this again (no way to directly do this)
device, err := w.info.Open()
if err != nil {
return err
}
// Wallet seems to be successfully opened, guess if the Ethereum app is running
w.device = device
w.commsLock = make(chan struct{}, 1)
w.commsLock <- struct{}{} // Enable lock

w.paths = make(map[common.Address]accounts.DerivationPath)
```

```
w.deriveReq = make(chan chan struct{})
w.deriveQuit = make(chan chan error)
w.healthQuit = make(chan chan error)

defer func() {
go w.heartbeat()
go w.selfDerive()
}()

if _, err = w.ledgerDerive(accounts.DefaultBaseDerivationPath); err != nil {
// Ethereum app is not running or in browser mode, nothing more to do, return
if err == errReplyInvalidHeader {
w.browser = true
}
return nil
}
// Try to resolve the Ethereum app's version, will fail prior to v1.0.2
if w.version, err = w.ledgerVersion(); err != nil {
w.version = [3]byte{1, 0, 0} // Assume worst case, can't verify if v1.0.0 or v1.0.1
}
return nil
}

// heartbeat is a health check loop for the Ledger wallets to periodically verify
// whether they are still present or if they malfunctioned. It is needed because:
//  - libusb on Windows doesn't support hotplug, so we can't detect USB unplugs
//  - communication timeout on the Ledger requires a device power cycle to fix
func (w *ledgerWallet) heartbeat() {
w.log.Debug("Ledger health-check started")
defer w.log.Debug("Ledger health-check stopped")

// Execute heartbeat checks until termination or error
var (
errc chan error
err  error
)
for errc == nil && err == nil {
// Wait until termination is requested or the heartbeat cycle arrives
select {
case errc = <-w.healthQuit:
// Termination requested
continue
```

```
case <-time.After(ledgerHeartbeatCycle):
// Heartbeat time
}
// Execute a tiny data exchange to see responsiveness
w.stateLock.RLock()
if w.device == nil {
// Terminated while waiting for the lock
w.stateLock.RUnlock()
continue
}
<-w.commsLock // Don't lock state while resolving version
_, err = w.ledgerVersion()
w.commsLock <- struct{}{}
w.stateLock.RUnlock()

if err != nil && err != errInvalidVersionReply {
w.stateLock.Lock() // Lock state to tear the wallet down
w.failure = err
w.close()
w.stateLock.Unlock()
}
// Ignore non hardware related errors
err = nil
}
// In case of error, wait for termination
if err != nil {
w.log.Debug("Ledger health-check failed", "err", err)
errc = <-w.healthQuit
}
errc <- err
}

// Close implements accounts.Wallet, closing the USB connection to the Ledger.
func (w *ledgerWallet) Close() error {
// Ensure the wallet was opened
w.stateLock.RLock()
hQuit, dQuit := w.healthQuit, w.deriveQuit
w.stateLock.RUnlock()

// Terminate the health checks
var herr error
if hQuit != nil {
```

```
errc := make(chan error)
hQuit <- errc
herr = <-errc // Save for later, we *must* close the USB
}
// Terminate the self-derivations
var derr error
if dQuit != nil {
errc := make(chan error)
dQuit <- errc
derr = <-errc // Save for later, we *must* close the USB
}
// Terminate the device connection
w.stateLock.Lock()
defer w.stateLock.Unlock()

w.healthQuit = nil
w.deriveQuit = nil
w.deriveReq = nil

if err := w.close(); err != nil {
return err
}
if herr != nil {
return herr
}
return derr
}

// close is the internal wallet closer that terminates the USB connection and
// resets all the fields to their defaults.
//
// Note, close assumes the state lock is held!
func (w *ledgerWallet) close() error {
// Allow duplicate closes, especially for health-check failures
if w.device == nil {
return nil
}
// Close the device, clear everything, then return
w.device.Close()
w.device = nil

w.browser, w.version = false, [3]byte{}
```

```
w.accounts, w.paths = nil, nil

return nil
}

// Accounts implements accounts.Wallet, returning the list of accounts pinned to
// the Ledger hardware wallet. If self-derivation was enabled, the account list
// is periodically expanded based on current chain state.
func (w *ledgerWallet) Accounts() []accounts.Account {
// Attempt self-derivation if it's running
reqc := make(chan struct{}, 1)
select {
case w.deriveReq <- reqc:
// Self-derivation request accepted, wait for it
<-reqc
default:
// Self-derivation offline, throttled or busy, skip
}
// Return whatever account list we ended up with
w.stateLock.RLock()
defer w.stateLock.RUnlock()

cpy := make([]accounts.Account, len(w.accounts))
copy(cpy, w.accounts)
return cpy
}

// selfDerive is an account derivation loop that upon request attempts to find
// new non-zero accounts.
func (w *ledgerWallet) selfDerive() {
w.log.Debug("Ledger self-derivation started")
defer w.log.Debug("Ledger self-derivation stopped")

// Execute self-derivations until termination or error
var (
reqc chan struct{}
errc chan error
err  error
)
for errc == nil && err == nil {
// Wait until either derivation or termination is requested
select {
```

```
case errc = <-w.deriveQuit:
// Termination requested
continue
case reqc = <-w.deriveReq:
// Account discovery requested
}
// Derivation needs a chain and device access, skip if either unavailable
w.stateLock.RLock()
if w.device == nil || w.deriveChain == nil || w.offline() {
w.stateLock.RUnlock()
reqc <- struct{}{}
continue
}
select {
case <-w.commsLock:
default:
w.stateLock.RUnlock()
reqc <- struct{}{}
continue
}
// Device lock obtained, derive the next batch of accounts
var (
accs  []accounts.Account
paths []accounts.DerivationPath

nextAddr = w.deriveNextAddr
nextPath = w.deriveNextPath

context = context.Background()
)
for empty := false; !empty; {
// Retrieve the next derived Ethereum account
if nextAddr == (common.Address{}) {
if nextAddr, err = w.ledgerDerive(nextPath); err != nil {
w.log.Warn("Ledger account derivation failed", "err", err)
break
}
}
// Check the account's status against the current chain state
var (
balance *big.Int
nonce   uint64
```

```
)
balance, err = w.deriveChain.BalanceAt(context, nextAddr, nil)
if err != nil {
w.log.Warn("Ledger balance retrieval failed", "err", err)
break
}
nonce, err = w.deriveChain.NonceAt(context, nextAddr, nil)
if err != nil {
w.log.Warn("Ledger nonce retrieval failed", "err", err)
break
}
// If the next account is empty, stop self-derivation, but add it nonetheless
if balance.Sign() == 0 && nonce == 0 {
empty = true
}
// We've just self-derived a new account, start tracking it locally
path := make(accounts.DerivationPath, len(nextPath))
copy(path[:], nextPath[:])
paths = append(paths, path)

account := accounts.Account{
Address: nextAddr,
URL:     accounts.URL{Scheme: w.url.Scheme, Path: fmt.Sprintf("%s/%s", w.url.Path, path)},
}
accs = append(accs, account)

// Display a log message to the user for new (or previously empty accounts)
if _, known := w.paths[nextAddr]; !known || (!empty && nextAddr == w.deriveNextAddr) {
w.log.Info("Ledger discovered new account", "address", nextAddr, "path", path, "balance",
balance, "nonce", nonce)
}
// Fetch the next potential account
if !empty {
nextAddr = common.Address{}
nextPath[len(nextPath)-1]++
}
}
// Self derivation complete, release device lock
w.commsLock <- struct{}{}
w.stateLock.RUnlock()

// Insert any accounts successfully derived
```

```
w.stateLock.Lock()
for i := 0; i < len(accs); i++ {
if _, ok := w.paths[accs[i].Address]; !ok {
w.accounts = append(w.accounts, accs[i])
w.paths[accs[i].Address] = paths[i]
}
}
// Shift the self-derivation forward
// TODO(karalabe): don't overwrite changes from wallet.SelfDerive
w.deriveNextAddr = nextAddr
w.deriveNextPath = nextPath
w.stateLock.Unlock()

// Notify the user of termination and loop after a bit of time (to avoid trashing)
reqc <- struct{}{}
if err == nil {
select {
case errc = <-w.deriveQuit:
// Termination requested, abort
case <-time.After(ledgerSelfDeriveThrottling):
// Waited enough, willing to self-derive again
}
}
}
// In case of error, wait for termination
if err != nil {
w.log.Debug("Ledger self-derivation failed", "err", err)
errc = <-w.deriveQuit
}
errc <- err
}

// Contains implements accounts.Wallet, returning whether a particular account is
// or is not pinned into this Ledger instance. Although we could attempt to resolve
// unpinned accounts, that would be an non-negligible hardware operation.
func (w *ledgerWallet) Contains(account accounts.Account) bool {
w.stateLock.RLock()
defer w.stateLock.RUnlock()

_, exists := w.paths[account.Address]
return exists
}
```

```
// Derive implements accounts.Wallet, deriving a new account at the specific
// derivation path. If pin is set to true, the account will be added to the list
// of tracked accounts.
func (w *ledgerWallet) Derive(path accounts.DerivationPath, pin bool) (accounts.Account, error) {
// Try to derive the actual account and update its URL if successful
w.stateLock.RLock() // Avoid device disappearing during derivation

if w.device == nil || w.offline() {
w.stateLock.RUnlock()
return accounts.Account{}, accounts.ErrWalletClosed
}
<-w.commsLock // Avoid concurrent hardware access
address, err := w.ledgerDerive(path)
w.commsLock <- struct{}{}

w.stateLock.RUnlock()

// If an error occurred or no pinning was requested, return
if err != nil {
return accounts.Account{}, err
}
account := accounts.Account{
Address: address,
URL:     accounts.URL{Scheme: w.url.Scheme, Path: fmt.Sprintf("%s/%s", w.url.Path, path)},
}
if !pin {
return account, nil
}
// Pinning needs to modify the state
w.stateLock.Lock()
defer w.stateLock.Unlock()

if _, ok := w.paths[address]; !ok {
w.accounts = append(w.accounts, account)
w.paths[address] = path
}
return account, nil
}

// SelfDerive implements accounts.Wallet, trying to discover accounts that the
// user used previously (based on the chain state), but ones that he/she did not
```

```
// explicitly pin to the wallet manually. To avoid chain head monitoring, self
// derivation only runs during account listing (and even then throttled).
func (w *ledgerWallet) SelfDerive(base accounts.DerivationPath, chain
ethereum.ChainStateReader) {
w.stateLock.Lock()
defer w.stateLock.Unlock()

w.deriveNextPath = make(accounts.DerivationPath, len(base))
copy(w.deriveNextPath[:], base[:])

w.deriveNextAddr = common.Address{}
w.deriveChain = chain
}

// SignHash implements accounts.Wallet, however signing arbitrary data is not
// supported for Ledger wallets, so this method will always return an error.
func (w *ledgerWallet) SignHash(acc accounts.Account, hash []byte) ([]byte, error) {
return nil, accounts.ErrNotSupported
}

// SignTx implements accounts.Wallet. It sends the transaction over to the Ledger
// wallet to request a confirmation from the user. It returns either the signed
// transaction or a failure if the user denied the transaction.
//
// Note, if the version of the Ethereum application running on the Ledger wallet is
// too old to sign EIP-155 transactions, but such is requested nonetheless, an error
// will be returned opposed to silently signing in Homestead mode.
func (w *ledgerWallet) SignTx(account accounts.Account, tx *types.Transaction, chainID *big.Int)
(*types.Transaction, error) {
w.stateLock.RLock() // Comms have own mutex, this is for the state fields
defer w.stateLock.RUnlock()

// If the wallet is closed, or the Ethereum app doesn't run, abort
if w.device == nil || w.offline() {
return nil, accounts.ErrWalletClosed
}
// Make sure the requested account is contained within
path, ok := w.paths[account.Address]
if !ok {
return nil, accounts.ErrUnknownAccount
}
// Ensure the wallet is capable of signing the given transaction
```

```
if chainID != nil && w.version[0] <= 1 && w.version[1] <= 0 && w.version[2] <= 2 {
return nil, fmt.Errorf("Ledger v%d.%d.%d doesn't support signing this transaction, please update to
v1.0.3 at least", w.version[0], w.version[1], w.version[2])
}
// All infos gathered and metadata checks out, request signing
<-w.commsLock
defer func() { w.commsLock <- struct{}{} }()

// Ensure the device isn't screwed with while user confirmation is pending
// TODO(karalabe): remove if hotplug lands on Windows
w.hub.commsLock.Lock()
w.hub.commsPend++
w.hub.commsLock.Unlock()

defer func() {
w.hub.commsLock.Lock()
w.hub.commsPend--
w.hub.commsLock.Unlock()
}()
return w.ledgerSign(path, account.Address, tx, chainID)
}

// SignHashWithPassphrase implements accounts.Wallet, however signing arbitrary
// data is not supported for Ledger wallets, so this method will always return
// an error.
func (w *ledgerWallet) SignHashWithPassphrase(account accounts.Account, passphrase string,
hash []byte) ([]byte, error) {
return nil, accounts.ErrNotSupported
}

// SignTxWithPassphrase implements accounts.Wallet, attempting to sign the given
// transaction with the given account using passphrase as extra authentication.
// Since the Ledger does not support extra passphrases, it is silently ignored.
func (w *ledgerWallet) SignTxWithPassphrase(account accounts.Account, passphrase string, tx
*types.Transaction, chainID *big.Int) (*types.Transaction, error) {
return w.SignTx(account, tx, chainID)
}

// ledgerVersion retrieves the current version of the Ethereum wallet app running
// on the Ledger wallet.
//
// The version retrieval protocol is defined as follows:
```

```
//
//   CLA | INS | P1 | P2 | Lc | Le
//   ----+-----+----+----+----+---
//    E0 | 06  | 00 | 00 | 00 | 04
//
// With no input data, and the output data being:
//
//   Description                                        | Length
//   ---------------------------------------------------+--------
//   Flags 01: arbitrary data signature enabled by user | 1 byte
//   Application major version                          | 1 byte
//   Application minor version                          | 1 byte
//   Application patch version                          | 1 byte
func (w *ledgerWallet) ledgerVersion() ([3]byte, error) {
// Send the request and wait for the response
reply, err := w.ledgerExchange(ledgerOpGetConfiguration, 0, 0, nil)
if err != nil {
return [3]byte{}, err
}
if len(reply) != 4 {
return [3]byte{}, errInvalidVersionReply
}
// Cache the version for future reference
var version [3]byte
copy(version[:], reply[1:])
return version, nil
}

// ledgerDerive retrieves the currently active Ethereum address from a Ledger
// wallet at the specified derivation path.
//
// The address derivation protocol is defined as follows:
//
//   CLA | INS | P1 | P2 | Lc  | Le
//   ----+-----+----+----+-----+---
//    E0 | 02  | 00 return address
//               01 display address and confirm before returning
//                  | 00: do not return the chain code
//                  | 01: return the chain code
//                       | var | 00
//
// Where the input data is:
```

```
//
//   Description                                     | Length
//   ------------------------------------------------+--------
//   Number of BIP 32 derivations to perform (max 10) | 1 byte
//   First derivation index (big endian)              | 4 bytes
//   ...                                              | 4 bytes
//   Last derivation index (big endian)               | 4 bytes
//
// And the output data is:
//
//   Description             | Length
//   ------------------------+-------------------
//   Public Key length       | 1 byte
//   Uncompressed Public Key | arbitrary
//   Ethereum address length | 1 byte
//   Ethereum address        | 40 bytes hex ascii
//   Chain code if requested | 32 bytes
func (w *ledgerWallet) ledgerDerive(derivationPath []uint32) (common.Address, error) {
// Flatten the derivation path into the Ledger request
path := make([]byte, 1+4*len(derivationPath))
path[0] = byte(len(derivationPath))
for i, component := range derivationPath {
binary.BigEndian.PutUint32(path[1+4*i:], component)
}
// Send the request and wait for the response
reply, err := w.ledgerExchange(ledgerOpRetrieveAddress, ledgerP1DirectlyFetchAddress,
ledgerP2DiscardAddressChainCode, path)
if err != nil {
return common.Address{}, err
}
// Discard the public key, we don't need that for now
if len(reply) < 1 || len(reply) < 1+int(reply[0]) {
return common.Address{}, errors.New("reply lacks public key entry")
}
reply = reply[1+int(reply[0]):]

// Extract the Ethereum hex address string
if len(reply) < 1 || len(reply) < 1+int(reply[0]) {
return common.Address{}, errors.New("reply lacks address entry")
}
hexstr := reply[1 : 1+int(reply[0])]
```

```
// Decode the hex sting into an Ethereum address and return
var address common.Address
hex.Decode(address[:], hexstr)
return address, nil
}

// ledgerSign sends the transaction to the Ledger wallet, and waits for the user
// to confirm or deny the transaction.
//
// The transaction signing protocol is defined as follows:
//
//   CLA | INS | P1 | P2 | Lc  | Le
//   ----+-----+----+----+-----+---
//    E0 | 04  | 00: first transaction data block
//               80: subsequent transaction data block
//                 | 00 | variable | variable
//
// Where the input for the first transaction block (first 255 bytes) is:
//
//   Description                                        | Length
//   ---------------------------------------------------+----------
//   Number of BIP 32 derivations to perform (max 10) | 1 byte
//   First derivation index (big endian)              | 4 bytes
//   ...                                              | 4 bytes
//   Last derivation index (big endian)               | 4 bytes
//   RLP transaction chunk                            | arbitrary
//
// And the input for subsequent transaction blocks (first 255 bytes) are:
//
//   Description           | Length
//   ----------------------+----------
//   RLP transaction chunk | arbitrary
//
// And the output data is:
//
//   Description | Length
//   ------------+---------
//   signature V | 1 byte
//   signature R | 32 bytes
//   signature S | 32 bytes
func (w *ledgerWallet) ledgerSign(derivationPath []uint32, address common.Address, tx
*types.Transaction, chainID *big.Int) (*types.Transaction, error) {
```

```
// Flatten the derivation path into the Ledger request
path := make([]byte, 1+4*len(derivationPath))
path[0] = byte(len(derivationPath))
for i, component := range derivationPath {
binary.BigEndian.PutUint32(path[1+4*i:], component)
}
// Create the transaction RLP based on whether legacy or EIP155 signing was requeste
var (
txrlp []byte
err   error
)
if chainID == nil {
if txrlp, err = rlp.EncodeToBytes([]interface{}{tx.Nonce(), tx.GasPrice(), tx.Gas(), tx.To(), tx.Value(),
tx.Data()}); err != nil {
return nil, err
}
} else {
if txrlp, err = rlp.EncodeToBytes([]interface{}{tx.Nonce(), tx.GasPrice(), tx.Gas(), tx.To(), tx.Value(),
tx.Data(), chainID, big.NewInt(0), big.NewInt(0)}); err != nil {
return nil, err
}
}
payload := append(path, txrlp...)

// Send the request and wait for the response
var (
op    = ledgerP1InitTransactionData
reply []byte
)
for len(payload) > 0 {
// Calculate the size of the next data chunk
chunk := 255
if chunk > len(payload) {
chunk = len(payload)
}
// Send the chunk over, ensuring it's processed correctly
reply, err = w.ledgerExchange(ledgerOpSignTransaction, op, 0, payload[:chunk])
if err != nil {
return nil, err
}
// Shift the payload and ensure subsequent chunks are marked as such
payload = payload[chunk:]
```

```
op = ledgerP1ContTransactionData
}
// Extract the Ethereum signature and do a sanity validation
if len(reply) != 65 {
return nil, errors.New("reply lacks signature")
}
signature := append(reply[1:], reply[0])

// Create the correct signer and signature transform based on the chain ID
var signer types.Signer
if chainID == nil {
signer = new(types.HomesteadSigner)
} else {
signer = types.NewEIP155Signer(chainID)
signature[64] = signature[64] - byte(chainID.Uint64()*2+35)
}
// Inject the final signature into the transaction and sanity check the sender
signed, err := tx.WithSignature(signer, signature)
if err != nil {
return nil, err
}
sender, err := types.Sender(signer, signed)
if err != nil {
return nil, err
}
if sender != address {
return nil, fmt.Errorf("signer mismatch: expected %s, got %s", address.Hex(), sender.Hex())
}
return signed, nil
}

// ledgerExchange performs a data exchange with the Ledger wallet, sending it a
// message and retrieving the response.
//
// The common transport header is defined as follows:
//
//  Description                         | Length
//  ------------------------------------+----------
//  Communication channel ID (big endian) | 2 bytes
//  Command tag                           | 1 byte
//  Packet sequence index (big endian)    | 2 bytes
//  Payload                               | arbitrary
```

```
//
// The Communication channel ID allows commands multiplexing over the same
// physical link. It is not used for the time being, and should be set to 0101
// to avoid compatibility issues with implementations ignoring a leading 00 byte.
//
// The Command tag describes the message content. Use TAG_APDU (0x05) for standard
// APDU payloads, or TAG_PING (0x02) for a simple link test.
//
// The Packet sequence index describes the current sequence for fragmented payloads.
// The first fragment index is 0x00.
//
// APDU Command payloads are encoded as follows:
//
//  Description             | Length
//  ----------------------------------
//  APDU length (big endian) | 2 bytes
//  APDU CLA                 | 1 byte
//  APDU INS                 | 1 byte
//  APDU P1                  | 1 byte
//  APDU P2                  | 1 byte
//  APDU length              | 1 byte
//  Optional APDU data       | arbitrary
func (w *ledgerWallet) ledgerExchange(opcode ledgerOpcode, p1 ledgerParam1, p2
ledgerParam2, data []byte) ([]byte, error) {
// Construct the message payload, possibly split into multiple chunks
apdu := make([]byte, 2, 7+len(data))

binary.BigEndian.PutUint16(apdu, uint16(5+len(data)))
apdu = append(apdu, []byte{0xe0, byte(opcode), byte(p1), byte(p2), byte(len(data))}...)
apdu = append(apdu, data...)

// Stream all the chunks to the device
header := []byte{0x01, 0x01, 0x05, 0x00, 0x00} // Channel ID and command tag appended
chunk := make([]byte, 64)
space := len(chunk) - len(header)

for i := 0; len(apdu) > 0; i++ {
// Construct the new message to stream
chunk = append(chunk[:0], header...)
binary.BigEndian.PutUint16(chunk[3:], uint16(i))

if len(apdu) > space {
```

```
chunk = append(chunk, apdu[:space]...)
apdu = apdu[space:]
} else {
chunk = append(chunk, apdu...)
apdu = nil
}
// Send over to the device
w.log.Trace("Data chunk sent to the Ledger", "chunk", hexutil.Bytes(chunk))
if _, err := w.device.Write(chunk); err != nil {
return nil, err
}
}
// Stream the reply back from the wallet in 64 byte chunks
var reply []byte
chunk = chunk[:64] // Yeah, we surely have enough space
for {
// Read the next chunk from the Ledger wallet
if _, err := io.ReadFull(w.device, chunk); err != nil {
return nil, err
}
w.log.Trace("Data chunk received from the Ledger", "chunk", hexutil.Bytes(chunk))

// Make sure the transport header matches
if chunk[0] != 0x01 || chunk[1] != 0x01 || chunk[2] != 0x05 {
return nil, errReplyInvalidHeader
}
// If it's the first chunk, retrieve the total message length
var payload []byte

if chunk[3] == 0x00 && chunk[4] == 0x00 {
reply = make([]byte, 0, int(binary.BigEndian.Uint16(chunk[5:7])))
payload = chunk[7:]
} else {
payload = chunk[5:]
}
// Append to the reply and stop when filled up
if left := cap(reply) - len(reply); left > len(payload) {
reply = append(reply, payload...)
} else {
reply = append(reply, payload[:left]...)
break
}
```

```
}
return reply[:len(reply)-2], nil
}

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

// Package usbwallet implements support for USB hardware wallets.
package usbwallet

// deviceID is a combined vendor/product identifier to uniquely identify a USB
// hardware device.
type deviceID struct {
Vendor  uint16 // The Vendor identifer
Product uint16 // The Product identifier
}

48:F:\git\coin\ethereum\go-ethereum\cmd\abigen\main.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"encoding/json"
"flag"
"fmt"
"io/ioutil"
"os"
"strings"

"github.com/ethereum/go-ethereum/accounts/abi/bind"
"github.com/ethereum/go-ethereum/common/compiler"
)

var (
abiFlag = flag.String("abi", "", "Path to the Ethereum contract ABI json to bind")
binFlag = flag.String("bin", "", "Path to the Ethereum contract bytecode (generate deploy method)")
typFlag = flag.String("type", "", "Struct name for the binding (default = package name)")

solFlag  = flag.String("sol", "", "Path to the Ethereum contract Solidity source to build and bind")
solcFlag = flag.String("solc", "solc", "Solidity compiler to use if source builds are requested")
excFlag  = flag.String("exc", "", "Comma separated types to exclude from binding")
```

```
pkgFlag  = flag.String("pkg", "", "Package name to generate the binding into")
outFlag  = flag.String("out", "", "Output file for the generated binding (default = stdout)")
langFlag = flag.String("lang", "go", "Destination language for the bindings (go, java, objc)")
)

func main() {
// Parse and ensure all needed inputs are specified
flag.Parse()

if *abiFlag == "" && *solFlag == "" {
fmt.Printf("No contract ABI (--abi) or Solidity source (--sol) specified\n")
os.Exit(-1)
} else if (*abiFlag != "" || *binFlag != "" || *typFlag != "") && *solFlag != "" {
fmt.Printf("Contract ABI (--abi), bytecode (--bin) and type (--type) flags are mutually exclusive with the Solidity source (--sol) flag\n")
os.Exit(-1)
}
if *pkgFlag == "" {
fmt.Printf("No destination package specified (--pkg)\n")
os.Exit(-1)
}
var lang bind.Lang
switch *langFlag {
case "go":
lang = bind.LangGo
case "java":
lang = bind.LangJava
case "objc":
lang = bind.LangObjC
default:
fmt.Printf("Unsupported destination language \"%s\" (--lang)\n", *langFlag)
os.Exit(-1)
}
// If the entire solidity code was specified, build and bind based on that
var (
abis  []string
bins  []string
types []string
)
if *solFlag != "" {
// Generate the list of types to exclude from binding
```

```
exclude := make(map[string]bool)
for _, kind := range strings.Split(*excFlag, ",") {
exclude[strings.ToLower(kind)] = true
}
contracts, err := compiler.CompileSolidity(*solcFlag, *solFlag)
if err != nil {
fmt.Printf("Failed to build Solidity contract: %v\n", err)
os.Exit(-1)
}
// Gather all non-excluded contract for binding
for name, contract := range contracts {
if exclude[strings.ToLower(name)] {
continue
}
abi, _ := json.Marshal(contract.Info.AbiDefinition) // Flatten the compiler parse
abis = append(abis, string(abi))
bins = append(bins, contract.Code)

nameParts := strings.Split(name, ":")
types = append(types, nameParts[len(nameParts)-1])
}
} else {
// Otherwise load up the ABI, optional bytecode and type name from the parameters
abi, err := ioutil.ReadFile(*abiFlag)
if err != nil {
fmt.Printf("Failed to read input ABI: %v\n", err)
os.Exit(-1)
}
abis = append(abis, string(abi))

bin := []byte{}
if *binFlag != "" {
if bin, err = ioutil.ReadFile(*binFlag); err != nil {
fmt.Printf("Failed to read input bytecode: %v\n", err)
os.Exit(-1)
}
}
bins = append(bins, string(bin))

kind := *typFlag
if kind == "" {
kind = *pkgFlag
```

```
}
types = append(types, kind)
}
// Generate the contract binding
code, err := bind.Bind(types, abis, bins, *pkgFlag, lang)
if err != nil {
fmt.Printf("Failed to generate ABI binding: %v\n", err)
os.Exit(-1)
}
// Either flush it out to a file or display on the standard output
if *outFlag == "" {
fmt.Printf("%s\n", code)
return
}
if err := ioutil.WriteFile(*outFlag, []byte(code), 0600); err != nil {
fmt.Printf("Failed to write ABI binding: %v\n", err)
os.Exit(-1)
}
}
```

49:F:\git\coin\ethereum\go-ethereum\cmd\bootnode\main.go

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

// bootnode runs a bootstrap node for the Ethereum Discovery Protocol.
package main

import (
"crypto/ecdsa"
"flag"
"fmt"
"os"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/p2p/discover"
"github.com/ethereum/go-ethereum/p2p/discv5"
"github.com/ethereum/go-ethereum/p2p/nat"
"github.com/ethereum/go-ethereum/p2p/netutil"
)

func main() {
```

```
var (
listenAddr  = flag.String("addr", ":30301", "listen address")
genKey      = flag.String("genkey", "", "generate a node key")
writeAddr   = flag.Bool("writeaddress", false, "write out the node's pubkey hash and quit")
nodeKeyFile = flag.String("nodekey", "", "private key filename")
nodeKeyHex  = flag.String("nodekeyhex", "", "private key as hex (for testing)")
natdesc     = flag.String("nat", "none", "port mapping mechanism (any|none|upnp|pmp|extip:<IP>)")
netrestrict = flag.String("netrestrict", "", "restrict network communication to the given IP networks
(CIDR masks)")
runv5       = flag.Bool("v5", false, "run a v5 topic discovery bootnode")
verbosity   = flag.Int("verbosity", int(log.LvlInfo), "log verbosity (0-9)")
vmodule     = flag.String("vmodule", "", "log verbosity pattern")

nodeKey *ecdsa.PrivateKey
err     error
)
flag.Parse()

glogger := log.NewGlogHandler(log.StreamHandler(os.Stderr, log.TerminalFormat(false)))
glogger.Verbosity(log.Lvl(*verbosity))
glogger.Vmodule(*vmodule)
log.Root().SetHandler(glogger)

natm, err := nat.Parse(*natdesc)
if err != nil {
utils.Fatalf("-nat: %v", err)
}
switch {
case *genKey != "":
nodeKey, err = crypto.GenerateKey()
if err != nil {
utils.Fatalf("could not generate key: %v", err)
}
if err = crypto.SaveECDSA(*genKey, nodeKey); err != nil {
utils.Fatalf("%v", err)
}
return
case *nodeKeyFile == "" && *nodeKeyHex == "":
utils.Fatalf("Use -nodekey or -nodekeyhex to specify a private key")
case *nodeKeyFile != "" && *nodeKeyHex != "":
utils.Fatalf("Options -nodekey and -nodekeyhex are mutually exclusive")
case *nodeKeyFile != "":
```

```
if nodeKey, err = crypto.LoadECDSA(*nodeKeyFile); err != nil {
utils.Fatalf("-nodekey: %v", err)
}
case *nodeKeyHex != "":
if nodeKey, err = crypto.HexToECDSA(*nodeKeyHex); err != nil {
utils.Fatalf("-nodekeyhex: %v", err)
}
}

if *writeAddr {
fmt.Printf("%v\n", discover.PubkeyID(&nodeKey.PublicKey))
os.Exit(0)
}

var restrictList *netutil.Netlist
if *netrestrict != "" {
restrictList, err = netutil.ParseNetlist(*netrestrict)
if err != nil {
utils.Fatalf("-netrestrict: %v", err)
}
}

if *runv5 {
if _, err := discv5.ListenUDP(nodeKey, *listenAddr, natm, "", restrictList); err != nil {
utils.Fatalf("%v", err)
}
} else {
if _, err := discover.ListenUDP(nodeKey, *listenAddr, natm, "", restrictList); err != nil {
utils.Fatalf("%v", err)
}
}

select {}
}

50:F:\git\coin\ethereum\go-ethereum\cmd\evm\compiler.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"errors"
```

```
"fmt"
"io/ioutil"

"github.com/ethereum/go-ethereum/cmd/evm/internal/compiler"

cli "gopkg.in/urfave/cli.v1"
)

var compileCommand = cli.Command{
Action:    compileCmd,
Name:      "compile",
Usage:     "compiles easm source to evm binary",
ArgsUsage: "<file>",
}

func compileCmd(ctx *cli.Context) error {
debug := ctx.GlobalBool(DebugFlag.Name)

if len(ctx.Args().First()) == 0 {
return errors.New("filename required")
}

fn := ctx.Args().First()
src, err := ioutil.ReadFile(fn)
if err != nil {
return err
}

bin, err := compiler.Compile(fn, src, debug)
if err != nil {
return err
}
fmt.Println(bin)
return nil
}

51:F:\git\coin\ethereum\go-ethereum\cmd\evm\disasm.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
```

```
"errors"
"fmt"
"io/ioutil"
"strings"

"github.com/ethereum/go-ethereum/core/asm"
cli "gopkg.in/urfave/cli.v1"
)

var disasmCommand = cli.Command{
Action:    disasmCmd,
Name:      "disasm",
Usage:     "disassembles evm binary",
ArgsUsage: "<file>",
}

func disasmCmd(ctx *cli.Context) error {
if len(ctx.Args().First()) == 0 {
return errors.New("filename required")
}

fn := ctx.Args().First()
in, err := ioutil.ReadFile(fn)
if err != nil {
return err
}

code := strings.TrimSpace(string(in[:]))
fmt.Printf("%v\n", code)
if err = asm.PrintDisassembled(code); err != nil {
return err
}
return nil
}

52:F:\git\coin\ethereum\go-ethereum\cmd\evm\internal\compiler\compiler.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package compiler

import (
"errors"
```

```
"fmt"

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

func Compile(fn string, src []byte, debug bool) (string, error) {
compiler := asm.NewCompiler(debug)
compiler.Feed(asm.Lex(fn, src, debug))

bin, compileErrors := compiler.Compile()
if len(compileErrors) > 0 {
// report errors
for _, err := range compileErrors {
fmt.Printf("%s:%v\n", fn, err)
}
return "", errors.New("compiling failed")
}
return bin, nil
}

53:F:\git\coin\ethereum\go-ethereum\cmd\evm\json_logger.go
// along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"encoding/json"
"io"
"time"

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

type JSONLogger struct {
encoder *json.Encoder
cfg     *vm.LogConfig
}

func NewJSONLogger(cfg *vm.LogConfig, writer io.Writer) *JSONLogger {
return &JSONLogger{json.NewEncoder(writer), cfg}
```

```
}

// CaptureState outputs state information on the logger.
func (l *JSONLogger) CaptureState(env *vm.EVM, pc uint64, op vm.OpCode, gas, cost uint64,
memory *vm.Memory, stack *vm.Stack, contract *vm.Contract, depth int, err error) error {
log := vm.StructLog{
Pc:         pc,
Op:         op,
Gas:        gas + cost,
GasCost:    cost,
MemorySize: memory.Len(),
Storage:    nil,
Depth:      depth,
Err:        err,
}
if !l.cfg.DisableMemory {
log.Memory = memory.Data()
}
if !l.cfg.DisableStack {
log.Stack = stack.Data()
}
return l.encoder.Encode(log)
}

// CaptureEnd is triggered at end of execution.
func (l *JSONLogger) CaptureEnd(output []byte, gasUsed uint64, t time.Duration) error {
type endLog struct {
Output  string              `json:"output"`
GasUsed math.HexOrDecimal64 `json:"gasUsed"`
Time    time.Duration       `json:"time"`
}
return l.encoder.Encode(endLog{common.Bytes2Hex(output), math.HexOrDecimal64(gasUsed),
t})
}

54:F:\git\coin\ethereum\go-ethereum\cmd\evm\main.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// evm executes EVM code snippets.
package main

import (
```

```
"fmt"
"math/big"
"os"

"github.com/ethereum/go-ethereum/cmd/utils"
"gopkg.in/urfave/cli.v1"
)

var gitCommit = "" // Git SHA1 commit hash of the release (set via linker flags)

var (
app = utils.NewApp(gitCommit, "the evm command line interface")

DebugFlag = cli.BoolFlag{
Name:  "debug",
Usage: "output full trace logs",
}
MemProfileFlag = cli.StringFlag{
Name:  "memprofile",
Usage: "creates a memory profile at the given path",
}
CPUProfileFlag = cli.StringFlag{
Name:  "cpuprofile",
Usage: "creates a CPU profile at the given path",
}
StatDumpFlag = cli.BoolFlag{
Name:  "statdump",
Usage: "displays stack and heap memory information",
}
CodeFlag = cli.StringFlag{
Name:  "code",
Usage: "EVM code",
}
CodeFileFlag = cli.StringFlag{
Name:  "codefile",
Usage: "file containing EVM code",
}
GasFlag = cli.Uint64Flag{
Name:  "gas",
Usage: "gas limit for the evm",
Value: 10000000000,
}
```

```
PriceFlag = utils.BigFlag{
Name:  "price",
Usage: "price set for the evm",
Value: new(big.Int),
}
ValueFlag = utils.BigFlag{
Name:  "value",
Usage: "value set for the evm",
Value: new(big.Int),
}
DumpFlag = cli.BoolFlag{
Name:  "dump",
Usage: "dumps the state after the run",
}
InputFlag = cli.StringFlag{
Name:  "input",
Usage: "input for the EVM",
}
VerbosityFlag = cli.IntFlag{
Name:  "verbosity",
Usage: "sets the verbosity level",
}
CreateFlag = cli.BoolFlag{
Name:  "create",
Usage: "indicates the action should be create rather than call",
}
DisableGasMeteringFlag = cli.BoolFlag{
Name:  "nogasmetering",
Usage: "disable gas metering",
}
GenesisFlag = cli.StringFlag{
Name:  "prestate",
Usage: "JSON file with prestate (genesis) config",
}
MachineFlag = cli.BoolFlag{
Name:  "json",
Usage: "output trace logs in machine readable format (json)",
}
SenderFlag = cli.StringFlag{
Name:  "sender",
Usage: "The transaction origin",
}
```

```
DisableMemoryFlag = cli.BoolFlag{
Name:  "nomemory",
Usage: "disable memory output",
}
DisableStackFlag = cli.BoolFlag{
Name:  "nostack",
Usage: "disable stack output",
}
)

func init() {
app.Flags = []cli.Flag{
CreateFlag,
DebugFlag,
VerbosityFlag,
CodeFlag,
CodeFileFlag,
GasFlag,
PriceFlag,
ValueFlag,
DumpFlag,
InputFlag,
DisableGasMeteringFlag,
MemProfileFlag,
CPUProfileFlag,
StatDumpFlag,
GenesisFlag,
MachineFlag,
SenderFlag,
DisableMemoryFlag,
DisableStackFlag,
}
app.Commands = []cli.Command{
compileCommand,
disasmCommand,
runCommand,
}
}

func main() {
if err := app.Run(os.Args); err != nil {
fmt.Fprintln(os.Stderr, err)
```

```
os.Exit(1)
}
}
```

55:F:\git\coin\ethereum\go-ethereum\cmd\evm\runner.go

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

package main

import (
"bytes"
"encoding/json"
"fmt"
"io/ioutil"
"os"
"runtime/pprof"
"time"

goruntime "runtime"

"github.com/ethereum/go-ethereum/cmd/evm/internal/compiler"
"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/core/vm/runtime"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/params"
cli "gopkg.in/urfave/cli.v1"
)

var runCommand = cli.Command{
Action:      runCmd,
Name:        "run",
Usage:       "run arbitrary evm binary",
ArgsUsage:   "<code>",
Description: `The run command runs arbitrary EVM code.`,
}

// readGenesis will read the given JSON format genesis file and return
```

```
// the initialized Genesis structure
func readGenesis(genesisPath string) *core.Genesis {
// Make sure we have a valid genesis JSON
//genesisPath := ctx.Args().First()
if len(genesisPath) == 0 {
utils.Fatalf("Must supply path to genesis JSON file")
}
file, err := os.Open(genesisPath)
if err != nil {
utils.Fatalf("Failed to read genesis file: %v", err)
}
defer file.Close()

genesis := new(core.Genesis)
if err := json.NewDecoder(file).Decode(genesis); err != nil {
utils.Fatalf("invalid genesis file: %v", err)
}
return genesis
}

func runCmd(ctx *cli.Context) error {
glogger := log.NewGlogHandler(log.StreamHandler(os.Stderr, log.TerminalFormat(false)))
glogger.Verbosity(log.Lvl(ctx.GlobalInt(VerbosityFlag.Name)))
log.Root().SetHandler(glogger)
logconfig := &vm.LogConfig{
DisableMemory: ctx.GlobalBool(DisableMemoryFlag.Name),
DisableStack:  ctx.GlobalBool(DisableStackFlag.Name),
}

var (
tracer      vm.Tracer
debugLogger *vm.StructLogger
statedb     *state.StateDB
chainConfig *params.ChainConfig
sender      = common.StringToAddress("sender")
)
if ctx.GlobalBool(MachineFlag.Name) {
tracer = NewJSONLogger(logconfig, os.Stdout)
} else if ctx.GlobalBool(DebugFlag.Name) {
debugLogger = vm.NewStructLogger(logconfig)
tracer = debugLogger
} else {
```

```
debugLogger = vm.NewStructLogger(logconfig)
}
if ctx.GlobalString(GenesisFlag.Name) != "" {
gen := readGenesis(ctx.GlobalString(GenesisFlag.Name))
_, statedb = gen.ToBlock()
chainConfig = gen.Config
} else {
db, _ := ethdb.NewMemDatabase()
statedb, _ = state.New(common.Hash{}, state.NewDatabase(db))
}
if ctx.GlobalString(SenderFlag.Name) != "" {
sender = common.HexToAddress(ctx.GlobalString(SenderFlag.Name))
}

statedb.CreateAccount(sender)

var (
code []byte
ret  []byte
err  error
)
if fn := ctx.Args().First(); len(fn) > 0 {
src, err := ioutil.ReadFile(fn)
if err != nil {
return err
}

bin, err := compiler.Compile(fn, src, false)
if err != nil {
return err
}
code = common.Hex2Bytes(bin)
} else if ctx.GlobalString(CodeFlag.Name) != "" {
code = common.Hex2Bytes(ctx.GlobalString(CodeFlag.Name))
} else {
var hexcode []byte
if ctx.GlobalString(CodeFileFlag.Name) != "" {
var err error
hexcode, err = ioutil.ReadFile(ctx.GlobalString(CodeFileFlag.Name))
if err != nil {
fmt.Printf("Could not load code from file: %v\n", err)
os.Exit(1)
```

```
}
} else {
var err error
hexcode, err = ioutil.ReadAll(os.Stdin)
if err != nil {
fmt.Printf("Could not load code from stdin: %v\n", err)
os.Exit(1)
}
}
code = common.Hex2Bytes(string(bytes.TrimRight(hexcode, "\n")))
}
initialGas := ctx.GlobalUint64(GasFlag.Name)
runtimeConfig := runtime.Config{
Origin:   sender,
State:    statedb,
GasLimit: initialGas,
GasPrice: utils.GlobalBig(ctx, PriceFlag.Name),
Value:    utils.GlobalBig(ctx, ValueFlag.Name),
EVMConfig: vm.Config{
Tracer:             tracer,
Debug:              ctx.GlobalBool(DebugFlag.Name) || ctx.GlobalBool(MachineFlag.Name),
DisableGasMetering: ctx.GlobalBool(DisableGasMeteringFlag.Name),
},
}

if cpuProfilePath := ctx.GlobalString(CPUProfileFlag.Name); cpuProfilePath != "" {
f, err := os.Create(cpuProfilePath)
if err != nil {
fmt.Println("could not create CPU profile: ", err)
os.Exit(1)
}
if err := pprof.StartCPUProfile(f); err != nil {
fmt.Println("could not start CPU profile: ", err)
os.Exit(1)
}
defer pprof.StopCPUProfile()
}

if chainConfig != nil {
runtimeConfig.ChainConfig = chainConfig
}
tstart := time.Now()
```

```
var leftOverGas uint64
if ctx.GlobalBool(CreateFlag.Name) {
input := append(code, common.Hex2Bytes(ctx.GlobalString(InputFlag.Name))...)
ret, _, leftOverGas, err = runtime.Create(input, &runtimeConfig)
} else {
receiver := common.StringToAddress("receiver")
statedb.SetCode(receiver, code)

ret, leftOverGas, err = runtime.Call(receiver,
common.Hex2Bytes(ctx.GlobalString(InputFlag.Name)), &runtimeConfig)
}
execTime := time.Since(tstart)

if ctx.GlobalBool(DumpFlag.Name) {
statedb.IntermediateRoot(true)
fmt.Println(string(statedb.Dump()))
}

if memProfilePath := ctx.GlobalString(MemProfileFlag.Name); memProfilePath != "" {
f, err := os.Create(memProfilePath)
if err != nil {
fmt.Println("could not create memory profile: ", err)
os.Exit(1)
}
if err := pprof.WriteHeapProfile(f); err != nil {
fmt.Println("could not write memory profile: ", err)
os.Exit(1)
}
f.Close()
}

if ctx.GlobalBool(DebugFlag.Name) {
if debugLogger != nil {
fmt.Fprintln(os.Stderr, "#### TRACE ####")
vm.WriteTrace(os.Stderr, debugLogger.StructLogs())
}
fmt.Fprintln(os.Stderr, "#### LOGS ####")
vm.WriteLogs(os.Stderr, statedb.Logs())
}

if ctx.GlobalBool(StatDumpFlag.Name) {
var mem goruntime.MemStats
```

```
goruntime.ReadMemStats(&mem)
fmt.Fprintf(os.Stderr, `evm execution time: %v
heap objects:       %d
allocations:        %d
total allocations:  %d
GC calls:           %d
Gas used:           %d

`, execTime, mem.HeapObjects, mem.Alloc, mem.TotalAlloc, mem.NumGC, initialGas-leftOverGas)
}
if tracer != nil {
tracer.CaptureEnd(ret, initialGas-leftOverGas, execTime)
} else {
fmt.Printf("0x%x\n", ret)
}

if err != nil {
fmt.Printf(" error: %v\n", err)
}
return nil
}

56:F:\git\coin\ethereum\go-ethereum\cmd\faucet\faucet.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// faucet is a Ether faucet backed by a light client.
package main

//go:generate go-bindata -nometadata -o website.go faucet.html

import (
"bytes"
"context"
"encoding/json"
"flag"
"fmt"
"html/template"
"io/ioutil"
"math"
"math/big"
"net/http"
```

```
"net/url"
"os"
"path/filepath"
"strconv"
"strings"
"sync"
"time"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/eth"
"github.com/ethereum/go-ethereum/eth/downloader"
"github.com/ethereum/go-ethereum/ethclient"
"github.com/ethereum/go-ethereum/ethstats"
"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/p2p/discover"
"github.com/ethereum/go-ethereum/p2p/discv5"
"github.com/ethereum/go-ethereum/p2p/nat"
"github.com/ethereum/go-ethereum/params"
"golang.org/x/net/websocket"
)

var (
genesisFlag = flag.String("genesis", "", "Genesis json file to seed the chain with")
apiPortFlag = flag.Int("apiport", 8080, "Listener port for the HTTP API connection")
ethPortFlag = flag.Int("ethport", 30303, "Listener port for the devp2p connection")
bootFlag    = flag.String("bootnodes", "", "Comma separated bootnode enode URLs to seed with")
netFlag     = flag.Uint64("network", 0, "Network ID to use for the Ethereum protocol")
statsFlag   = flag.String("ethstats", "", "Ethstats network monitoring auth string")

netnameFlag = flag.String("faucet.name", "", "Network name to assign to the faucet")
payoutFlag  = flag.Int("faucet.amount", 1, "Number of Ethers to pay out per user request")
minutesFlag = flag.Int("faucet.minutes", 1440, "Number of minutes to wait between funding
rounds")
tiersFlag   = flag.Int("faucet.tiers", 3, "Number of funding tiers to enable (x3 time, x2.5 funds)")
```

```
accJSONFlag = flag.String("account.json", "", "Key json file to fund user requests with")
accPassFlag = flag.String("account.pass", "", "Decryption password to access faucet funds")

githubUser  = flag.String("github.user", "", "GitHub user to authenticate with for Gist access")
githubToken = flag.String("github.token", "", "GitHub personal token to access Gists with")

captchaToken  = flag.String("captcha.token", "", "Recaptcha site key to authenticate client side")
captchaSecret = flag.String("captcha.secret", "", "Recaptcha secret key to authenticate server
side")

logFlag = flag.Int("loglevel", 3, "Log level to use for Ethereum and the faucet")
)

var (
ether = new(big.Int).Exp(big.NewInt(10), big.NewInt(18), nil)
)

func main() {
// Parse the flags and set up the logger to print everything requested
flag.Parse()
log.Root().SetHandler(log.LvlFilterHandler(log.Lvl(*logFlag), log.StreamHandler(os.Stderr,
log.TerminalFormat(true))))

// Construct the payout tiers
amounts := make([]string, *tiersFlag)
periods := make([]string, *tiersFlag)
for i := 0; i < *tiersFlag; i++ {
// Calculate the amount for the next tier and format it
amount := float64(*payoutFlag) * math.Pow(2.5, float64(i))
amounts[i] = fmt.Sprintf("%s Ethers", strconv.FormatFloat(amount, 'f', -1, 64))
if amount == 1 {
amounts[i] = strings.TrimSuffix(amounts[i], "s")
}
// Calculate the period for the next tier and format it
period := *minutesFlag * int(math.Pow(3, float64(i)))
periods[i] = fmt.Sprintf("%d mins", period)
if period%60 == 0 {
period /= 60
periods[i] = fmt.Sprintf("%d hours", period)

if period%24 == 0 {
period /= 24
```

```
periods[i] = fmt.Sprintf("%d days", period)
}
}
if period == 1 {
periods[i] = strings.TrimSuffix(periods[i], "s")
}
}
// Load up and render the faucet website
tmpl, err := Asset("faucet.html")
if err != nil {
log.Crit("Failed to load the faucet template", "err", err)
}
website := new(bytes.Buffer)
err = template.Must(template.New("").Parse(string(tmpl))).Execute(website, map[string]interface{}{
"Network":   *netnameFlag,
"Amounts":   amounts,
"Periods":   periods,
"Recaptcha": *captchaToken,
})
if err != nil {
log.Crit("Failed to render the faucet template", "err", err)
}
// Load and parse the genesis block requested by the user
blob, err := ioutil.ReadFile(*genesisFlag)
if err != nil {
log.Crit("Failed to read genesis block contents", "genesis", *genesisFlag, "err", err)
}
genesis := new(core.Genesis)
if err = json.Unmarshal(blob, genesis); err != nil {
log.Crit("Failed to parse genesis block json", "err", err)
}
// Convert the bootnodes to internal enode representations
var enodes []*discv5.Node
for _, boot := range strings.Split(*bootFlag, ",") {
if url, err := discv5.ParseNode(boot); err == nil {
enodes = append(enodes, url)
} else {
log.Error("Failed to parse bootnode URL", "url", boot, "err", err)
}
}
// Load up the account key and decrypt its password
if blob, err = ioutil.ReadFile(*accPassFlag); err != nil {
```

```
log.Crit("Failed to read account password contents", "file", *accPassFlag, "err", err)
}
pass := string(blob)

ks := keystore.NewKeyStore(filepath.Join(os.Getenv("HOME"), ".faucet", "keys"),
keystore.StandardScryptN, keystore.StandardScryptP)
if blob, err = ioutil.ReadFile(*accJSONFlag); err != nil {
log.Crit("Failed to read account key contents", "file", *accJSONFlag, "err", err)
}
acc, err := ks.Import(blob, pass, pass)
if err != nil {
log.Crit("Failed to import faucet signer account", "err", err)
}
ks.Unlock(acc, pass)

// Assemble and start the faucet light service
faucet, err := newFaucet(genesis, *ethPortFlag, enodes, *netFlag, *statsFlag, ks, website.Bytes())
if err != nil {
log.Crit("Failed to start faucet", "err", err)
}
defer faucet.close()

if err := faucet.listenAndServe(*apiPortFlag); err != nil {
log.Crit("Failed to launch faucet API", "err", err)
}
}

// request represents an accepted funding request.
type request struct {
Username string             `json:"username"` // GitHub user for displaying an avatar
Account  common.Address     `json:"account"`  // Ethereum address being funded
Time     time.Time          `json:"time"`     // Timestamp when te request was accepted
Tx       *types.Transaction `json:"tx"`       // Transaction funding the account
}

// faucet represents a crypto faucet backed by an Ethereum light client.
type faucet struct {
config *params.ChainConfig // Chain configurations for signing
stack  *node.Node          // Ethereum protocol stack
client *ethclient.Client   // Client connection to the Ethereum chain
index  []byte              // Index page to serve up on the web
```

```
keystore *keystore.KeyStore // Keystore containing the single signer
account  accounts.Account   // Account funding user faucet requests
nonce    uint64             // Current pending nonce of the faucet
price    *big.Int           // Current gas price to issue funds with

conns    []*websocket.Conn    // Currently live websocket connections
timeouts map[string]time.Time // History of users and their funding timeouts
reqs     []*request           // Currently pending funding requests
update   chan struct{}        // Channel to signal request updates

lock sync.RWMutex // Lock protecting the faucet's internals
}

func newFaucet(genesis *core.Genesis, port int, enodes []*discv5.Node, network uint64, stats
string, ks *keystore.KeyStore, index []byte) (*faucet, error) {
// Assemble the raw devp2p protocol stack
stack, err := node.New(&node.Config{
Name:    "geth",
Version: params.Version,
DataDir: filepath.Join(os.Getenv("HOME"), ".faucet"),
P2P: p2p.Config{
NAT:              nat.Any(),
NoDiscovery:      true,
DiscoveryV5:      true,
ListenAddr:       fmt.Sprintf(":%d", port),
DiscoveryV5Addr:  fmt.Sprintf(":%d", port+1),
MaxPeers:         25,
BootstrapNodesV5: enodes,
},
})
if err != nil {
return nil, err
}
// Assemble the Ethereum light client protocol
if err := stack.Register(func(ctx *node.ServiceContext) (node.Service, error) {
cfg := eth.DefaultConfig
cfg.SyncMode = downloader.LightSync
cfg.NetworkId = network
cfg.Genesis = genesis
return les.New(ctx, &cfg)
}); err != nil {
return nil, err
```

```
}
// Assemble the ethstats monitoring and reporting service'
if stats != "" {
if err := stack.Register(func(ctx *node.ServiceContext) (node.Service, error) {
var serv *les.LightEthereum
ctx.Service(&serv)
return ethstats.New(stats, nil, serv)
}); err != nil {
return nil, err
}
}
// Boot up the client and ensure it connects to bootnodes
if err := stack.Start(); err != nil {
return nil, err
}
for _, boot := range enodes {
old, _ := discover.ParseNode(boot.String())
stack.Server().AddPeer(old)
}
// Attach to the client and retrieve and interesting metadatas
api, err := stack.Attach()
if err != nil {
stack.Stop()
return nil, err
}
client := ethclient.NewClient(api)

return &faucet{
config:   genesis.Config,
stack:    stack,
client:   client,
index:    index,
keystore: ks,
account:  ks.Accounts()[0],
timeouts: make(map[string]time.Time),
update:   make(chan struct{}, 1),
}, nil
}

// close terminates the Ethereum connection and tears down the faucet.
func (f *faucet) close() error {
return f.stack.Stop()
```

```
}

// listenAndServe registers the HTTP handlers for the faucet and boots it up
// for service user funding requests.
func (f *faucet) listenAndServe(port int) error {
go f.loop()

http.HandleFunc("/", f.webHandler)
http.Handle("/api", websocket.Handler(f.apiHandler))

return http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
}

// webHandler handles all non-api requests, simply flattening and returning the
// faucet website.
func (f *faucet) webHandler(w http.ResponseWriter, r *http.Request) {
w.Write(f.index)
}

// apiHandler handles requests for Ether grants and transaction statuses.
func (f *faucet) apiHandler(conn *websocket.Conn) {
// Start tracking the connection and drop at the end
f.lock.Lock()
f.conns = append(f.conns, conn)
f.lock.Unlock()

defer func() {
f.lock.Lock()
for i, c := range f.conns {
if c == conn {
f.conns = append(f.conns[:i], f.conns[i+1:]...)
break
}
}
f.lock.Unlock()
}()
// Send a few initial stats to the client
balance, _ := f.client.BalanceAt(context.Background(), f.account.Address, nil)
nonce, _ := f.client.NonceAt(context.Background(), f.account.Address, nil)

websocket.JSON.Send(conn, map[string]interface{}{
"funds":    balance.Div(balance, ether),
```

```
"funded":   nonce,
"peers":    f.stack.Server().PeerCount(),
"requests": f.reqs,
})
// Send the initial block to the client
ctx, cancel := context.WithTimeout(context.Background(), time.Second)
header, err := f.client.HeaderByNumber(ctx, nil)
cancel()

if err != nil {
log.Error("Failed to retrieve latest header", "err", err)
} else {
websocket.JSON.Send(conn, header)
}
// Keep reading requests from the websocket until the connection breaks
for {
// Fetch the next funding request and validate against github
var msg struct {
URL     string `json:"url"`
Tier    uint   `json:"tier"`
Captcha string `json:"captcha"`
}
if err := websocket.JSON.Receive(conn, &msg); err != nil {
return
}
if !strings.HasPrefix(msg.URL, "https://gist.github.com/") {
websocket.JSON.Send(conn, map[string]string{"error": "URL doesn't link to GitHub Gists"})
continue
}
if msg.Tier >= uint(*tiersFlag) {
websocket.JSON.Send(conn, map[string]string{"error": "Invalid funding tier requested"})
continue
}
log.Info("Faucet funds requested", "gist", msg.URL, "tier", msg.Tier)

// If captcha verifications are enabled, make sure we're not dealing with a robot
if *captchaToken != "" {
form := url.Values{}
form.Add("secret", *captchaSecret)
form.Add("response", msg.Captcha)

res, err := http.PostForm("https://www.google.com/recaptcha/api/siteverify", form)
```

```
if err != nil {
websocket.JSON.Send(conn, map[string]string{"error": err.Error()})
continue
}
var result struct {
Success bool            `json:"success"`
Errors  json.RawMessage `json:"error-codes"`
}
err = json.NewDecoder(res.Body).Decode(&result)
res.Body.Close()
if err != nil {
websocket.JSON.Send(conn, map[string]string{"error": err.Error()})
continue
}
if !result.Success {
log.Warn("Captcha verification failed", "err", string(result.Errors))
websocket.JSON.Send(conn, map[string]string{"error": "Beep-bop, you're a robot!"})
continue
}
}
// Retrieve the gist from the GitHub Gist APIs
parts := strings.Split(msg.URL, "/")
req, _ := http.NewRequest("GET", "https://api.github.com/gists/"+parts[len(parts)-1], nil)
if *githubUser != "" {
req.SetBasicAuth(*githubUser, *githubToken)
}
res, err := http.DefaultClient.Do(req)
if err != nil {
websocket.JSON.Send(conn, map[string]string{"error": err.Error()})
continue
}
var gist struct {
Owner struct {
Login string `json:"login"`
} `json:"owner"`
Files map[string]struct {
Content string `json:"content"`
} `json:"files"`
}
err = json.NewDecoder(res.Body).Decode(&gist)
res.Body.Close()
if err != nil {
```

```
websocket.JSON.Send(conn, map[string]string{"error": err.Error()})
continue
}
if gist.Owner.Login == "" {
websocket.JSON.Send(conn, map[string]string{"error": "Anonymous Gists not allowed"})
continue
}
// Iterate over all the files and look for Ethereum addresses
var address common.Address
for _, file := range gist.Files {
if len(file.Content) == 2+common.AddressLength*2 {
address = common.HexToAddress(file.Content)
}
}
if address == (common.Address{}) {
websocket.JSON.Send(conn, map[string]string{"error": "No Ethereum address found to fund"})
continue
}
// Validate the user's existence since the API is unhelpful here
if res, err = http.Head("https://github.com/" + gist.Owner.Login); err != nil {
websocket.JSON.Send(conn, map[string]string{"error": err.Error()})
continue
}
res.Body.Close()

if res.StatusCode != 200 {
websocket.JSON.Send(conn, map[string]string{"error": "Invalid user... boom!"})
continue
}
// Ensure the user didn't request funds too recently
f.lock.Lock()
var (
fund    bool
timeout time.Time
)
if timeout = f.timeouts[gist.Owner.Login]; time.Now().After(timeout) {
// User wasn't funded recently, create the funding transaction
amount := new(big.Int).Mul(big.NewInt(int64(*payoutFlag)), ether)
amount = new(big.Int).Mul(amount, new(big.Int).Exp(big.NewInt(5), big.NewInt(int64(msg.Tier)),
nil))
amount = new(big.Int).Div(amount, new(big.Int).Exp(big.NewInt(2), big.NewInt(int64(msg.Tier)),
nil))
```

```
tx := types.NewTransaction(f.nonce+uint64(len(f.reqs)), address, amount, big.NewInt(21000),
f.price, nil)
signed, err := f.keystore.SignTx(f.account, tx, f.config.ChainId)
if err != nil {
websocket.JSON.Send(conn, map[string]string{"error": err.Error()})
f.lock.Unlock()
continue
}
// Submit the transaction and mark as funded if successful
if err := f.client.SendTransaction(context.Background(), signed); err != nil {
websocket.JSON.Send(conn, map[string]string{"error": err.Error()})
f.lock.Unlock()
continue
}
f.reqs = append(f.reqs, &request{
Username: gist.Owner.Login,
Account:  address,
Time:     time.Now(),
Tx:       signed,
})
f.timeouts[gist.Owner.Login] = time.Now().Add(time.Duration(*minutesFlag*int(math.Pow(3,
float64(msg.Tier)))) * time.Minute)
fund = true
}
f.lock.Unlock()

// Send an error if too frequent funding, othewise a success
if !fund {
websocket.JSON.Send(conn, map[string]string{"error": fmt.Sprintf("%s left until next allowance",
common.PrettyDuration(timeout.Sub(time.Now())))})
continue
}
websocket.JSON.Send(conn, map[string]string{"success": fmt.Sprintf("Funding request accepted
for %s into %s", gist.Owner.Login, address.Hex())})
select {
case f.update <- struct{}{}:
default:
}
}
}
```

```
// loop keeps waiting for interesting events and pushes them out to connected
// websockets.
func (f *faucet) loop() {
// Wait for chain events and push them to clients
heads := make(chan *types.Header, 16)
sub, err := f.client.SubscribeNewHead(context.Background(), heads)
if err != nil {
log.Crit("Failed to subscribe to head events", "err", err)
}
defer sub.Unsubscribe()

for {
select {
case head := <-heads:
// New chain head arrived, query the current stats and stream to clients
balance, _ := f.client.BalanceAt(context.Background(), f.account.Address, nil)
balance = new(big.Int).Div(balance, ether)

price, _ := f.client.SuggestGasPrice(context.Background())
nonce, _ := f.client.NonceAt(context.Background(), f.account.Address, nil)

f.lock.Lock()
f.price, f.nonce = price, nonce
for len(f.reqs) > 0 && f.reqs[0].Tx.Nonce() < f.nonce {
f.reqs = f.reqs[1:]
}
f.lock.Unlock()

f.lock.RLock()
for _, conn := range f.conns {
if err := websocket.JSON.Send(conn, map[string]interface{}{
"funds":    balance,
"funded":   f.nonce,
"peers":    f.stack.Server().PeerCount(),
"requests": f.reqs,
}); err != nil {
log.Warn("Failed to send stats to client", "err", err)
conn.Close()
continue
}
if err := websocket.JSON.Send(conn, head); err != nil {
log.Warn("Failed to send header to client", "err", err)
```

```
conn.Close()
}
}
f.lock.RUnlock()

case <-f.update:
// Pending requests updated, stream to clients
f.lock.RLock()
for _, conn := range f.conns {
if err := websocket.JSON.Send(conn, map[string]interface{}{"requests": f.reqs}); err != nil {
log.Warn("Failed to send requests to client", "err", err)
conn.Close()
}
}
f.lock.RUnlock()
}
}
}

57:F:\git\coin\ethereum\go-ethereum\cmd\faucet\website.go
"path/filepath"
"strings"
"time"
)

func bindataRead(data []byte, name string) ([]byte, error) {
gz, err := gzip.NewReader(bytes.NewBuffer(data))
if err != nil {
return nil, fmt.Errorf("Read %q: %v", name, err)
}

var buf bytes.Buffer
_, err = io.Copy(&buf, gz)
clErr := gz.Close()

if err != nil {
return nil, fmt.Errorf("Read %q: %v", name, err)
}
if clErr != nil {
return nil, err
}
```

```
return buf.Bytes(), nil
}

type asset struct {
bytes []byte
info  os.FileInfo
}

type bindataFileInfo struct {
name    string
size    int64
mode    os.FileMode
modTime time.Time
}

func (fi bindataFileInfo) Name() string {
return fi.name
}
func (fi bindataFileInfo) Size() int64 {
return fi.size
}
func (fi bindataFileInfo) Mode() os.FileMode {
return fi.mode
}
func (fi bindataFileInfo) ModTime() time.Time {
return fi.modTime
}
func (fi bindataFileInfo) IsDir() bool {
return false
}
func (fi bindataFileInfo) Sys() interface{} {
return nil
}

var _faucetHtml =
[]byte("\x1f\x8b\x08\x00\x00\x00\x00\x00\x00\xff\xac\x59\x6d\x6f\xdc\x36\x12\xfe\xec\xfc\x8a\xa9\
x2e\xad\x77\x61\x4b\xb2\xe3\x20\x2d\xd6\xd2\x16\x41\x9a\x4b\x7b\x38\xb4\x45\x9b\xe2\xae\x68\
x8b\x03\x25\xcd\x4a\x8c\x29\x52\x25\x87\xbb\xde\x1a\xfb\xdf\x0f\x24\x25\xad\x76\x6d\xa7\xb9\x4
b\xf3\x61\x23\x92\x33\xcf\xbc\x51\xf3\x22\x67\x9f\x7c\xf5\xdd\xab\xb7\x3f\x7f\xff\x1a\x1a\x6a\xc5\
xf2\x49\xe6\xfe\x03\xc1\x64\x9d\x47\x28\xa3\xe5\x93\x93\xac\x41\x56\x2d\x9f\x9c\x9c\x64\x2d\x1
2\x83\xb2\x61\xda\x20\xe5\x91\xa5\x55\xfc\x45\xb4\x3f\x68\x88\xba\x18\x7f\xb7\x7c\x9d\x47\xff\x
8e\x7f\x7a\x19\xbf\x52\x6d\xc7\x88\x17\x02\x23\x28\x95\x24\x94\x94\x47\xdf\xbc\xce\xb1\xaa\x71
```

\xc2\x27\x59\x8b\x79\xb4\xe6\xb8\xe9\x94\xa6\x09\xe9\x86\x57\xd4\xe4\x15\xae\x79\x89\xb1\x5f\x9c\x03\x97\x9c\x38\x13\xb1\x29\x99\xc0\xfc\x32\x5a\x3e\x71\x38\xc4\x49\xe0\xf2\xee\x2e\xf9\x16\x69\xa3\xf4\xcd\x6e\xb7\x80\x37\x9c\xbe\xb6\x05\xfc\x9d\xd9\x12\x29\x4b\x03\x89\xa7\x16\x5c\xde\x40\xa3\x71\x95\x47\x4e\x67\xb3\x48\xd3\xb2\x92\xef\x4c\x52\x0a\x65\xab\x95\x60\x1a\x93\x52\xb5\x29\x7b\xc7\x6e\x53\xc1\x0b\x93\xd2\x86\x13\xa1\x8e\x0b\xa5\xc8\x90\x66\x5d\x7a\x95\x5c\x25\x9f\xa7\xa5\x31\xe9\xb8\x97\xb4\x5c\x26\xa5\x31\x11\x68\x14\x79\x64\x68\x2b\xd0\x34\x88\x14\x41\xba\xfc\xff\xe4\xae\x94\xa4\x98\x6d\xd0\xa8\x16\xd3\xe7\xc9\xe7\xc9\x85\x17\x39\xdd\x7e\xbf\x54\x27\xd6\x94\x9a\x77\x04\x46\x97\x1f\x2c\xf7\xdd\xef\x16\xf5\x36\xbd\x4a\x2e\x93\xcb\x7e\xe1\xe5\xbc\x33\xd1\x32\x4b\x03\xe0\xf2\xa3\xb0\x63\xa9\x68\x9b\x3e\x4b\x9e\x27\x97\x69\xc7\xca\x1b\x56\x63\x35\x48\x72\x47\xc9\xb0\xf9\x97\xc9\x7d\x2c\x86\xef\x8e\x43\xf8\x57\x08\x6b\x55\x8b\x92\x92\x77\x26\x7d\x96\x5c\x7e\x91\x5c\x0c\x1b\xf7\xf1\xbd\x00\x17\x34\x27\xea\x24\x59\xa3\x26\x5e\x32\x11\x97\x28\x09\x35\xdc\xb9\xdd\x93\x96\xcb\xb8\x41\x5e\x37\xb4\x80\xcb\x8b\x8b\x4f\xaf\x1f\xda\x5d\x37\x61\xbb\xe2\xa6\x13\x6c\xbb\x80\x95\xc0\xdb\xb0\xc5\x04\xaf\x65\xcc\x09\x5b\xb3\x80\x80\xec\x0f\x76\x5e\x66\xa7\x55\xad\xd1\x98\x5e\x58\xa7\x0c\x27\xae\xe4\xc2\xdd\x28\x46\x7c\x8d\x0f\xd1\x9a\x8e\xc9\x7b\x0c\xac\x30\x4a\x58\xc2\x23\x45\x0a\xa1\xca\x9b\xb0\xe7\x5f\xe3\xa9\x11\xa5\x12\x4a\x2f\x60\xd3\xf0\x9e\x0d\xbc\x20\xe8\x34\xf6\xf0\xd0\xb1\xaa\xe2\xb2\x5e\xc0\x8b\xae\xb7\x07\x5a\xa6\x6b\x2e\x17\x70\xb1\x67\xc9\xd2\xc1\x8d\x59\x1a\x32\xd6\x93\x93\xac\x50\xd5\xd6\xc7\xb0\xe2\x6b\x28\x05\x33\x26\x8f\x8e\x5c\xec\x33\xd1\x01\x81\x4b\x40\x8c\xcb\xe1\xe8\xe0\x4c\xab\x4d\x04\x5e\x50\x1e\x05\x25\xe2\x42\x11\xa9\x76\x01\x97\x4e\xbd\x9e\xe5\x08\x4f\xc4\xa2\x8e\x2f\x9f\x0d\x87\x27\x59\x73\x39\x80\x10\xde\x52\xec\xe3\x33\x46\x26\x5a\x66\x7c\xe0\x5d\x31\x58\xb1\xb8\x60\xd4\x44\xc0\x34\x67\x71\xc3\xab\x0a\x65\x1e\x91\xb6\xe8\xee\x11\x5f\xc2\x34\xef\x0d\x69\xef\xa5\xa5\x06\xa5\xb3\x93\xb0\xea\x93\x20\x1c\xc3\xd6\x9c\x1a\x5b\xc4\x4c\xd0\xa3\xe0\x59\xda\x5c\x0e\x26\xa5\x15\x5f\xf7\x1e\x99\x3c\x1e\x39\xe7\x71\xfb\xbf\x80\xfe\x41\xad\x56\x06\x29\x9e\xb8\x63\x42\xcc\x65\x67\x29\xae\xb5\xb2\xdd\x78\x7e\x92\xf9\x5d\xe0\x55\x1e\xd5\xdc\x50\x04\xb4\xed\x7a\xdf\x45\xa3\x49\x4a\xb7\xb1\x0b\x9d\x56\x22\x82\x4e\xb0\x12\x1b\x25\x2a\xd4\x79\xd4\xfb\xe4\x0d\x37\x04\x3f\xfd\xf0\x4f\xe8\x03\xcc\x65\x0d\x5b\x65\x35\xbc\xa6\x06\x35\xda\x16\x58\x55\xb9\xcb\x9d\x24\xc9\x44\xb6\xbf\xe9\xf7\xb5\x8b\x0b\x92\x7b\xaa\x93\xac\xb0\x44\x6a\x24\x2c\x48\x42\x41\x32\xae\x70\xc5\xac\x20\xa8\xb4\xea\x2a\xb5\x91\x31\xa9\xba\x76\x05\x31\x58\x10\x98\x22\xa8\x18\xb1\xfe\x28\x8f\x06\xda\x21\x28\xcc\x74\xaa\xb3\x5d\x1f\x96\xb0\x89\xb7\x1d\x93\x15\x56\x2e\x94\xc2\x60\xb4\x7c\xc3\xd7\x08\x2d\x06\x5b\x4e\x8e\x23\x5d\x32\x8d\x14\x4f\x41\x1f\x88\x74\x50\x26\x98\x04\xfd\xbf\xcc\x8a\x01\x69\x34\xa1\x45\x69\xe1\x60\x15\x6b\x97\x85\xa2\xe5\xdd\x9d\x66\xb2\x46\x78\xca\xab\xdb\x73\x78\xca\x5a\x65\x25\xc1\x22\x87\xe4\xa5\x7f\x34\xbb\xdd\x01\x3a\x40\x26\xf8\x32\x63\xef\x7b\x19\x40\xc9\x52\xf0\xf2\x26\x8f\x88\xa3\xce\xef\xee\x1c\xf8\x6e\x77\x0d\x77\x77\x7c\x05\x4f\x93\x1f\xb0\x64\x1d\x95\x0d\xdb\xed\x6a\x3d\x3c\x27\x78\x8b\xa5\x25\x9c\xcd\xef\xee\x50\x18\xdc\xed\x8c\x2d\x5a\x4e\xb3\x81\xdd\xed\xcb\x6a\xb7\x73\x3a\xf7\x7a\xee\x76\x90\x3a\x50\x59\xe1\x2d\x3c\x4d\xbe\x47\xcd\x55\x65\x20\xd0\x67\x29\x5b\x66\xa9\xe0\xcb\x9e\xef\xd0\x49\xa9\x15\xfb\xfb\x92\xba\x0b\x33\x5e\x6d\xff\xa6\x78\x55\xa7\x9a\x3e\x70\xf1\xeb\x78\xd4\xbe\xbf\x0f\x86\x13\xde\xe0\x36\x8f\xee\xee\xa6\xbc\xfd\x69\xc9\x84\x28\x98\xf3\x4b\x30\x6d\x64\xfa\x03\xdd\x3d\x5d\x73\xe3\x3b\xaf\xe5\xa0\xc1\x5e\xed\x0f\x7c\x93\x8f\xd2\x1c\xa9\x6e\x01\x57\xcf\x26\x39\xee\xa1\x97\xfc\xc5\xd1\x4b\x7e\xf5\x20\x71\xc7\x24\x0a\xf0\xbf\xb1\x69\x99\x18\x9e\xfb\xb7\x65\xf2\xf2\x1d\x33\xc5\x2e\xa3\x8f\xaa\x8d\x95\xe1\xe2\x1a\xd4\x1a\xf5\x4a\xa8\xcd\

x02\x98\x25\x75\x0d\x2d\xbb\x1d\xab\xe3\xd5\xc5\xc5\x54\x6f\xd7\x31\xb2\x42\xa0\x4f\x28\x1a\x7f\xb7\x68\xc8\x8c\x89\x24\x1c\xf9\x5f\x97\x4f\x2a\x94\x06\xab\x23\x6f\x38\x89\xce\xb5\x9e\x6a\x12\xfa\xd1\x99\x0f\xea\xbe\x52\x6a\x2c\x38\x53\x35\x7a\xe8\x49\x6d\x8c\x96\x19\xe9\x3d\xdd\x49\x46\xd5\xff\x54\x30\xb4\x6b\x08\x1f\xab\x17\x21\xa3\x39\xdb\x3b\x44\x1d\xba\x11\x77\x65\xc1\x2f\xb3\x94\xaa\x8f\x90\xec\x2e\x61\xc1\x0c\x7e\x88\x78\xdf\x17\xec\xc5\xfb\xe5\xc7\xca\x6f\x90\x69\x2a\x90\x3d\x5e\xd2\x26\x0a\xac\xac\xac\x26\xf6\xfb\xdc\xf9\xb1\x0a\x58\xc9\xd7\xa8\x0d\xa7\xed\x87\x6a\x80\xd5\x5e\x85\xb0\x3e\x54\x21\x4b\x49\xbf\xff\xae\x4d\x17\x7f\xd1\xcb\xfd\x67\x0d\xcc\xd5\xf2\x6b\xb5\x81\x4a\xa1\x01\x6a\xb8\x01\xd7\x7e\x7c\x99\xa5\xcd\xd5\x48\xd2\x2d\xdf\xba\x03\xef\x54\x58\x85\x0e\x84\x1b\xd0\x56\xfa\xca\xab\x24\x50\x83\x87\xcd\x8b\x0c\x4f\x09\xbc\x55\xae\x01\x5c\xa3\x24\x68\x99\xe0\x25\x57\xd6\x00\x2b\x49\x69\x03\x2b\xad\x5a\xc0\xdb\x86\x59\x43\x0e\xc8\xa5\x0f\xb6\x66\x5c\xf8\x77\xc9\x87\x14\x94\x06\x56\x96\xb6\xb5\xae\x81\x95\x35\xa0\x54\xb6\x6e\x7a\x5d\x48\x41\x28\x4c\x42\xc9\x7a\xd4\xc7\x74\xac\x05\x46\xc4\xca\x1b\x73\x0e\x43\x56\x00\xa6\x11\x88\x63\xe5\xb8\xfa\x3e\x82\x95\xa5\x2f\x66\x09\xbc\x94\x5b\x25\x11\x1a\xb6\xf6\x8a\x1c\x11\x40\xcb\xb6\x03\x50\xaf\xd7\x86\x53\xc3\x83\xe1\x1d\xea\xd6\x4d\x24\x15\x08\xde\x72\x32\x49\x96\x76\x53\xdf\xa9\x43\xd6\x73\x30\xbc\xed\xc4\x16\x4a\x8d\x8c\x10\x18\x64\xec\x68\x98\x74\xad\x51\x12\x7a\x3a\x3f\x8e\x44\x40\x4c\xd7\x6e\x54\xff\x0f\x2b\x94\xa5\x45\x21\x98\xbc\x71\xad\xc2\xd8\x0e\xb9\xb2\xe6\x95\x7a\xb8\x11\x82\x8e\x19\xa7\x21\x97\xa4\xbc\xd2\xfd\x6c\x6e\x60\xe6\x56\x2b\x2e\xd0\x8f\xef\xfe\x1e\xc8\x53\x67\xb1\x9b\xb1\xe6\xe7\x50\xaa\x6e\x1b\xb8\x3d\x9f\x53\xcd\xf8\xde\x6b\x84\x62\x85\x5a\x23\x84\xc6\xae\x50\xb7\xc0\x64\x05\x2b\xae\x11\xd8\x86\x6d\x3f\x81\x9f\x95\x85\x92\x49\x20\xcd\xca\x9b\x20\xdb\x6a\xed\x2e\x44\x87\xd2\x25\xfd\x7d\x88\x0a\x14\x6a\xe3\x49\x02\xda\x8a\xa3\xf0\xf1\x32\x88\xd0\xa8\x0d\xb4\xb6\xf4\x06\xba\x40\xa1\x3b\xd8\x30\x4e\x60\x25\x71\x11\xec\x26\xab\x25\x94\xaa\xc5\x83\x28\xdc\xab\xda\x19\xb6\xcb\xb7\xce\xee\x7b\x97\x79\xac\xb7\xa0\xf1\x55\x20\x87\x4e\x2b\xc2\xd2\x0d\x46\xc0\x6a\xc6\xa5\x71\x76\xfa\x38\x63\xfb\x01\xf5\x78\x7c\xea\x1f\xf6\x93\xa8\x3f\x4e\x53\x78\x23\x54\xc1\x04\xac\x5d\x96\x29\x84\x7b\x11\x15\xb8\x96\xf7\xc0\x5b\x86\x18\x59\x03\x6a\xe5\x77\x83\xe6\x8e\x7f\xcd\xb4\xbb\xed\xd8\x76\x04\x79\x3f\x47\xb9\x3d\x83\x7a\xdd\x4f\x87\x6e\xe9\x7a\xae\x70\xde\x0b\xfd\x0a\x57\x5c\x86\xa0\xae\xac\x0c\xe6\x51\xc3\x08\x42\x17\x62\x80\xf9\x60\x83\xd5\x02\xfa\x48\x07\xc8\x51\x80\xa7\x83\x7c\x64\x9f\xdd\xf3\x73\xff\xd0\xfb\x68\xde\xcf\x81\x01\x26\x31\x28\xab\xd9\x3f\x7e\xfc\xee\xdb\xc4\x90\xe6\xb2\xe6\xab\xed\xec\xce\x6a\xb1\x80\xa7\xb3\xe8\x6f\x7e\x3c\x98\xff\x72\xf1\x5b\xb2\x66\xc2\xe2\xb9\x37\x60\xe1\x7f\xef\x89\x39\x87\xfe\x71\x01\x87\x12\x77\xf3\xf9\xf5\xc3\x2d\xdb\xa4\xc3\xd4\x68\x90\x66\x8e\x70\x8c\xe4\xee\xfa\xd0\x49\x0c\x5a\xa4\x46\xf9\xbb\xa8\xb1\x54\x52\x62\x49\x60\x3b\x25\x7b\x9f\x80\x50\xc6\x0c\x8e\xd9\x53\x4c\x7c\x33\x18\xcf\x57\x30\x1b\xc2\xf5\x29\x3c\x83\x3c\x87\x8b\xe1\xac\xf7\x0c\xe4\x20\x71\x03\xff\xc2\xe2\x47\x55\xde\x20\xcd\xa2\x8d\x71\x69\x21\x82\x33\x10\xaa\x64\x0e\x2f\x69\x94\x21\x38\x83\x28\x65\x1d\x8f\xe6\x61\x9a\xde\x81\x6b\x91\xff\x1c\xec\x83\xb0\xc2\xf7\x86\xa0\xe9\xd9\x59\xb8\x36\x43\xe8\x94\x6c\xd1\x18\x56\xe3\xd4\x42\x9f\xe5\x47\x53\x9c\x23\x5a\x53\x43\x0e\x3e\xc4\x1d\xd3\x06\x03\x49\xe2\x3a\x8b\x5e\x8a\x77\x87\x27\xcb\x73\x90\x56\x88\x91\xff\x44\xa3\x7b\x99\x7b\xb2\xdd\x93\x03\xf2\x24\x24\xe1\x4f\xf2\x1c\x5c\x99\x75\x31\xaa\xf6\x9c\xee\xfa\x84\x86\x60\x9e\xb8\x4a\xbf\xe7\x98\x8f\x70\xf7\xd0\xb0\xfa\x33\x38\xac\x8e\xf1\xb0\x7a\x04\xd0\xf7\x5f\xef\xc3\x0b\xfd\xda\x04\xce\x6f\x3c\x82\x26\x6d\x5b\xa0\x7e\x1f\x5c\xe8\xbf\x7a\x38\xef\xea\x6f\x24\x4d\x78\xcf\xe1\xf2\xc5\xfc\x11\x74\xd4\x5a\x3d\x0a\x2e\x15\x6d\x67\x77\x82\x6d\x5d\xd5\x81\x53\x52\xdd\x2b\xdf\x2e\x9d\x9e\x83\x93\xb5\x80\x11\xe1\xdc\

```
x0f\xc2\x0b\x38\xf5\xab\xd3\xdd\x23\xd2\x8c\x2d\x4b\x57\x8f\x3e\x46\x5e\x8f\x31\x4a\xec\xd7\x8f
\xca\x1c\xeb\xcb\x81\x50\xf8\xec\x33\xb8\x77\x7a\x78\x05\xdd\x1d\xee\x0b\x25\xe4\x10\x45\x3d\
xfc\xc9\x4a\x69\x98\xb9\x43\x9e\x5f\x5c\x03\xcf\xa6\x30\x89\x40\x59\x53\x73\x0d\xfc\xec\x6c\x8f\
x74\x32\xc0\x9c\xe5\x10\xb9\x89\x20\xa3\x6a\xe9\x3b\xb3\xd0\xbe\xfd\x1a\xb9\x09\xb0\xd6\xca\x
ca\x6a\xe1\x52\xee\xec\x74\xdf\x0c\x4c\xfa\x80\xb3\x03\x95\x7f\xe1\xbf\x25\xd6\xa0\xf6\x95\xfb\x
0c\xa2\xa4\x93\xf5\x97\x7e\x6e\x7c\xf1\xfc\x74\x7e\x0d\x7b\x4c\x3f\x4d\x2e\xa0\x74\xb3\xd5\x35\
x84\xf9\xc4\x77\x89\x30\x4e\x56\x7e\x55\x28\x5d\xa1\x8e\x35\xab\xb8\x35\x0b\x78\xde\xdd\x5e\
xff\x3a\x4c\x9e\xbe\x97\xf5\x7a\x77\x1a\x97\x0f\xe9\x32\xb4\x4b\x67\x10\x65\xa9\x23\x1a\x58\x4
6\x2b\xa7\x5f\x0d\xe1\x81\x2e\x1c\xc6\x6f\x7a\xfd\x7e\xcb\xab\x4a\xa0\x53\xc2\x0b\x0c\x1f\x5f\x
2b\xab\x7d\xe2\x9a\x85\xf5\xec\x58\x0f\xe2\x2d\xce\x13\x2b\xf9\xed\x6c\x1e\xf7\x34\xc3\xfa\x1c\
x4e\x8d\xcb\xcf\x95\x39\x9d\x27\x8d\x6d\x99\xe4\x7f\xe0\xcc\xb5\xf4\xf3\xa0\xb7\xd3\xd8\xf5\xe9
\x63\xb4\x77\x93\x17\x6d\x9c\x31\xe7\x49\x43\xad\x98\x45\x19\xf9\x2f\x93\x4e\xb9\x31\xc4\x1e\x
25\x6c\x1f\xde\xc8\xdd\x61\x0e\x2d\x85\x32\x78\x54\x23\xc0\x20\xbd\xe5\x2d\x2a\x4b\xb3\xb1\x8
e\x9c\xbb\xb9\xf7\x62\x7e\x0d\xbb\xfd\x07\xdc\x34\x85\xd7\xc6\x4d\x12\xdc\x34\xc0\x60\x83\x85\
xf1\xf9\x1d\x7a\x1e\x5f\xce\x43\xd9\x7e\xf9\xfd\x37\x93\xd2\x3d\xa2\xce\xbc\x72\xe3\x07\xec\x87
\xea\xe4\x83\x5f\xcc\x37\x9b\x4d\x52\x2b\x55\x8b\xf0\xad\x7c\x2c\xa4\xae\x7a\x24\xef\xdc\xb8\x
6a\xb6\xb2\x84\x0a\x57\xa8\x97\x13\xf8\xbe\xba\x66\x69\xf8\x96\x9b\xa5\xe1\xef\x54\xff\x0d\x00\
x00\xff\xff\x71\x50\x77\xf3\xb8\x1a\x00\x00")

func faucetHtmlBytes() ([]byte, error) {
return bindataRead(
_faucetHtml,
"faucet.html",
)
}

func faucetHtml() (*asset, error) {
bytes, err := faucetHtmlBytes()
if err != nil {
return nil, err
}

info := bindataFileInfo{name: "faucet.html", size: 0, mode: os.FileMode(0), modTime: time.Unix(0,
0)}
a := &asset{bytes: bytes, info: info}
return a, nil
}

// Asset loads and returns the asset for the given name.
// It returns an error if the asset could not be found or
// could not be loaded.
func Asset(name string) ([]byte, error) {
```

```
cannonicalName := strings.Replace(name, "\\", "/", -1)
if f, ok := _bindata[cannonicalName]; ok {
a, err := f()
if err != nil {
return nil, fmt.Errorf("Asset %s can't read by error: %v", name, err)
}
return a.bytes, nil
}
return nil, fmt.Errorf("Asset %s not found", name)
}

// MustAsset is like Asset but panics when Asset would return an error.
// It simplifies safe initialization of global variables.
func MustAsset(name string) []byte {
a, err := Asset(name)
if err != nil {
panic("asset: Asset(" + name + "): " + err.Error())
}

return a
}

// AssetInfo loads and returns the asset info for the given name.
// It returns an error if the asset could not be found or
// could not be loaded.
func AssetInfo(name string) (os.FileInfo, error) {
cannonicalName := strings.Replace(name, "\\", "/", -1)
if f, ok := _bindata[cannonicalName]; ok {
a, err := f()
if err != nil {
return nil, fmt.Errorf("AssetInfo %s can't read by error: %v", name, err)
}
return a.info, nil
}
return nil, fmt.Errorf("AssetInfo %s not found", name)
}

// AssetNames returns the names of the assets.
func AssetNames() []string {
names := make([]string, 0, len(_bindata))
for name := range _bindata {
names = append(names, name)
```

```
}
return names
}

// _bindata is a table, holding each asset generator, mapped to its name.
var _bindata = map[string]func() (*asset, error){
"faucet.html": faucetHtml,
}

// AssetDir returns the file names below a certain
// directory embedded in the file by go-bindata.
// For example if you run go-bindata on data/... and data contains the
// following hierarchy:
//     data/
//       foo.txt
//       img/
//         a.png
//         b.png
// then AssetDir("data") would return []string{"foo.txt", "img"}
// AssetDir("data/img") would return []string{"a.png", "b.png"}
// AssetDir("foo.txt") and AssetDir("notexist") would return an error
// AssetDir("") will return []string{"data"}.
func AssetDir(name string) ([]string, error) {
node := _bintree
if len(name) != 0 {
cannonicalName := strings.Replace(name, "\\", "/", -1)
pathList := strings.Split(cannonicalName, "/")
for _, p := range pathList {
node = node.Children[p]
if node == nil {
return nil, fmt.Errorf("Asset %s not found", name)
}
}
}
if node.Func != nil {
return nil, fmt.Errorf("Asset %s not found", name)
}
rv := make([]string, 0, len(node.Children))
for childName := range node.Children {
rv = append(rv, childName)
}
return rv, nil
```

```
}

type bintree struct {
Func     func() (*asset, error)
Children map[string]*bintree
}
var _bintree = &bintree{nil, map[string]*bintree{
"faucet.html": &bintree{faucetHtml, map[string]*bintree{}},
}}

// RestoreAsset restores an asset under the given directory
func RestoreAsset(dir, name string) error {
data, err := Asset(name)
if err != nil {
return err
}
info, err := AssetInfo(name)
if err != nil {
return err
}
err = os.MkdirAll(_filePath(dir, filepath.Dir(name)), os.FileMode(0755))
if err != nil {
return err
}
err = ioutil.WriteFile(_filePath(dir, name), data, info.Mode())
if err != nil {
return err
}
err = os.Chtimes(_filePath(dir, name), info.ModTime(), info.ModTime())
if err != nil {
return err
}
return nil
}

// RestoreAssets restores an asset under the given directory recursively
func RestoreAssets(dir, name string) error {
children, err := AssetDir(name)
// File
if err != nil {
return RestoreAsset(dir, name)
}
```

```
// Dir
for _, child := range children {
err = RestoreAssets(dir, filepath.Join(name, child))
if err != nil {
return err
}
}
return nil
}

func _filePath(dir, name string) string {
cannonicalName := strings.Replace(name, "\\", "/", -1)
return filepath.Join(append([]string{dir}, strings.Split(cannonicalName, "/")...)...)
}


58:F:\git\coin\ethereum\go-ethereum\cmd\geth\accountcmd.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"fmt"
"io/ioutil"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/console"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/log"
"gopkg.in/urfave/cli.v1"
)

var (
walletCommand = cli.Command{
Name:      "wallet",
Usage:     "Manage Ethereum presale wallets",
ArgsUsage: "",
Category:  "ACCOUNT COMMANDS",
Description: `
    geth wallet import /path/to/my/presale.wallet
```

will prompt for your password and imports your ether presale account.
It can be used non-interactively with the --password option taking a
passwordfile as argument containing the wallet password in plaintext.`,
Subcommands: []cli.Command{
{

Name: "import",
Usage: "Import Ethereum presale wallet",
ArgsUsage: "<keyFile>",
Action: utils.MigrateFlags(importWallet),
Category: "ACCOUNT COMMANDS",
Flags: []cli.Flag{
utils.DataDirFlag,
utils.KeyStoreDirFlag,
utils.PasswordFileFlag,
utils.LightKDFFlag,
},
Description: `
geth wallet [options] /path/to/my/presale.wallet

will prompt for your password and imports your ether presale account.
It can be used non-interactively with the --password option taking a
passwordfile as argument containing the wallet password in plaintext.`,
},
},
}

accountCommand = cli.Command{
Name: "account",
Usage: "Manage accounts",
Category: "ACCOUNT COMMANDS",
Description: `

Manage accounts, list all existing accounts, import a private key into a new
account, create a new account or update an existing account.

It supports interactive mode, when you are prompted for password as well as
non-interactive mode where passwords are supplied via a given password file.
Non-interactive mode is only meant for scripted use on test networks or known
safe environments.

Make sure you remember the password you gave when creating a new account (with either new or import). Without it you are not able to unlock your account.

Note that exporting your key in unencrypted format is NOT supported.

Keys are stored under <DATADIR>/keystore.
It is safe to transfer the entire directory or the individual keys therein
between ethereum nodes by simply copying.

Make sure you backup your keys regularly.`,
Subcommands: []cli.Command{
{
Name: "list",
Usage: "Print summary of existing accounts",
Action: utils.MigrateFlags(accountList),
Flags: []cli.Flag{
utils.DataDirFlag,
utils.KeyStoreDirFlag,
},
Description: `
Print a short summary of all accounts`,
},
{
Name: "new",
Usage: "Create a new account",
Action: utils.MigrateFlags(accountCreate),
Flags: []cli.Flag{
utils.DataDirFlag,
utils.KeyStoreDirFlag,
utils.PasswordFileFlag,
utils.LightKDFFlag,
},
Description: `
    geth account new

Creates a new account and prints the address.

The account is saved in encrypted format, you are prompted for a passphrase.

You must remember this passphrase to unlock your account in the future.

For non-interactive use the passphrase can be specified with the --password flag:

```
Note, this is meant to be used for testing only, it is a bad idea to save your
password to file or expose in any other way.
`,
},
{
Name:      "update",
Usage:     "Update an existing account",
Action:    utils.MigrateFlags(accountUpdate),
ArgsUsage: "<address>",
Flags: []cli.Flag{
utils.DataDirFlag,
utils.KeyStoreDirFlag,
utils.LightKDFFlag,
},
Description: `
    geth account update <address>

Update an existing account.

The account is saved in the newest version in encrypted format, you are prompted
for a passphrase to unlock the account and another to save the updated file.

This same command can therefore be used to migrate an account of a deprecated
format to the newest format or change the password for an account.

For non-interactive use the passphrase can be specified with the --password flag:

    geth account update [options] <address>

Since only one password can be given, only format update can be performed,
changing your password is only possible interactively.
`,
},
{
Name:   "import",
Usage:  "Import a private key into a new account",
Action: utils.MigrateFlags(accountImport),
Flags: []cli.Flag{
utils.DataDirFlag,
utils.KeyStoreDirFlag,
utils.PasswordFileFlag,
```

```
utils.LightKDFFlag,
},
ArgsUsage: "<keyFile>",
Description: `
    geth account import <keyfile>

Imports an unencrypted private key from <keyfile> and creates a new account.
Prints the address.

The keyfile is assumed to contain an unencrypted private key in hexadecimal format.

The account is saved in encrypted format, you are prompted for a passphrase.

You must remember this passphrase to unlock your account in the future.

For non-interactive use the passphrase can be specified with the -password flag:

    geth account import [options] <keyfile>

Note:
As you can directly copy your encrypted accounts to another ethereum instance,
this import mechanism is not needed when you transfer an account between
nodes.
`,
},
},
}
)

func accountList(ctx *cli.Context) error {
stack, _ := makeConfigNode(ctx)
var index int
for _, wallet := range stack.AccountManager().Wallets() {
for _, account := range wallet.Accounts() {
fmt.Printf("Account #%d: {%x} %s\n", index, account.Address, &account.URL)
index++
}
}
return nil
}

// tries unlocking the specified account a few times.
```

```
func unlockAccount(ctx *cli.Context, ks *keystore.KeyStore, address string, i int, passwords
[]string) (accounts.Account, string) {
account, err := utils.MakeAddress(ks, address)
if err != nil {
utils.Fatalf("Could not list accounts: %v", err)
}
for trials := 0; trials < 3; trials++ {
prompt := fmt.Sprintf("Unlocking account %s | Attempt %d/%d", address, trials+1, 3)
password := getPassPhrase(prompt, false, i, passwords)
err = ks.Unlock(account, password)
if err == nil {
log.Info("Unlocked account", "address", account.Address.Hex())
return account, password
}
if err, ok := err.(*keystore.AmbiguousAddrError); ok {
log.Info("Unlocked account", "address", account.Address.Hex())
return ambiguousAddrRecovery(ks, err, password), password
}
if err != keystore.ErrDecrypt {
// No need to prompt again if the error is not decryption-related.
break
}
}
// All trials expended to unlock account, bail out
utils.Fatalf("Failed to unlock account %s (%v)", address, err)

return accounts.Account{}, ""
}

// getPassPhrase retrieves the password associated with an account, either fetched
// from a list of preloaded passphrases, or requested interactively from the user.
func getPassPhrase(prompt string, confirmation bool, i int, passwords []string) string {
// If a list of passwords was supplied, retrieve from them
if len(passwords) > 0 {
if i < len(passwords) {
return passwords[i]
}
return passwords[len(passwords)-1]
}
// Otherwise prompt the user for the password
if prompt != "" {
fmt.Println(prompt)
```

```
}
password, err := console.Stdin.PromptPassword("Passphrase: ")
if err != nil {
utils.Fatalf("Failed to read passphrase: %v", err)
}
if confirmation {
confirm, err := console.Stdin.PromptPassword("Repeat passphrase: ")
if err != nil {
utils.Fatalf("Failed to read passphrase confirmation: %v", err)
}
if password != confirm {
utils.Fatalf("Passphrases do not match")
}
}
return password
}

func ambiguousAddrRecovery(ks *keystore.KeyStore, err *keystore.AmbiguousAddrError, auth
string) accounts.Account {
fmt.Printf("Multiple key files exist for address %x:\n", err.Addr)
for _, a := range err.Matches {
fmt.Println("  ", a.URL)
}
fmt.Println("Testing your passphrase against all of them...")
var match *accounts.Account
for _, a := range err.Matches {
if err := ks.Unlock(a, auth); err == nil {
match = &a
break
}
}
if match == nil {
utils.Fatalf("None of the listed files could be unlocked.")
}
fmt.Printf("Your passphrase unlocked %s\n", match.URL)
fmt.Println("In order to avoid this warning, you need to remove the following duplicate key files:")
for _, a := range err.Matches {
if a != *match {
fmt.Println("  ", a.URL)
}
}
return *match
```

```
}

// accountCreate creates a new account into the keystore defined by the CLI flags.
func accountCreate(ctx *cli.Context) error {
stack, _ := makeConfigNode(ctx)
password := getPassPhrase("Your new account is locked with a password. Please give a
password. Do not forget this password.", true, 0, utils.MakePasswordList(ctx))

ks := stack.AccountManager().Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore)
account, err := ks.NewAccount(password)
if err != nil {
utils.Fatalf("Failed to create account: %v", err)
}
fmt.Printf("Address: {%x}\n", account.Address)
return nil
}

// accountUpdate transitions an account from a previous format to the current
// one, also providing the possibility to change the pass-phrase.
func accountUpdate(ctx *cli.Context) error {
if len(ctx.Args()) == 0 {
utils.Fatalf("No accounts specified to update")
}
stack, _ := makeConfigNode(ctx)
ks := stack.AccountManager().Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore)

for _, addr := range ctx.Args() {
account, oldPassword := unlockAccount(ctx, ks, addr, 0, nil)
newPassword := getPassPhrase("Please give a new password. Do not forget this password.",
true, 0, nil)
if err := ks.Update(account, oldPassword, newPassword); err != nil {
utils.Fatalf("Could not update the account: %v", err)
}
}
return nil
}

func importWallet(ctx *cli.Context) error {
keyfile := ctx.Args().First()
if len(keyfile) == 0 {
utils.Fatalf("keyfile must be given as argument")
}
```

```
keyJson, err := ioutil.ReadFile(keyfile)
if err != nil {
utils.Fatalf("Could not read wallet file: %v", err)
}

stack, _ := makeConfigNode(ctx)
passphrase := getPassPhrase("", false, 0, utils.MakePasswordList(ctx))

ks := stack.AccountManager().Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore)
acct, err := ks.ImportPreSaleKey(keyJson, passphrase)
if err != nil {
utils.Fatalf("%v", err)
}
fmt.Printf("Address: {%x}\n", acct.Address)
return nil
}

func accountImport(ctx *cli.Context) error {
keyfile := ctx.Args().First()
if len(keyfile) == 0 {
utils.Fatalf("keyfile must be given as argument")
}
key, err := crypto.LoadECDSA(keyfile)
if err != nil {
utils.Fatalf("Failed to load the private key: %v", err)
}
stack, _ := makeConfigNode(ctx)
passphrase := getPassPhrase("Your new account is locked with a password. Please give a
password. Do not forget this password.", true, 0, utils.MakePasswordList(ctx))

ks := stack.AccountManager().Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore)
acct, err := ks.ImportECDSA(key, passphrase)
if err != nil {
utils.Fatalf("Could not create the account: %v", err)
}
fmt.Printf("Address: {%x}\n", acct.Address)
return nil
}

59:F:\git\coin\ethereum\go-ethereum\cmd\geth\accountcmd_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.
```

```
package main

import (
"io/ioutil"
"path/filepath"
"runtime"
"strings"
"testing"

"github.com/cespare/cp"
)

// These tests are 'smoke tests' for the account related
// subcommands and flags.
//
// For most tests, the test files from package accounts
// are copied into a temporary keystore directory.

func tmpDatadirWithKeystore(t *testing.T) string {
datadir := tmpdir(t)
keystore := filepath.Join(datadir, "keystore")
source := filepath.Join("..", "..", "accounts", "keystore", "testdata", "keystore")
if err := cp.CopyAll(keystore, source); err != nil {
t.Fatal(err)
}
return datadir
}

func TestAccountListEmpty(t *testing.T) {
geth := runGeth(t, "account", "list")
geth.ExpectExit()
}

func TestAccountList(t *testing.T) {
datadir := tmpDatadirWithKeystore(t)
geth := runGeth(t, "account", "list", "--datadir", datadir)
defer geth.ExpectExit()
if runtime.GOOS == "windows" {
geth.Expect(`
Account #0: {7ef5a6135f1fd6a02593eedc869c6d41d934aef8}
keystore://{{.Datadir}}\keystore\UTC--2016-03-22T12-57-55.920751759Z--
7ef5a6135f1fd6a02593eedc869c6d41d934aef8
```

```
Account #1: {f466859ead1932d743d622cb74fc058882e8648a} keystore://{{.Datadir}}\keystore\aaa
Account #2: {289d485d9771714cce91d3393d764e1311907acc}
keystore://{{.Datadir}}\keystore\zzz
`)
} else {
geth.Expect(`
Account #0: {7ef5a6135f1fd6a02593eedc869c6d41d934aef8}
keystore://{{.Datadir}}/keystore/UTC--2016-03-22T12-57-55.920751759Z--
7ef5a6135f1fd6a02593eedc869c6d41d934aef8
Account #1: {f466859ead1932d743d622cb74fc058882e8648a} keystore://{{.Datadir}}/keystore/aaa
Account #2: {289d485d9771714cce91d3393d764e1311907acc}
keystore://{{.Datadir}}/keystore/zzz
`)
}
}

func TestAccountNew(t *testing.T) {
geth := runGeth(t, "account", "new", "--lightkdf")
defer geth.ExpectExit()
geth.Expect(`
Your new account is locked with a password. Please give a password. Do not forget this
password.
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "foobar"}}
Repeat passphrase: {{.InputLine "foobar"}}
`)
geth.ExpectRegexp(`Address: \{[0-9a-f]{40}\}\n`)
}

func TestAccountNewBadRepeat(t *testing.T) {
geth := runGeth(t, "account", "new", "--lightkdf")
defer geth.ExpectExit()
geth.Expect(`
Your new account is locked with a password. Please give a password. Do not forget this
password.
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "something"}}
Repeat passphrase: {{.InputLine "something else"}}
Fatal: Passphrases do not match
`)
}
```

```
func TestAccountUpdate(t *testing.T) {
datadir := tmpDatadirWithKeystore(t)
geth := runGeth(t, "account", "update",
"--datadir", datadir, "--lightkdf",
"f466859ead1932d743d622cb74fc058882e8648a")
defer geth.ExpectExit()
geth.Expect(`
Unlocking account f466859ead1932d743d622cb74fc058882e8648a | Attempt 1/3
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "foobar"}}
Please give a new password. Do not forget this password.
Passphrase: {{.InputLine "foobar2"}}
Repeat passphrase: {{.InputLine "foobar2"}}
`)
}

func TestWalletImport(t *testing.T) {
geth := runGeth(t, "wallet", "import", "--lightkdf", "testdata/guswallet.json")
defer geth.ExpectExit()
geth.Expect(`
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "foo"}}
Address: {d4584b5f6229b7be90727b0fc8c6b91bb427821f}
`)

files, err := ioutil.ReadDir(filepath.Join(geth.Datadir, "keystore"))
if len(files) != 1 {
t.Errorf("expected one key file in keystore directory, found %d files (error: %v)", len(files), err)
}
}

func TestWalletImportBadPassword(t *testing.T) {
geth := runGeth(t, "wallet", "import", "--lightkdf", "testdata/guswallet.json")
defer geth.ExpectExit()
geth.Expect(`
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "wrong"}}
Fatal: could not decrypt key with given passphrase
`)
}

func TestUnlockFlag(t *testing.T) {
```

```
datadir := tmpDatadirWithKeystore(t)
geth := runGeth(t,
"--datadir", datadir, "--nat", "none", "--nodiscover", "--dev",
"--unlock", "f466859ead1932d743d622cb74fc058882e8648a",
"js", "testdata/empty.js")
geth.Expect(`
Unlocking account f466859ead1932d743d622cb74fc058882e8648a | Attempt 1/3
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "foobar"}}
`)
geth.ExpectExit()

wantMessages := []string{
"Unlocked account",
"=0xf466859ead1932d743d622cb74fc058882e8648a",
}
for _, m := range wantMessages {
if !strings.Contains(geth.StderrText(), m) {
t.Errorf("stderr text does not contain %q", m)
}
}
}

func TestUnlockFlagWrongPassword(t *testing.T) {
datadir := tmpDatadirWithKeystore(t)
geth := runGeth(t,
"--datadir", datadir, "--nat", "none", "--nodiscover", "--dev",
"--unlock", "f466859ead1932d743d622cb74fc058882e8648a")
defer geth.ExpectExit()
geth.Expect(`
Unlocking account f466859ead1932d743d622cb74fc058882e8648a | Attempt 1/3
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "wrong1"}}
Unlocking account f466859ead1932d743d622cb74fc058882e8648a | Attempt 2/3
Passphrase: {{.InputLine "wrong2"}}
Unlocking account f466859ead1932d743d622cb74fc058882e8648a | Attempt 3/3
Passphrase: {{.InputLine "wrong3"}}
Fatal: Failed to unlock account f466859ead1932d743d622cb74fc058882e8648a (could not
decrypt key with given passphrase)
`)
}
```

```
// https://github.com/ethereum/go-ethereum/issues/1785
func TestUnlockFlagMultiIndex(t *testing.T) {
datadir := tmpDatadirWithKeystore(t)
geth := runGeth(t,
"--datadir", datadir, "--nat", "none", "--nodiscover", "--dev",
"--unlock", "0,2",
"js", "testdata/empty.js")
geth.Expect(`
Unlocking account 0 | Attempt 1/3
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "foobar"}}
Unlocking account 2 | Attempt 1/3
Passphrase: {{.InputLine "foobar"}}
`)
geth.ExpectExit()

wantMessages := []string{
"Unlocked account",
"=0x7ef5a6135f1fd6a02593eedc869c6d41d934aef8",
"=0x289d485d9771714cce91d3393d764e1311907acc",
}
for _, m := range wantMessages {
if !strings.Contains(geth.StderrText(), m) {
t.Errorf("stderr text does not contain %q", m)
}
}
}

func TestUnlockFlagPasswordFile(t *testing.T) {
datadir := tmpDatadirWithKeystore(t)
geth := runGeth(t,
"--datadir", datadir, "--nat", "none", "--nodiscover", "--dev",
"--password", "testdata/passwords.txt", "--unlock", "0,2",
"js", "testdata/empty.js")
geth.ExpectExit()

wantMessages := []string{
"Unlocked account",
"=0x7ef5a6135f1fd6a02593eedc869c6d41d934aef8",
"=0x289d485d9771714cce91d3393d764e1311907acc",
}
for _, m := range wantMessages {
```

```
if !strings.Contains(geth.StderrText(), m) {
t.Errorf("stderr text does not contain %q", m)
}
}
}

func TestUnlockFlagPasswordFileWrongPassword(t *testing.T) {
datadir := tmpDatadirWithKeystore(t)
geth := runGeth(t,
"--datadir", datadir, "--nat", "none", "--nodiscover", "--dev",
"--password", "testdata/wrong-passwords.txt", "--unlock", "0,2")
defer geth.ExpectExit()
geth.Expect(`
Fatal: Failed to unlock account 0 (could not decrypt key with given passphrase)
`)
}

func TestUnlockFlagAmbiguous(t *testing.T) {
store := filepath.Join("..", "..", "accounts", "keystore", "testdata", "dupes")
geth := runGeth(t,
"--keystore", store, "--nat", "none", "--nodiscover", "--dev",
"--unlock", "f466859ead1932d743d622cb74fc058882e8648a",
"js", "testdata/empty.js")
defer geth.ExpectExit()

// Helper for the expect template, returns absolute keystore path.
geth.SetTemplateFunc("keypath", func(file string) string {
abs, _ := filepath.Abs(filepath.Join(store, file))
return abs
})
geth.Expect(`
Unlocking account f466859ead1932d743d622cb74fc058882e8648a | Attempt 1/3
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "foobar"}}
Multiple key files exist for address f466859ead1932d743d622cb74fc058882e8648a:
  keystore://{{keypath "1"}}
  keystore://{{keypath "2"}}
Testing your passphrase against all of them...
Your passphrase unlocked keystore://{{keypath "1"}}
In order to avoid this warning, you need to remove the following duplicate key files:
  keystore://{{keypath "2"}}
`)
```

```
geth.ExpectExit()

wantMessages := []string{
"Unlocked account",
"=0xf466859ead1932d743d622cb74fc058882e8648a",
}
for _, m := range wantMessages {
if !strings.Contains(geth.StderrText(), m) {
t.Errorf("stderr text does not contain %q", m)
}
}
}

func TestUnlockFlagAmbiguousWrongPassword(t *testing.T) {
store := filepath.Join("..", "..", "accounts", "keystore", "testdata", "dupes")
geth := runGeth(t,
"--keystore", store, "--nat", "none", "--nodiscover", "--dev",
"--unlock", "f466859ead1932d743d622cb74fc058882e8648a")
defer geth.ExpectExit()

// Helper for the expect template, returns absolute keystore path.
geth.SetTemplateFunc("keypath", func(file string) string {
abs, _ := filepath.Abs(filepath.Join(store, file))
return abs
})
geth.Expect(`
Unlocking account f466859ead1932d743d622cb74fc058882e8648a | Attempt 1/3
!! Unsupported terminal, password will be echoed.
Passphrase: {{.InputLine "wrong"}}
Multiple key files exist for address f466859ead1932d743d622cb74fc058882e8648a:
   keystore://{{keypath "1"}}
   keystore://{{keypath "2"}}
Testing your passphrase against all of them...
Fatal: None of the listed files could be unlocked.
`)
geth.ExpectExit()
}

60:F:\git\coin\ethereum\go-ethereum\cmd\geth\bugcmd.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main
```

```
import (
"bytes"
"fmt"
"io"
"io/ioutil"
"net/url"
"os/exec"
"runtime"
"strings"

"github.com/ethereum/go-ethereum/cmd/internal/browser"
"github.com/ethereum/go-ethereum/params"

"github.com/ethereum/go-ethereum/cmd/utils"
cli "gopkg.in/urfave/cli.v1"
)

var bugCommand = cli.Command{
Action:    utils.MigrateFlags(reportBug),
Name:      "bug",
Usage:     "opens a window to report a bug on the geth repo",
ArgsUsage: " ",
Category:  "MISCELLANEOUS COMMANDS",
}

const issueUrl = "https://github.com/ethereum/go-ethereum/issues/new"

// reportBug reports a bug by opening a new URL to the go-ethereum GH issue
// tracker and setting default values as the issue body.
func reportBug(ctx *cli.Context) error {
// execute template and write contents to buff
var buff bytes.Buffer

fmt.Fprintln(&buff, header)
fmt.Fprintln(&buff, "Version:", params.Version)
fmt.Fprintln(&buff, "Go Version:", runtime.Version())
fmt.Fprintln(&buff, "OS:", runtime.GOOS)
printOSDetails(&buff)

// open a new GH issue
if !browser.Open(issueUrl + "?body=" + url.QueryEscape(buff.String())) {
```

```
fmt.Printf("Please file a new issue at %s using this template:\n%s", issueUrl, buff.String())
}
return nil
}

// copied from the Go source. Copyright 2017 The Go Authors
func printOSDetails(w io.Writer) {
switch runtime.GOOS {
case "darwin":
printCmdOut(w, "uname -v: ", "uname", "-v")
printCmdOut(w, "", "sw_vers")
case "linux":
printCmdOut(w, "uname -sr: ", "uname", "-sr")
printCmdOut(w, "", "lsb_release", "-a")
case "openbsd", "netbsd", "freebsd", "dragonfly":
printCmdOut(w, "uname -v: ", "uname", "-v")
case "solaris":
out, err := ioutil.ReadFile("/etc/release")
if err == nil {
fmt.Fprintf(w, "/etc/release: %s\n", out)
} else {
fmt.Printf("failed to read /etc/release: %v\n", err)
}
}
}

// printCmdOut prints the output of running the given command.
// It ignores failures; 'go bug' is best effort.
//
// copied from the Go source. Copyright 2017 The Go Authors
func printCmdOut(w io.Writer, prefix, path string, args ...string) {
cmd := exec.Command(path, args...)
out, err := cmd.Output()
if err != nil {
fmt.Printf("%s %s: %v\n", path, strings.Join(args, " "), err)
return
}
fmt.Fprintf(w, "%s%s\n", prefix, bytes.TrimSpace(out))
}

const header = `Please answer these questions before submitting your issue. Thanks!
```

What did you do?

What did you expect to see?

What did you see instead?

System details
`

61:F:\git\coin\ethereum\go-ethereum\cmd\geth\chaincmd.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"encoding/json"
"fmt"
"os"
"runtime"
"strconv"
"sync/atomic"
"time"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/console"
"github.com/ethereum/go-ethereum/core"
"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/trie"
"github.com/syndtr/goleveldb/leveldb/util"
"gopkg.in/urfave/cli.v1"
)

var (
initCommand = cli.Command{
Action: utils.MigrateFlags(initGenesis),
Name: "init",
Usage: "Bootstrap and initialize a new genesis block",
ArgsUsage: "<genesisPath>",

```
Flags: []cli.Flag{
utils.DataDirFlag,
utils.LightModeFlag,
},
Category: "BLOCKCHAIN COMMANDS",
Description: `
The init command initializes a new genesis block and definition for the network.
This is a destructive action and changes the network in which you will be
participating.

It expects the genesis file as argument.`,
}
importCommand = cli.Command{
Action:    utils.MigrateFlags(importChain),
Name:      "import",
Usage:     "Import a blockchain file",
ArgsUsage: "<filename> (<filename 2> ... <filename N>) ",
Flags: []cli.Flag{
utils.DataDirFlag,
utils.CacheFlag,
utils.LightModeFlag,
},
Category: "BLOCKCHAIN COMMANDS",
Description: `
The import command imports blocks from an RLP-encoded form. The form can be one file
with several RLP-encoded blocks, or several files can be used.

If only one file is used, import error will result in failure. If several files are used,
processing will proceed even if an individual RLP-file import failure occurs.`,
}
exportCommand = cli.Command{
Action:    utils.MigrateFlags(exportChain),
Name:      "export",
Usage:     "Export blockchain into file",
ArgsUsage: "<filename> [<blockNumFirst> <blockNumLast>]",
Flags: []cli.Flag{
utils.DataDirFlag,
utils.CacheFlag,
utils.LightModeFlag,
},
Category: "BLOCKCHAIN COMMANDS",
Description: `
```

```
Requires a first argument of the file to write to.
Optional second and third arguments control the first and
last block to write. In this mode, the file will be appended
if already existing.`,
}
removedbCommand = cli.Command{
Action:    utils.MigrateFlags(removeDB),
Name:      "removedb",
Usage:     "Remove blockchain and state databases",
ArgsUsage: " ",
Flags: []cli.Flag{
utils.DataDirFlag,
utils.LightModeFlag,
},
Category: "BLOCKCHAIN COMMANDS",
Description: `
Remove blockchain and state databases`,
}
dumpCommand = cli.Command{
Action:    utils.MigrateFlags(dump),
Name:      "dump",
Usage:     "Dump a specific block from storage",
ArgsUsage: "[<blockHash> | <blockNum>]...",
Flags: []cli.Flag{
utils.DataDirFlag,
utils.CacheFlag,
utils.LightModeFlag,
},
Category: "BLOCKCHAIN COMMANDS",
Description: `
The arguments are interpreted as block numbers or hashes.
Use "ethereum dump 0" to dump the genesis block.`,
}
)

// initGenesis will initialise the given JSON format genesis file and writes it as
// the zero'd block (i.e. genesis) or will fail hard if it can't succeed.
func initGenesis(ctx *cli.Context) error {
// Make sure we have a valid genesis JSON
genesisPath := ctx.Args().First()
if len(genesisPath) == 0 {
utils.Fatalf("Must supply path to genesis JSON file")
```

```
}
file, err := os.Open(genesisPath)
if err != nil {
utils.Fatalf("Failed to read genesis file: %v", err)
}
defer file.Close()

genesis := new(core.Genesis)
if err := json.NewDecoder(file).Decode(genesis); err != nil {
utils.Fatalf("invalid genesis file: %v", err)
}
// Open an initialise both full and light databases
stack := makeFullNode(ctx)
for _, name := range []string{"chaindata", "lightchaindata"} {
chaindb, err := stack.OpenDatabase(name, 0, 0)
if err != nil {
utils.Fatalf("Failed to open database: %v", err)
}
_, hash, err := core.SetupGenesisBlock(chaindb, genesis)
if err != nil {
utils.Fatalf("Failed to write genesis block: %v", err)
}
log.Info("Successfully wrote genesis state", "database", name, "hash", hash)
}
return nil
}

func importChain(ctx *cli.Context) error {
if len(ctx.Args()) < 1 {
utils.Fatalf("This command requires an argument.")
}
stack := makeFullNode(ctx)
chain, chainDb := utils.MakeChain(ctx, stack)
defer chainDb.Close()

// Start periodically gathering memory profiles
var peakMemAlloc, peakMemSys uint64
go func() {
stats := new(runtime.MemStats)
for {
runtime.ReadMemStats(stats)
if atomic.LoadUint64(&peakMemAlloc) < stats.Alloc {
```

```
atomic.StoreUint64(&peakMemAlloc, stats.Alloc)
}
if atomic.LoadUint64(&peakMemSys) < stats.Sys {
atomic.StoreUint64(&peakMemSys, stats.Sys)
}
time.Sleep(5 * time.Second)
}
}()
// Import the chain
start := time.Now()

if len(ctx.Args()) == 1 {
if err := utils.ImportChain(chain, ctx.Args().First()); err != nil {
utils.Fatalf("Import error: %v", err)
}
} else {
for _, arg := range ctx.Args() {
if err := utils.ImportChain(chain, arg); err != nil {
log.Error("Import error", "file", arg, "err", err)
}
}
}

fmt.Printf("Import done in %v.\n\n", time.Since(start))

// Output pre-compaction stats mostly to see the import trashing
db := chainDb.(*ethdb.LDBDatabase)

stats, err := db.LDB().GetProperty("leveldb.stats")
if err != nil {
utils.Fatalf("Failed to read database stats: %v", err)
}
fmt.Println(stats)
fmt.Printf("Trie cache misses:  %d\n", trie.CacheMisses())
fmt.Printf("Trie cache unloads: %d\n\n", trie.CacheUnloads())

// Print the memory statistics used by the importing
mem := new(runtime.MemStats)
runtime.ReadMemStats(mem)

fmt.Printf("Object memory: %.3f MB current, %.3f MB peak\n", float64(mem.Alloc)/1024/1024,
float64(atomic.LoadUint64(&peakMemAlloc))/1024/1024)
```

```
fmt.Printf("System memory: %.3f MB current, %.3f MB peak\n", float64(mem.Sys)/1024/1024,
float64(atomic.LoadUint64(&peakMemSys))/1024/1024)
fmt.Printf("Allocations:   %.3f million\n", float64(mem.Mallocs)/1000000)
fmt.Printf("GC pause:      %v\n\n", time.Duration(mem.PauseTotalNs))

if ctx.GlobalIsSet(utils.NoCompactionFlag.Name) {
return nil
}

// Compact the entire database to more accurately measure disk io and print the stats
start = time.Now()
fmt.Println("Compacting entire database...")
if err = db.LDB().CompactRange(util.Range{}); err != nil {
utils.Fatalf("Compaction failed: %v", err)
}
fmt.Printf("Compaction done in %v.\n\n", time.Since(start))

stats, err = db.LDB().GetProperty("leveldb.stats")
if err != nil {
utils.Fatalf("Failed to read database stats: %v", err)
}
fmt.Println(stats)

return nil
}

func exportChain(ctx *cli.Context) error {
if len(ctx.Args()) < 1 {
utils.Fatalf("This command requires an argument.")
}
stack := makeFullNode(ctx)
chain, _ := utils.MakeChain(ctx, stack)
start := time.Now()

var err error
fp := ctx.Args().First()
if len(ctx.Args()) < 3 {
err = utils.ExportChain(chain, fp)
} else {
// This can be improved to allow for numbers larger than 9223372036854775807
first, ferr := strconv.ParseInt(ctx.Args().Get(1), 10, 64)
last, lerr := strconv.ParseInt(ctx.Args().Get(2), 10, 64)
```

```
if ferr != nil || lerr != nil {
utils.Fatalf("Export error in parsing parameters: block number not an integer\n")
}
if first < 0 || last < 0 {
utils.Fatalf("Export error: block number must be greater than 0\n")
}
err = utils.ExportAppendChain(chain, fp, uint64(first), uint64(last))
}

if err != nil {
utils.Fatalf("Export error: %v\n", err)
}
fmt.Printf("Export done in %v", time.Since(start))
return nil
}

func removeDB(ctx *cli.Context) error {
stack, _ := makeConfigNode(ctx)

for _, name := range []string{"chaindata", "lightchaindata"} {
// Ensure the database exists in the first place
logger := log.New("database", name)

dbdir := stack.ResolvePath(name)
if !common.FileExist(dbdir) {
logger.Info("Database doesn't exist, skipping", "path", dbdir)
continue
}
// Confirm removal and execute
fmt.Println(dbdir)
confirm, err := console.Stdin.PromptConfirm("Remove this database?")
switch {
case err != nil:
utils.Fatalf("%v", err)
case !confirm:
logger.Warn("Database deletion aborted")
default:
start := time.Now()
os.RemoveAll(dbdir)
logger.Info("Database successfully deleted", "elapsed", common.PrettyDuration(time.Since(start)))
}
}
```

```
return nil
}

func dump(ctx *cli.Context) error {
stack := makeFullNode(ctx)
chain, chainDb := utils.MakeChain(ctx, stack)
for _, arg := range ctx.Args() {
var block *types.Block
if hashish(arg) {
block = chain.GetBlockByHash(common.HexToHash(arg))
} else {
num, _ := strconv.Atoi(arg)
block = chain.GetBlockByNumber(uint64(num))
}
if block == nil {
fmt.Println("{}")
utils.Fatalf("block not found")
} else {
state, err := state.New(block.Root(), state.NewDatabase(chainDb))
if err != nil {
utils.Fatalf("could not create new state: %v", err)
}
fmt.Printf("%s\n", state.Dump())
}
}
chainDb.Close()
return nil
}

// hashish returns true for strings that look like hashes.
func hashish(x string) bool {
_, err := strconv.Atoi(x)
return err != nil
}

62:F:\git\coin\ethereum\go-ethereum\cmd\geth\config.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bufio"
```

```
"encoding/hex"
"errors"
"fmt"
"io"
"os"
"reflect"
"unicode"

cli "gopkg.in/urfave/cli.v1"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/contracts/release"
"github.com/ethereum/go-ethereum/eth"
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/params"
whisper "github.com/ethereum/go-ethereum/whisper/whisperv5"
"github.com/naoina/toml"
)

var (
dumpConfigCommand = cli.Command{
Action:      utils.MigrateFlags(dumpConfig),
Name:        "dumpconfig",
Usage:       "Show configuration values",
ArgsUsage:   "",
Flags:       append(append(nodeFlags, rpcFlags...), whisperFlags...),
Category:    "MISCELLANEOUS COMMANDS",
Description: `The dumpconfig command shows configuration values.`,
}

configFileFlag = cli.StringFlag{
Name:  "config",
Usage: "TOML configuration file",
}
)

// These settings ensure that TOML keys use the same names as Go struct fields.
var tomlSettings = toml.Config{
NormFieldName: func(rt reflect.Type, key string) string {
return key
},
FieldToKey: func(rt reflect.Type, field string) string {
```

```
return field
},
MissingField: func(rt reflect.Type, field string) error {
link := ""
if unicode.IsUpper(rune(rt.Name()[0])) && rt.PkgPath() != "main" {
link = fmt.Sprintf(", see https://godoc.org/%s#%s for available fields", rt.PkgPath(), rt.Name())
}
return fmt.Errorf("field '%s' is not defined in %s%s", field, rt.String(), link)
},
}

type ethstatsConfig struct {
URL string `toml:",omitempty"`
}

type gethConfig struct {
Eth      eth.Config
Shh      whisper.Config
Node     node.Config
Ethstats ethstatsConfig
}

func loadConfig(file string, cfg *gethConfig) error {
f, err := os.Open(file)
if err != nil {
return err
}
defer f.Close()

err = tomlSettings.NewDecoder(bufio.NewReader(f)).Decode(cfg)
// Add file name to errors that have a line number.
if _, ok := err.(*toml.LineError); ok {
err = errors.New(file + ", " + err.Error())
}
return err
}

func defaultNodeConfig() node.Config {
cfg := node.DefaultConfig
cfg.Name = clientIdentifier
cfg.Version = params.VersionWithCommit(gitCommit)
cfg.HTTPModules = append(cfg.HTTPModules, "eth", "shh")
```

```
cfg.WSModules = append(cfg.WSModules, "eth", "shh")
cfg.IPCPath = "geth.ipc"
return cfg
}

func makeConfigNode(ctx *cli.Context) (*node.Node, gethConfig) {
// Load defaults.
cfg := gethConfig{
Eth:  eth.DefaultConfig,
Shh:  whisper.DefaultConfig,
Node: defaultNodeConfig(),
}

// Load config file.
if file := ctx.GlobalString(configFileFlag.Name); file != "" {
if err := loadConfig(file, &cfg); err != nil {
utils.Fatalf("%v", err)
}
}

// Apply flags.
utils.SetNodeConfig(ctx, &cfg.Node)
stack, err := node.New(&cfg.Node)
if err != nil {
utils.Fatalf("Failed to create the protocol stack: %v", err)
}
utils.SetEthConfig(ctx, stack, &cfg.Eth)
if ctx.GlobalIsSet(utils.EthStatsURLFlag.Name) {
cfg.Ethstats.URL = ctx.GlobalString(utils.EthStatsURLFlag.Name)
}

utils.SetShhConfig(ctx, stack, &cfg.Shh)

return stack, cfg
}

// enableWhisper returns true in case one of the whisper flags is set.
func enableWhisper(ctx *cli.Context) bool {
for _, flag := range whisperFlags {
if ctx.GlobalIsSet(flag.GetName()) {
return true
}
```

```
}
return false
}

func makeFullNode(ctx *cli.Context) *node.Node {
stack, cfg := makeConfigNode(ctx)

utils.RegisterEthService(stack, &cfg.Eth)

// Whisper must be explicitly enabled by specifying at least 1 whisper flag or in dev mode
shhEnabled := enableWhisper(ctx)
shhAutoEnabled := !ctx.GlobalIsSet(utils.WhisperEnabledFlag.Name) &&
ctx.GlobalIsSet(utils.DevModeFlag.Name)
if shhEnabled || shhAutoEnabled {
if ctx.GlobalIsSet(utils.WhisperMaxMessageSizeFlag.Name) {
cfg.Shh.MaxMessageSize = uint32(ctx.Int(utils.WhisperMaxMessageSizeFlag.Name))
}
if ctx.GlobalIsSet(utils.WhisperMinPOWFlag.Name) {
cfg.Shh.MinimumAcceptedPOW = ctx.Float64(utils.WhisperMinPOWFlag.Name)
}
utils.RegisterShhService(stack, &cfg.Shh)
}

// Add the Ethereum Stats daemon if requested.
if cfg.Ethstats.URL != "" {
utils.RegisterEthStatsService(stack, cfg.Ethstats.URL)
}

// Add the release oracle service so it boots along with node.
if err := stack.Register(func(ctx *node.ServiceContext) (node.Service, error) {
config := release.Config{
Oracle: relOracle,
Major:  uint32(params.VersionMajor),
Minor:  uint32(params.VersionMinor),
Patch:  uint32(params.VersionPatch),
}
commit, _ := hex.DecodeString(gitCommit)
copy(config.Commit[:], commit)
return release.NewReleaseService(ctx, config)
}); err != nil {
utils.Fatalf("Failed to register the Geth release oracle service: %v", err)
}
```

```
return stack
}

// dumpConfig is the dumpconfig command.
func dumpConfig(ctx *cli.Context) error {
_, cfg := makeConfigNode(ctx)
comment := ""

if cfg.Eth.Genesis != nil {
cfg.Eth.Genesis = nil
comment += "# Note: this config doesn't contain the genesis block.\n\n"
}

out, err := tomlSettings.Marshal(&cfg)
if err != nil {
return err
}
io.WriteString(os.Stdout, comment)
os.Stdout.Write(out)
return nil
}

63:F:\git\coin\ethereum\go-ethereum\cmd\geth\consolecmd.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"os"
"os/signal"
"strings"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/console"
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/rpc"
"gopkg.in/urfave/cli.v1"
)

var (
consoleFlags = []cli.Flag{utils.JSpathFlag, utils.ExecFlag, utils.PreloadJSFlag}
```

```
consoleCommand = cli.Command{
Action:   utils.MigrateFlags(localConsole),
Name:     "console",
Usage:    "Start an interactive JavaScript environment",
Flags:    append(append(append(nodeFlags, rpcFlags...), consoleFlags...), whisperFlags...),
Category: "CONSOLE COMMANDS",
Description: `
The Geth console is an interactive shell for the JavaScript runtime environment
which exposes a node admin interface as well as the Đapp JavaScript API.
See https://github.com/ethereum/go-ethereum/wiki/Javascipt-Console.`,
}

attachCommand = cli.Command{
Action:    utils.MigrateFlags(remoteConsole),
Name:      "attach",
Usage:     "Start an interactive JavaScript environment (connect to node)",
ArgsUsage: "[endpoint]",
Flags:     append(consoleFlags, utils.DataDirFlag),
Category:  "CONSOLE COMMANDS",
Description: `
The Geth console is an interactive shell for the JavaScript runtime environment
which exposes a node admin interface as well as the Đapp JavaScript API.
See https://github.com/ethereum/go-ethereum/wiki/Javascipt-Console.
This command allows to open a console on a running geth node.`,
}

javascriptCommand = cli.Command{
Action:    utils.MigrateFlags(ephemeralConsole),
Name:      "js",
Usage:     "Execute the specified JavaScript files",
ArgsUsage: "<jsfile> [jsfile...]",
Flags:     append(nodeFlags, consoleFlags...),
Category:  "CONSOLE COMMANDS",
Description: `
The JavaScript VM exposes a node admin interface as well as the Đapp
JavaScript API. See https://github.com/ethereum/go-ethereum/wiki/Javascipt-Console`,
}
)

// localConsole starts a new geth node, attaching a JavaScript console to it at the
// same time.
func localConsole(ctx *cli.Context) error {
```

```
// Create and start the node based on the CLI flags
node := makeFullNode(ctx)
startNode(ctx, node)
defer node.Stop()

// Attach to the newly started node and start the JavaScript console
client, err := node.Attach()
if err != nil {
utils.Fatalf("Failed to attach to the inproc geth: %v", err)
}
config := console.Config{
DataDir: utils.MakeDataDir(ctx),
DocRoot: ctx.GlobalString(utils.JSpathFlag.Name),
Client:  client,
Preload: utils.MakeConsolePreloads(ctx),
}

console, err := console.New(config)
if err != nil {
utils.Fatalf("Failed to start the JavaScript console: %v", err)
}
defer console.Stop(false)

// If only a short execution was requested, evaluate and return
if script := ctx.GlobalString(utils.ExecFlag.Name); script != "" {
console.Evaluate(script)
return nil
}
// Otherwise print the welcome screen and enter interactive mode
console.Welcome()
console.Interactive()

return nil
}

// remoteConsole will connect to a remote geth instance, attaching a JavaScript
// console to it.
func remoteConsole(ctx *cli.Context) error {
// Attach to a remotely running geth instance and start the JavaScript console
client, err := dialRPC(ctx.Args().First())
if err != nil {
utils.Fatalf("Unable to attach to remote geth: %v", err)
```

```
}
config := console.Config{
DataDir: utils.MakeDataDir(ctx),
DocRoot: ctx.GlobalString(utils.JSpathFlag.Name),
Client:  client,
Preload: utils.MakeConsolePreloads(ctx),
}

console, err := console.New(config)
if err != nil {
utils.Fatalf("Failed to start the JavaScript console: %v", err)
}
defer console.Stop(false)

if script := ctx.GlobalString(utils.ExecFlag.Name); script != "" {
console.Evaluate(script)
return nil
}

// Otherwise print the welcome screen and enter interactive mode
console.Welcome()
console.Interactive()

return nil
}

// dialRPC returns a RPC client which connects to the given endpoint.
// The check for empty endpoint implements the defaulting logic
// for "geth attach" and "geth monitor" with no argument.
func dialRPC(endpoint string) (*rpc.Client, error) {
if endpoint == "" {
endpoint = node.DefaultIPCEndpoint(clientIdentifier)
} else if strings.HasPrefix(endpoint, "rpc:") || strings.HasPrefix(endpoint, "ipc:") {
// Backwards compatibility with geth < 1.5 which required
// these prefixes.
endpoint = endpoint[4:]
}
return rpc.Dial(endpoint)
}

// ephemeralConsole starts a new geth node, attaches an ephemeral JavaScript
// console to it, executes each of the files specified as arguments and tears
```

```
// everything down.
func ephemeralConsole(ctx *cli.Context) error {
// Create and start the node based on the CLI flags
node := makeFullNode(ctx)
startNode(ctx, node)
defer node.Stop()

// Attach to the newly started node and start the JavaScript console
client, err := node.Attach()
if err != nil {
utils.Fatalf("Failed to attach to the inproc geth: %v", err)
}
config := console.Config{
DataDir: utils.MakeDataDir(ctx),
DocRoot: ctx.GlobalString(utils.JSpathFlag.Name),
Client:  client,
Preload: utils.MakeConsolePreloads(ctx),
}

console, err := console.New(config)
if err != nil {
utils.Fatalf("Failed to start the JavaScript console: %v", err)
}
defer console.Stop(false)

// Evaluate each of the specified JavaScript files
for _, file := range ctx.Args() {
if err = console.Execute(file); err != nil {
utils.Fatalf("Failed to execute %s: %v", file, err)
}
}
// Wait for pending callbacks, but stop for Ctrl-C.
abort := make(chan os.Signal, 1)
signal.Notify(abort, os.Interrupt)

go func() {
<-abort
os.Exit(0)
}()
console.Stop(true)

return nil
```

```
}

64:F:\git\coin\ethereum\go-ethereum\cmd\geth\consolecmd_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"crypto/rand"
"math/big"
"os"
"path/filepath"
"runtime"
"strconv"
"strings"
"testing"
"time"

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

const (
ipcAPIs  = "admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 shh:1.0 txpool:1.0
web3:1.0"
httpAPIs = "eth:1.0 net:1.0 rpc:1.0 web3:1.0"
)

// Tests that a node embedded within a console can be started up properly and
// then terminated by closing the input stream.
func TestConsoleWelcome(t *testing.T) {
coinbase := "0x8605cdbbdb6d264aa742e77020dcbc58fcdce182"

// Start a geth console, make sure it's cleaned up and terminate the console
geth := runGeth(t,
"--port", "0", "--maxpeers", "0", "--nodiscover", "--nat", "none",
"--etherbase", coinbase, "--shh",
"console")

// Gather all the infos the welcome message needs to contain
geth.SetTemplateFunc("goos", func() string { return runtime.GOOS })
geth.SetTemplateFunc("goarch", func() string { return runtime.GOARCH })
geth.SetTemplateFunc("gover", runtime.Version)
```

```
geth.SetTemplateFunc("gethver", func() string { return params.Version })
geth.SetTemplateFunc("niltime", func() string { return time.Unix(0, 0).Format(time.RFC1123) })
geth.SetTemplateFunc("apis", func() string { return ipcAPIs })

// Verify the actual welcome message to the required template
geth.Expect(`
Welcome to the Geth JavaScript console!

instance: Geth/v{{gethver}}/{{goos}}-{{goarch}}/{{gover}}
coinbase: {{.Etherbase}}
at block: 0 ({{niltime}})
 datadir: {{.Datadir}}
 modules: {{apis}}

> {{.InputLine "exit"}}
`)
geth.ExpectExit()
}

// Tests that a console can be attached to a running node via various means.
func TestIPCAttachWelcome(t *testing.T) {
// Configure the instance for IPC attachement
coinbase := "0x8605cdbbdb6d264aa742e77020dcbc58fcdce182"
var ipc string
if runtime.GOOS == "windows" {
ipc = `\\.\pipe\geth` + strconv.Itoa(trulyRandInt(100000, 999999))
} else {
ws := tmpdir(t)
defer os.RemoveAll(ws)
ipc = filepath.Join(ws, "geth.ipc")
}
// Note: we need --shh because testAttachWelcome checks for default
// list of ipc modules and shh is included there.
geth := runGeth(t,
"--port", "0", "--maxpeers", "0", "--nodiscover", "--nat", "none",
"--etherbase", coinbase, "--shh", "--ipcpath", ipc)

time.Sleep(2 * time.Second) // Simple way to wait for the RPC endpoint to open
testAttachWelcome(t, geth, "ipc:"+ipc, ipcAPIs)

geth.Interrupt()
geth.ExpectExit()
```

```
}

func TestHTTPAttachWelcome(t *testing.T) {
coinbase := "0x8605cdbbdb6d264aa742e77020dcbc58fcdce182"
port := strconv.Itoa(trulyRandInt(1024, 65536)) // Yeah, sometimes this will fail, sorry :P
geth := runGeth(t,
"--port", "0", "--maxpeers", "0", "--nodiscover", "--nat", "none",
"--etherbase", coinbase, "--rpc", "--rpcport", port)

time.Sleep(2 * time.Second) // Simple way to wait for the RPC endpoint to open
testAttachWelcome(t, geth, "http://localhost:"+port, httpAPIs)

geth.Interrupt()
geth.ExpectExit()
}

func TestWSAttachWelcome(t *testing.T) {
coinbase := "0x8605cdbbdb6d264aa742e77020dcbc58fcdce182"
port := strconv.Itoa(trulyRandInt(1024, 65536)) // Yeah, sometimes this will fail, sorry :P

geth := runGeth(t,
"--port", "0", "--maxpeers", "0", "--nodiscover", "--nat", "none",
"--etherbase", coinbase, "--ws", "--wsport", port)

time.Sleep(2 * time.Second) // Simple way to wait for the RPC endpoint to open
testAttachWelcome(t, geth, "ws://localhost:"+port, httpAPIs)

geth.Interrupt()
geth.ExpectExit()
}

func testAttachWelcome(t *testing.T, geth *testgeth, endpoint, apis string) {
// Attach to a running geth note and terminate immediately
attach := runGeth(t, "attach", endpoint)
defer attach.ExpectExit()
attach.CloseStdin()

// Gather all the infos the welcome message needs to contain
attach.SetTemplateFunc("goos", func() string { return runtime.GOOS })
attach.SetTemplateFunc("goarch", func() string { return runtime.GOARCH })
attach.SetTemplateFunc("gover", runtime.Version)
attach.SetTemplateFunc("gethver", func() string { return params.Version })
```

```
attach.SetTemplateFunc("etherbase", func() string { return geth.Etherbase })
attach.SetTemplateFunc("niltime", func() string { return time.Unix(0, 0).Format(time.RFC1123) })
attach.SetTemplateFunc("ipc", func() bool { return strings.HasPrefix(endpoint, "ipc") })
attach.SetTemplateFunc("datadir", func() string { return geth.Datadir })
attach.SetTemplateFunc("apis", func() string { return apis })

// Verify the actual welcome message to the required template
attach.Expect(`
Welcome to the Geth JavaScript console!

instance: Geth/v{{gethver}}/{{goos}}-{{goarch}}/{{gover}}
coinbase: {{etherbase}}
at block: 0 ({{niltime}}){{if ipc}}
 datadir: {{datadir}}{{end}}
 modules: {{apis}}

> {{.InputLine "exit" }}
`)
attach.ExpectExit()
}

// trulyRandInt generates a crypto random integer used by the console tests to
// not clash network ports with other tests running cocurrently.
func trulyRandInt(lo, hi int) int {
num, _ := rand.Int(rand.Reader, big.NewInt(int64(hi-lo)))
return int(num.Int64()) + lo
}

65:F:\git\coin\ethereum\go-ethereum\cmd\geth\dao_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"io/ioutil"
"math/big"
"os"
"path/filepath"
"testing"

"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core"
```

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

// Genesis block for nodes which don't care about the DAO fork (i.e. not configured)
var daoOldGenesis = `{
"alloc"      : {},
"coinbase"   : "0x0000000000000000000000000000000000000000",
"difficulty" : "0x20000",
"extraData"  : "",
"gasLimit"   : "0x2fefd8",
"nonce"      : "0x0000000000000042",
"mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp"  : "0x00",
"config"     : {}
}`

// Genesis block for nodes which actively oppose the DAO fork
var daoNoForkGenesis = `{
"alloc"      : {},
"coinbase"   : "0x0000000000000000000000000000000000000000",
"difficulty" : "0x20000",
"extraData"  : "",
"gasLimit"   : "0x2fefd8",
"nonce"      : "0x0000000000000042",
"mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp"  : "0x00",
"config"     : {
"daoForkBlock"   : 314,
"daoForkSupport" : false
}
}`

// Genesis block for nodes which actively support the DAO fork
var daoProForkGenesis = `{
"alloc"      : {},
```

```
"coinbase"   : "0x0000000000000000000000000000000000000000",
"difficulty" : "0x20000",
"extraData"  : "",
"gasLimit"   : "0x2fefd8",
"nonce"      : "0x0000000000000042",
"mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp"  : "0x00",
"config"     : {
"daoForkBlock"   : 314,
"daoForkSupport" : true
}
}`

var daoGenesisHash =
common.HexToHash("5e1fc79cb4ffa4739177b5408045cd5d51c6cf766133f23f7cd72ee1f8d790e0
")
var daoGenesisForkBlock = big.NewInt(314)

// TestDAOForkBlockNewChain tests that the DAO hard-fork number and the nodes
support/opposition is correctly
// set in the database after various initialization procedures and invocations.
func TestDAOForkBlockNewChain(t *testing.T) {
for i, arg := range []struct {
genesis     string
expectBlock *big.Int
expectVote  bool
}{
// Test DAO Default Mainnet
{"", params.MainnetChainConfig.DAOForkBlock, true},
// test DAO Init Old Privnet
{daoOldGenesis, nil, false},
// test DAO Default No Fork Privnet
{daoNoForkGenesis, daoGenesisForkBlock, false},
// test DAO Default Pro Fork Privnet
{daoProForkGenesis, daoGenesisForkBlock, true},
} {
testDAOForkBlockNewChain(t, i, arg.genesis, arg.expectBlock, arg.expectVote)
}
}
```

```
func testDAOForkBlockNewChain(t *testing.T, test int, genesis string, expectBlock *big.Int,
expectVote bool) {
// Create a temporary data directory to use and inspect later
datadir := tmpdir(t)
defer os.RemoveAll(datadir)

// Start a Geth instance with the requested flags set and immediately terminate
if genesis != "" {
json := filepath.Join(datadir, "genesis.json")
if err := ioutil.WriteFile(json, []byte(genesis), 0600); err != nil {
t.Fatalf("test %d: failed to write genesis file: %v", test, err)
}
runGeth(t, "--datadir", datadir, "init", json).WaitExit()
} else {
// Force chain initialization
args := []string{"--port", "0", "--maxpeers", "0", "--nodiscover", "--nat", "none", "--ipcdisable", "--
datadir", datadir}
geth := runGeth(t, append(args, []string{"--exec", "2+2", "console"}...)...)
geth.WaitExit()
}
// Retrieve the DAO config flag from the database
path := filepath.Join(datadir, "geth", "chaindata")
db, err := ethdb.NewLDBDatabase(path, 0, 0)
if err != nil {
t.Fatalf("test %d: failed to open test database: %v", test, err)
}
defer db.Close()

genesisHash :=
common.HexToHash("0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb
8fa3")
if genesis != "" {
genesisHash = daoGenesisHash
}
config, err := core.GetChainConfig(db, genesisHash)
if err != nil {
t.Errorf("test %d: failed to retrieve chain config: %v", test, err)
return // we want to return here, the other checks can't make it past this point (nil panic).
}
// Validate the DAO hard-fork block number against the expected value
if config.DAOForkBlock == nil {
```

```
if expectBlock != nil {
t.Errorf("test %d: dao hard-fork block mismatch: have nil, want %v", test, expectBlock)
}
} else if expectBlock == nil {
t.Errorf("test %d: dao hard-fork block mismatch: have %v, want nil", test, config.DAOForkBlock)
} else if config.DAOForkBlock.Cmp(expectBlock) != 0 {
t.Errorf("test %d: dao hard-fork block mismatch: have %v, want %v", test, config.DAOForkBlock,
expectBlock)
}
if config.DAOForkSupport != expectVote {
t.Errorf("test %d: dao hard-fork support mismatch: have %v, want %v", test,
config.DAOForkSupport, expectVote)
}
}
```

66:F:\git\coin\ethereum\go-ethereum\cmd\geth\genesis_test.go

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

package main

import (
"io/ioutil"
"os"
"path/filepath"
"testing"
)

var customGenesisTests = []struct {
genesis string
query   string
result  string
}{
// Plain genesis file without anything extra
{
genesis: `{
"alloc"      : {},
"coinbase"   : "0x0000000000000000000000000000000000000000",
"difficulty" : "0x20000",
"extraData"  : "",
"gasLimit"   : "0x2fefd8",
"nonce"      : "0x0000000000000042",
"mixhash"    :
```

```
"0x0000000000000000000000000000000000000000000000000000000000000000",
"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp"  : "0x00"
}`,
query:  "eth.getBlock(0).nonce",
result: "0x0000000000000042",
},
// Genesis file with an empty chain configuration (ensure missing fields work)
{
genesis: `{
"alloc"      : {},
"coinbase"   : "0x0000000000000000000000000000000000000000",
"difficulty" : "0x20000",
"extraData"  : "",
"gasLimit"   : "0x2fefd8",
"nonce"      : "0x0000000000000042",
"mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp"  : "0x00",
"config"     : {}
}`,
query:  "eth.getBlock(0).nonce",
result: "0x0000000000000042",
},
// Genesis file with specific chain configurations
{
genesis: `{
"alloc"      : {},
"coinbase"   : "0x0000000000000000000000000000000000000000",
"difficulty" : "0x20000",
"extraData"  : "",
"gasLimit"   : "0x2fefd8",
"nonce"      : "0x0000000000000042",
"mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp"  : "0x00",
"config"     : {
```

```
"homesteadBlock" : 314,
"daoForkBlock"   : 141,
"daoForkSupport" : true
},
}`,
query:  "eth.getBlock(0).nonce",
result: "0x0000000000000042",
},
}

// Tests that initializing Geth with a custom genesis block and chain definitions
// work properly.
func TestCustomGenesis(t *testing.T) {
for i, tt := range customGenesisTests {
// Create a temporary data directory to use and inspect later
datadir := tmpdir(t)
defer os.RemoveAll(datadir)

// Initialize the data directory with the custom genesis block
json := filepath.Join(datadir, "genesis.json")
if err := ioutil.WriteFile(json, []byte(tt.genesis), 0600); err != nil {
t.Fatalf("test %d: failed to write genesis file: %v", i, err)
}
runGeth(t, "--datadir", datadir, "init", json).WaitExit()

// Query the custom genesis block
geth := runGeth(t,
"--datadir", datadir, "--maxpeers", "0", "--port", "0",
"--nodiscover", "--nat", "none", "--ipcdisable",
"--exec", tt.query, "console")
geth.ExpectRegexp(tt.result)
geth.ExpectExit()
}
}

67:F:\git\coin\ethereum\go-ethereum\cmd\geth\main.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// geth is the official command-line client for Ethereum.
package main

import (
```

```
"fmt"
"os"
"runtime"
"strings"
"time"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/console"
"github.com/ethereum/go-ethereum/eth"
"github.com/ethereum/go-ethereum/ethclient"
"github.com/ethereum/go-ethereum/internal/debug"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/metrics"
"github.com/ethereum/go-ethereum/node"
"gopkg.in/urfave/cli.v1"
)

const (
clientIdentifier = "geth" // Client identifier to advertise over the network
)

var (
// Git SHA1 commit hash of the release (set via linker flags)
gitCommit = ""
// Ethereum address of the Geth release oracle.
relOracle = common.HexToAddress("0xfa7b9770ca4cb04296cac84f37736d4041251cdf")
// The app that holds all commands and flags.
app = utils.NewApp(gitCommit, "the go-ethereum command line interface")
// flags that configure the node
nodeFlags = []cli.Flag{
utils.IdentityFlag,
utils.UnlockedAccountFlag,
utils.PasswordFileFlag,
utils.BootnodesFlag,
utils.BootnodesV4Flag,
utils.BootnodesV5Flag,
utils.DataDirFlag,
utils.KeyStoreDirFlag,
utils.NoUSBFlag,
```

utils.EthashCacheDirFlag,
utils.EthashCachesInMemoryFlag,
utils.EthashCachesOnDiskFlag,
utils.EthashDatasetDirFlag,
utils.EthashDatasetsInMemoryFlag,
utils.EthashDatasetsOnDiskFlag,
utils.TxPoolPriceLimitFlag,
utils.TxPoolPriceBumpFlag,
utils.TxPoolAccountSlotsFlag,
utils.TxPoolGlobalSlotsFlag,
utils.TxPoolAccountQueueFlag,
utils.TxPoolGlobalQueueFlag,
utils.TxPoolLifetimeFlag,
utils.FastSyncFlag,
utils.LightModeFlag,
utils.SyncModeFlag,
utils.LightServFlag,
utils.LightPeersFlag,
utils.LightKDFFlag,
utils.CacheFlag,
utils.TrieCacheGenFlag,
utils.ListenPortFlag,
utils.MaxPeersFlag,
utils.MaxPendingPeersFlag,
utils.EtherbaseFlag,
utils.GasPriceFlag,
utils.MinerThreadsFlag,
utils.MiningEnabledFlag,
utils.TargetGasLimitFlag,
utils.NATFlag,
utils.NoDiscoverFlag,
utils.DiscoveryV5Flag,
utils.NetrestrictFlag,
utils.NodeKeyFileFlag,
utils.NodeKeyHexFlag,
utils.DevModeFlag,
utils.TestnetFlag,
utils.RinkebyFlag,
utils.VMEnableDebugFlag,
utils.NetworkIdFlag,
utils.RPCCORSDomainFlag,
utils.EthStatsURLFlag,

```
utils.MetricsEnabledFlag,
utils.FakePoWFlag,
utils.NoCompactionFlag,
utils.GpoBlocksFlag,
utils.GpoPercentileFlag,
utils.ExtraDataFlag,
configFileFlag,
}

rpcFlags = []cli.Flag{
utils.RPCEnabledFlag,
utils.RPCListenAddrFlag,
utils.RPCPortFlag,
utils.RPCApiFlag,
utils.WSEnabledFlag,
utils.WSListenAddrFlag,
utils.WSPortFlag,
utils.WSApiFlag,
utils.WSAllowedOriginsFlag,
utils.IPCDisabledFlag,
utils.IPCPathFlag,
}

whisperFlags = []cli.Flag{
utils.WhisperEnabledFlag,
utils.WhisperMaxMessageSizeFlag,
utils.WhisperMinPOWFlag,
}
)

func init() {
// Initialize the CLI app and start Geth
app.Action = geth
app.HideVersion = true // we have a command to print the version
app.Copyright = "Copyright 2013-2017 The go-ethereum Authors"
app.Commands = []cli.Command{
// See chaincmd.go:
initCommand,
importCommand,
exportCommand,
removedbCommand,
dumpCommand,
```

```
// See monitorcmd.go:
monitorCommand,
// See accountcmd.go:
accountCommand,
walletCommand,
// See consolecmd.go:
consoleCommand,
attachCommand,
javascriptCommand,
// See misccmd.go:
makedagCommand,
versionCommand,
bugCommand,
licenseCommand,
// See config.go
dumpConfigCommand,
}

app.Flags = append(app.Flags, nodeFlags...)
app.Flags = append(app.Flags, rpcFlags...)
app.Flags = append(app.Flags, consoleFlags...)
app.Flags = append(app.Flags, debug.Flags...)
app.Flags = append(app.Flags, whisperFlags...)

app.Before = func(ctx *cli.Context) error {
runtime.GOMAXPROCS(runtime.NumCPU())
if err := debug.Setup(ctx); err != nil {
return err
}
// Start system runtime metrics collection
go metrics.CollectProcessMetrics(3 * time.Second)

utils.SetupNetwork(ctx)
return nil
}

app.After = func(ctx *cli.Context) error {
debug.Exit()
console.Stdin.Close() // Resets terminal mode.
return nil
}
}
```

```
func main() {
if err := app.Run(os.Args); err != nil {
fmt.Fprintln(os.Stderr, err)
os.Exit(1)
}
}

// geth is the main entry point into the system if no special subcommand is ran.
// It creates a default node based on the command line arguments and runs it in
// blocking mode, waiting for it to be shut down.
func geth(ctx *cli.Context) error {
node := makeFullNode(ctx)
startNode(ctx, node)
node.Wait()
return nil
}

// startNode boots up the system node and all registered protocols, after which
// it unlocks any requested accounts, and starts the RPC/IPC interfaces and the
// miner.
func startNode(ctx *cli.Context, stack *node.Node) {
// Start up the node itself
utils.StartNode(stack)

// Unlock any account specifically requested
ks := stack.AccountManager().Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore)

passwords := utils.MakePasswordList(ctx)
unlocks := strings.Split(ctx.GlobalString(utils.UnlockedAccountFlag.Name), ",")
for i, account := range unlocks {
if trimmed := strings.TrimSpace(account); trimmed != "" {
unlockAccount(ctx, ks, trimmed, i, passwords)
}
}
// Register wallet event handlers to open and auto-derive wallets
events := make(chan accounts.WalletEvent, 16)
stack.AccountManager().Subscribe(events)

go func() {
// Create an chain state reader for self-derivation
rpcClient, err := stack.Attach()
```

```
if err != nil {
utils.Fatalf("Failed to attach to self: %v", err)
}
stateReader := ethclient.NewClient(rpcClient)

// Open and self derive any wallets already attached
for _, wallet := range stack.AccountManager().Wallets() {
if err := wallet.Open(""); err != nil {
log.Warn("Failed to open wallet", "url", wallet.URL(), "err", err)
} else {
wallet.SelfDerive(accounts.DefaultBaseDerivationPath, stateReader)
}
}
// Listen for wallet event till termination
for event := range events {
if event.Arrive {
if err := event.Wallet.Open(""); err != nil {
log.Warn("New wallet appeared, failed to open", "url", event.Wallet.URL(), "err", err)
} else {
log.Info("New wallet appeared", "url", event.Wallet.URL(), "status", event.Wallet.Status())
event.Wallet.SelfDerive(accounts.DefaultBaseDerivationPath, stateReader)
}
} else {
log.Info("Old wallet dropped", "url", event.Wallet.URL())
event.Wallet.Close()
}
}
}()
// Start auxiliary services if enabled
if ctx.GlobalBool(utils.MiningEnabledFlag.Name) {
// Mining only makes sense if a full Ethereum node is running
var ethereum *eth.Ethereum
if err := stack.Service(&ethereum); err != nil {
utils.Fatalf("ethereum service not running: %v", err)
}
// Use a reduced number of threads if requested
if threads := ctx.GlobalInt(utils.MinerThreadsFlag.Name); threads > 0 {
type threaded interface {
SetThreads(threads int)
}
if th, ok := ethereum.Engine().(threaded); ok {
th.SetThreads(threads)
```

```
}
}
// Set the gas price to the limits from the CLI and start mining
ethereum.TxPool().SetGasPrice(utils.GlobalBig(ctx, utils.GasPriceFlag.Name))
if err := ethereum.StartMining(true); err != nil {
utils.Fatalf("Failed to start mining: %v", err)
}
}
}
```

68:F:\git\coin\ethereum\go-ethereum\cmd\geth\misccmd.go

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

package main

import (
"fmt"
"io/ioutil"
"os"
"path/filepath"
"runtime"
"strconv"
"strings"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/eth"
"github.com/ethereum/go-ethereum/params"
"gopkg.in/urfave/cli.v1"
)

var (
makedagCommand = cli.Command{
Action:    utils.MigrateFlags(makedag),
Name:      "makedag",
Usage:     "Generate ethash DAG (for testing)",
ArgsUsage: "<blockNum> <outputDir>",
Category:  "MISCELLANEOUS COMMANDS",
Description: `
The makedag command generates an ethash DAG in /tmp/dag.

This command exists to support the system testing project.
```

```
Regular users do not need to execute it.
`,
}
versionCommand = cli.Command{
Action:    utils.MigrateFlags(version),
Name:      "version",
Usage:     "Print version numbers",
ArgsUsage: " ",
Category:  "MISCELLANEOUS COMMANDS",
Description: `
The output of this command is supposed to be machine-readable.
`,
}
licenseCommand = cli.Command{
Action:    utils.MigrateFlags(license),
Name:      "license",
Usage:     "Display license information",
ArgsUsage: " ",
Category:  "MISCELLANEOUS COMMANDS",
}
)

func makedag(ctx *cli.Context) error {
args := ctx.Args()
wrongArgs := func() {
utils.Fatalf(`Usage: geth makedag <block number> <outputdir>`)
}
switch {
case len(args) == 2:
blockNum, err := strconv.ParseUint(args[0], 0, 64)
dir := args[1]
if err != nil {
wrongArgs()
} else {
dir = filepath.Clean(dir)
// seems to require a trailing slash
if !strings.HasSuffix(dir, "/") {
dir = dir + "/"
}
_, err = ioutil.ReadDir(dir)
if err != nil {
utils.Fatalf("Can't find dir")
```

```
}
fmt.Println("making DAG, this could take awhile...")
ethash.MakeDataset(blockNum, dir)
}
default:
wrongArgs()
}
return nil
}

func version(ctx *cli.Context) error {
fmt.Println(strings.Title(clientIdentifier))
fmt.Println("Version:", params.Version)
if gitCommit != "" {
fmt.Println("Git Commit:", gitCommit)
}
fmt.Println("Architecture:", runtime.GOARCH)
fmt.Println("Protocol Versions:", eth.ProtocolVersions)
fmt.Println("Network Id:", eth.DefaultConfig.NetworkId)
fmt.Println("Go Version:", runtime.Version())
fmt.Println("Operating System:", runtime.GOOS)
fmt.Printf("GOPATH=%s\n", os.Getenv("GOPATH"))
fmt.Printf("GOROOT=%s\n", runtime.GOROOT())
return nil
}

func license(_ *cli.Context) error {
fmt.Println(`Geth is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

Geth is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with geth. If not, see <http://www.gnu.org/licenses/>.
`)
return nil
}
```

69:F:\git\coin\ethereum\go-ethereum\cmd\geth\monitorcmd.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"fmt"
"math"
"reflect"
"runtime"
"sort"
"strings"
"time"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/rpc"
"github.com/gizak/termui"
"gopkg.in/urfave/cli.v1"
)

var (
monitorCommandAttachFlag = cli.StringFlag{
Name: "attach",
Value: node.DefaultIPCEndpoint(clientIdentifier),
Usage: "API endpoint to attach to",
}
monitorCommandRowsFlag = cli.IntFlag{
Name: "rows",
Value: 5,
Usage: "Maximum rows in the chart grid",
}
monitorCommandRefreshFlag = cli.IntFlag{
Name: "refresh",
Value: 3,
Usage: "Refresh interval in seconds",
}
monitorCommand = cli.Command{
Action: utils.MigrateFlags(monitor), // keep track of migration progress
Name: "monitor",
Usage: "Monitor and visualize node metrics",

```
ArgsUsage: " ",
Category:  "MONITOR COMMANDS",
Description: `
The Geth monitor is a tool to collect and visualize various internal metrics
gathered by the node, supporting different chart types as well as the capacity
to display multiple metrics simultaneously.
`,
Flags: []cli.Flag{
monitorCommandAttachFlag,
monitorCommandRowsFlag,
monitorCommandRefreshFlag,
},
}
)

// monitor starts a terminal UI based monitoring tool for the requested metrics.
func monitor(ctx *cli.Context) error {
var (
client *rpc.Client
err    error
)
// Attach to an Ethereum node over IPC or RPC
endpoint := ctx.String(monitorCommandAttachFlag.Name)
if client, err = dialRPC(endpoint); err != nil {
utils.Fatalf("Unable to attach to geth node: %v", err)
}
defer client.Close()

// Retrieve all the available metrics and resolve the user pattens
metrics, err := retrieveMetrics(client)
if err != nil {
utils.Fatalf("Failed to retrieve system metrics: %v", err)
}
monitored := resolveMetrics(metrics, ctx.Args())
if len(monitored) == 0 {
list := expandMetrics(metrics, "")
sort.Strings(list)

if len(list) > 0 {
utils.Fatalf("No metrics specified.\n\nAvailable:\n - %s", strings.Join(list, "\n - "))
} else {
utils.Fatalf("No metrics collected by geth (--%s).\n", utils.MetricsEnabledFlag.Name)
```

```
}
}
sort.Strings(monitored)
if cols := len(monitored) / ctx.Int(monitorCommandRowsFlag.Name); cols > 6 {
utils.Fatalf("Requested metrics (%d) spans more that 6 columns:\n - %s", len(monitored),
strings.Join(monitored, "\n - "))
}
// Create and configure the chart UI defaults
if err := termui.Init(); err != nil {
utils.Fatalf("Unable to initialize terminal UI: %v", err)
}
defer termui.Close()

rows := len(monitored)
if max := ctx.Int(monitorCommandRowsFlag.Name); rows > max {
rows = max
}
cols := (len(monitored) + rows - 1) / rows
for i := 0; i < rows; i++ {
termui.Body.AddRows(termui.NewRow())
}
// Create each individual data chart
footer := termui.NewPar("")
footer.Block.Border = true
footer.Height = 3

charts := make([]*termui.LineChart, len(monitored))
units := make([]int, len(monitored))
data := make([][]float64, len(monitored))
for i := 0; i < len(monitored); i++ {
charts[i] = createChart((termui.TermHeight() - footer.Height) / rows)
row := termui.Body.Rows[i%rows]
row.Cols = append(row.Cols, termui.NewCol(12/cols, 0, charts[i]))
}
termui.Body.AddRows(termui.NewRow(termui.NewCol(12, 0, footer)))

refreshCharts(client, monitored, data, units, charts, ctx, footer)
termui.Body.Align()
termui.Render(termui.Body)

// Watch for various system events, and periodically refresh the charts
termui.Handle("/sys/kbd/C-c", func(termui.Event) {
```

```
termui.StopLoop()
})
termui.Handle("/sys/wnd/resize", func(termui.Event) {
termui.Body.Width = termui.TermWidth()
for _, chart := range charts {
chart.Height = (termui.TermHeight() - footer.Height) / rows
}
termui.Body.Align()
termui.Render(termui.Body)
})
go func() {
tick := time.NewTicker(time.Duration(ctx.Int(monitorCommandRefreshFlag.Name)) * time.Second)
for range tick.C {
if refreshCharts(client, monitored, data, units, charts, ctx, footer) {
termui.Body.Align()
}
termui.Render(termui.Body)
}
}()
termui.Loop()
return nil
}

// retrieveMetrics contacts the attached geth node and retrieves the entire set
// of collected system metrics.
func retrieveMetrics(client *rpc.Client) (map[string]interface{}, error) {
var metrics map[string]interface{}
err := client.Call(&metrics, "debug_metrics", true)
return metrics, err
}

// resolveMetrics takes a list of input metric patterns, and resolves each to one
// or more canonical metric names.
func resolveMetrics(metrics map[string]interface{}, patterns []string) []string {
res := []string{}
for _, pattern := range patterns {
res = append(res, resolveMetric(metrics, pattern, "")...)
}
return res
}

// resolveMetrics takes a single of input metric pattern, and resolves it to one
```

```
// or more canonical metric names.
func resolveMetric(metrics map[string]interface{}, pattern string, path string) []string {
results := []string{}

// If a nested metric was requested, recurse optionally branching (via comma)
parts := strings.SplitN(pattern, "/", 2)
if len(parts) > 1 {
for _, variation := range strings.Split(parts[0], ",") {
if submetrics, ok := metrics[variation].(map[string]interface{}); !ok {
utils.Fatalf("Failed to retrieve system metrics: %s", path+variation)
return nil
} else {
results = append(results, resolveMetric(submetrics, parts[1], path+variation+"/")...)
}
}
return results
}
// Depending what the last link is, return or expand
for _, variation := range strings.Split(pattern, ",") {
switch metric := metrics[variation].(type) {
case float64:
// Final metric value found, return as singleton
results = append(results, path+variation)

case map[string]interface{}:
results = append(results, expandMetrics(metric, path+variation+"/")...)

default:
utils.Fatalf("Metric pattern resolved to unexpected type: %v", reflect.TypeOf(metric))
return nil
}
}
return results
}

// expandMetrics expands the entire tree of metrics into a flat list of paths.
func expandMetrics(metrics map[string]interface{}, path string) []string {
// Iterate over all fields and expand individually
list := []string{}
for name, metric := range metrics {
switch metric := metric.(type) {
case float64:
```

```
// Final metric value found, append to list
list = append(list, path+name)

case map[string]interface{}:
// Tree of metrics found, expand recursively
list = append(list, expandMetrics(metric, path+name+"/")...)

default:
utils.Fatalf("Metric pattern %s resolved to unexpected type: %v", path+name,
reflect.TypeOf(metric))
return nil
}
}
return list
}

// fetchMetric iterates over the metrics map and retrieves a specific one.
func fetchMetric(metrics map[string]interface{}, metric string) float64 {
parts := strings.Split(metric, "/")
for _, part := range parts[:len(parts)-1] {
var found bool
metrics, found = metrics[part].(map[string]interface{})
if !found {
return 0
}
}
if v, ok := metrics[parts[len(parts)-1]].(float64); ok {
return v
}
return 0
}

// refreshCharts retrieves a next batch of metrics, and inserts all the new
// values into the active datasets and charts
func refreshCharts(client *rpc.Client, metrics []string, data [][]float64, units []int, charts
[]*termui.LineChart, ctx *cli.Context, footer *termui.Par) (realign bool) {
values, err := retrieveMetrics(client)
for i, metric := range metrics {
if len(data) < 512 {
data[i] = append([]float64{fetchMetric(values, metric)}, data[i]...)
} else {
data[i] = append([]float64{fetchMetric(values, metric)}, data[i][:len(data[i])-1]...)
```

```
}
if updateChart(metric, data[i], &units[i], charts[i], err) {
realign = true
}
}
updateFooter(ctx, err, footer)
return
}

// updateChart inserts a dataset into a line chart, scaling appropriately as to
// not display weird labels, also updating the chart label accordingly.
func updateChart(metric string, data []float64, base *int, chart *termui.LineChart, err error) (realign
bool) {
dataUnits := []string{"", "K", "M", "G", "T", "E"}
timeUnits := []string{"ns", "µs", "ms", "s", "ks", "ms"}
colors := []termui.Attribute{termui.ColorBlue, termui.ColorCyan, termui.ColorGreen,
termui.ColorYellow, termui.ColorRed, termui.ColorRed}

// Extract only part of the data that's actually visible
if chart.Width*2 < len(data) {
data = data[:chart.Width*2]
}
// Find the maximum value and scale under 1K
high := 0.0
if len(data) > 0 {
high = data[0]
for _, value := range data[1:] {
high = math.Max(high, value)
}
}
unit, scale := 0, 1.0
for high >= 1000 && unit+1 < len(dataUnits) {
high, unit, scale = high/1000, unit+1, scale*1000
}
// If the unit changes, re-create the chart (hack to set max height...)
if unit != *base {
realign, *base, *chart = true, unit, *createChart(chart.Height)
}
// Update the chart's data points with the scaled values
if cap(chart.Data) < len(data) {
chart.Data = make([]float64, len(data))
}
```

```
chart.Data = chart.Data[:len(data)]
for i, value := range data {
chart.Data[i] = value / scale
}
// Update the chart's label with the scale units
units := dataUnits
if strings.Contains(metric, "/Percentiles/") || strings.Contains(metric, "/pauses/") ||
strings.Contains(metric, "/time/") {
units = timeUnits
}
chart.BorderLabel = metric
if len(units[unit]) > 0 {
chart.BorderLabel += " [" + units[unit] + "]"
}
chart.LineColor = colors[unit] | termui.AttrBold
if err != nil {
chart.LineColor = termui.ColorRed | termui.AttrBold
}
return
}

// createChart creates an empty line chart with the default configs.
func createChart(height int) *termui.LineChart {
chart := termui.NewLineChart()
if runtime.GOOS == "windows" {
chart.Mode = "dot"
}
chart.DataLabels = []string{""}
chart.Height = height
chart.AxesColor = termui.ColorWhite
chart.PaddingBottom = -2

chart.BorderLabelFg = chart.BorderFg | termui.AttrBold
chart.BorderFg = chart.BorderBg

return chart
}

// updateFooter updates the footer contents based on any encountered errors.
func updateFooter(ctx *cli.Context, err error, footer *termui.Par) {
// Generate the basic footer
refresh := time.Duration(ctx.Int(monitorCommandRefreshFlag.Name)) * time.Second
```

```
footer.Text = fmt.Sprintf("Press Ctrl+C to quit. Refresh interval: %v.", refresh)
footer.TextFgColor = termui.ThemeAttr("par.fg") | termui.AttrBold

// Append any encountered errors
if err != nil {
footer.Text = fmt.Sprintf("Error: %v.", err)
footer.TextFgColor = termui.ColorRed | termui.AttrBold
}
}

70:F:\git\coin\ethereum\go-ethereum\cmd\geth\run_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"fmt"
"io/ioutil"
"os"
"testing"

"github.com/docker/docker/pkg/reexec"
"github.com/ethereum/go-ethereum/internal/cmdtest"
)

func tmpdir(t *testing.T) string {
dir, err := ioutil.TempDir("", "geth-test")
if err != nil {
t.Fatal(err)
}
return dir
}

type testgeth struct {
*cmdtest.TestCmd

// template variables for expect
Datadir   string
Etherbase string
}

func init() {
```

```
// Run the app if we've been exec'd as "geth-test" in runGeth.
reexec.Register("geth-test", func() {
if err := app.Run(os.Args); err != nil {
fmt.Fprintln(os.Stderr, err)
os.Exit(1)
}
os.Exit(0)
})
}

func TestMain(m *testing.M) {
// check if we have been reexec'd
if reexec.Init() {
return
}
os.Exit(m.Run())
}

// spawns geth with the given command line args. If the args don't set --datadir, the
// child g gets a temporary data directory.
func runGeth(t *testing.T, args ...string) *testgeth {
tt := &testgeth{}
tt.TestCmd = cmdtest.NewTestCmd(t, tt)
for i, arg := range args {
switch {
case arg == "-datadir" || arg == "--datadir":
if i < len(args)-1 {
tt.Datadir = args[i+1]
}
case arg == "-etherbase" || arg == "--etherbase":
if i < len(args)-1 {
tt.Etherbase = args[i+1]
}
}
}
if tt.Datadir == "" {
tt.Datadir = tmpdir(t)
tt.Cleanup = func() { os.RemoveAll(tt.Datadir) }
args = append([]string{"-datadir", tt.Datadir}, args...)
// Remove the temporary datadir if something fails below.
defer func() {
if t.Failed() {
```

```
tt.Cleanup()
}
}()
}

// Boot "geth". This actually runs the test binary but the TestMain
// function will prevent any tests from running.
tt.Run("geth-test", args...)

return tt
}

71:F:\git\coin\ethereum\go-ethereum\cmd\geth\usage.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// Contains the geth command usage template and generator.

package main

import (
"io"
"sort"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/internal/debug"
"gopkg.in/urfave/cli.v1"
)

// AppHelpTemplate is the test template for the default, global app help topic.
var AppHelpTemplate = `NAME:
   {{.App.Name}} - {{.App.Usage}}

   Copyright 2013-2017 The go-ethereum Authors

USAGE:
   {{.App.HelpName}} [options]{{if .App.Commands}} command [command options]{{end}} {{if
.App.ArgsUsage}}{{.App.ArgsUsage}}{{else}}[arguments...]{{end}}
   {{if .App.Version}}
VERSION:
   {{.App.Version}}
   {{end}}{{if len .App.Authors}}
AUTHOR(S):
```

```
   {{range .App.Authors}}{{ . }}{{end}}
   {{end}}{{if .App.Commands}}
COMMANDS:
   {{range .App.Commands}}{{join .Names ", "}}{{ "\t" }}{{.Usage}}
   {{end}}{{end}}{{if .FlagGroups}}
{{range .FlagGroups}}{{.Name}} OPTIONS:
  {{range .Flags}}{{.}}
  {{end}}
{{end}}{{end}}{{if .App.Copyright }}
COPYRIGHT:
   {{.App.Copyright}}
   {{end}}
`

// flagGroup is a collection of flags belonging to a single topic.
type flagGroup struct {
Name  string
Flags []cli.Flag
}

// AppHelpFlagGroups is the application flags, grouped by functionality.
var AppHelpFlagGroups = []flagGroup{
{
Name: "ETHEREUM",
Flags: []cli.Flag{
configFileFlag,
utils.DataDirFlag,
utils.KeyStoreDirFlag,
utils.NoUSBFlag,
utils.NetworkIdFlag,
utils.TestnetFlag,
utils.RinkebyFlag,
utils.DevModeFlag,
utils.SyncModeFlag,
utils.EthStatsURLFlag,
utils.IdentityFlag,
utils.LightServFlag,
utils.LightPeersFlag,
utils.LightKDFFlag,
},
},
{
```

```
Name: "ETHASH",
Flags: []cli.Flag{
utils.EthashCacheDirFlag,
utils.EthashCachesInMemoryFlag,
utils.EthashCachesOnDiskFlag,
utils.EthashDatasetDirFlag,
utils.EthashDatasetsInMemoryFlag,
utils.EthashDatasetsOnDiskFlag,
},
},
{
Name: "TRANSACTION POOL",
Flags: []cli.Flag{
utils.TxPoolPriceLimitFlag,
utils.TxPoolPriceBumpFlag,
utils.TxPoolAccountSlotsFlag,
utils.TxPoolGlobalSlotsFlag,
utils.TxPoolAccountQueueFlag,
utils.TxPoolGlobalQueueFlag,
utils.TxPoolLifetimeFlag,
},
},
{
Name: "PERFORMANCE TUNING",
Flags: []cli.Flag{
utils.CacheFlag,
utils.TrieCacheGenFlag,
},
},
{
Name: "ACCOUNT",
Flags: []cli.Flag{
utils.UnlockedAccountFlag,
utils.PasswordFileFlag,
},
},
{
Name: "API AND CONSOLE",
Flags: []cli.Flag{
utils.RPCEnabledFlag,
utils.RPCListenAddrFlag,
utils.RPCPortFlag,
```

```
utils.RPCApiFlag,
utils.WSEnabledFlag,
utils.WSListenAddrFlag,
utils.WSPortFlag,
utils.WSApiFlag,
utils.WSAllowedOriginsFlag,
utils.IPCDisabledFlag,
utils.IPCPathFlag,
utils.RPCCORSDomainFlag,
utils.JSpathFlag,
utils.ExecFlag,
utils.PreloadJSFlag,
},
},
{
Name: "NETWORKING",
Flags: []cli.Flag{
utils.BootnodesFlag,
utils.BootnodesV4Flag,
utils.BootnodesV5Flag,
utils.ListenPortFlag,
utils.MaxPeersFlag,
utils.MaxPendingPeersFlag,
utils.NATFlag,
utils.NoDiscoverFlag,
utils.DiscoveryV5Flag,
utils.NetrestrictFlag,
utils.NodeKeyFileFlag,
utils.NodeKeyHexFlag,
},
},
{
Name: "MINER",
Flags: []cli.Flag{
utils.MiningEnabledFlag,
utils.MinerThreadsFlag,
utils.EtherbaseFlag,
utils.TargetGasLimitFlag,
utils.GasPriceFlag,
utils.ExtraDataFlag,
},
},
```

```
{
Name: "GAS PRICE ORACLE",
Flags: []cli.Flag{
utils.GpoBlocksFlag,
utils.GpoPercentileFlag,
},
},
{
Name: "VIRTUAL MACHINE",
Flags: []cli.Flag{
utils.VMEnableDebugFlag,
},
},
{
Name: "LOGGING AND DEBUGGING",
Flags: append([]cli.Flag{
utils.MetricsEnabledFlag,
utils.FakePoWFlag,
utils.NoCompactionFlag,
}, debug.Flags...),
},
{
Name:  "WHISPER (EXPERIMENTAL)",
Flags: whisperFlags,
},
{
Name: "DEPRECATED",
Flags: []cli.Flag{
utils.FastSyncFlag,
utils.LightModeFlag,
},
},
{
Name: "MISC",
},
}

// byCategory sorts an array of flagGroup by Name in the order
// defined in AppHelpFlagGroups.
type byCategory []flagGroup

func (a byCategory) Len() int      { return len(a) }
```

```
func (a byCategory) Swap(i, j int) { a[i], a[j] = a[j], a[i] }
func (a byCategory) Less(i, j int) bool {
iCat, jCat := a[i].Name, a[j].Name
iIdx, jIdx := len(AppHelpFlagGroups), len(AppHelpFlagGroups) // ensure non categorized flags
come last

for i, group := range AppHelpFlagGroups {
if iCat == group.Name {
iIdx = i
}
if jCat == group.Name {
jIdx = i
}
}

return iIdx < jIdx
}

func flagCategory(flag cli.Flag) string {
for _, category := range AppHelpFlagGroups {
for _, flg := range category.Flags {
if flg.GetName() == flag.GetName() {
return category.Name
}
}
}
return "MISC"
}

func init() {
// Override the default app help template
cli.AppHelpTemplate = AppHelpTemplate

// Define a one shot struct to pass to the usage template
type helpData struct {
App        interface{}
FlagGroups []flagGroup
}

// Override the default app help printer, but only for the global app help
originalHelpPrinter := cli.HelpPrinter
cli.HelpPrinter = func(w io.Writer, tmpl string, data interface{}) {
```

```
if tmpl == AppHelpTemplate {
// Iterate over all the flags and add any uncategorized ones
categorized := make(map[string]struct{})
for _, group := range AppHelpFlagGroups {
for _, flag := range group.Flags {
categorized[flag.String()] = struct{}{}
}
}
uncategorized := []cli.Flag{}
for _, flag := range data.(*cli.App).Flags {
if _, ok := categorized[flag.String()]; !ok {
uncategorized = append(uncategorized, flag)
}
}
if len(uncategorized) > 0 {
// Append all ungategorized options to the misc group
miscs := len(AppHelpFlagGroups[len(AppHelpFlagGroups)-1].Flags)
AppHelpFlagGroups[len(AppHelpFlagGroups)-1].Flags =
append(AppHelpFlagGroups[len(AppHelpFlagGroups)-1].Flags, uncategorized...)

// Make sure they are removed afterwards
defer func() {
AppHelpFlagGroups[len(AppHelpFlagGroups)-1].Flags =
AppHelpFlagGroups[len(AppHelpFlagGroups)-1].Flags[:miscs]
}()
}
// Render out custom usage screen
originalHelpPrinter(w, tmpl, helpData{data, AppHelpFlagGroups})
} else if tmpl == utils.CommandHelpTemplate {
// Iterate over all command specific flags and categorize them
categorized := make(map[string][]cli.Flag)
for _, flag := range data.(cli.Command).Flags {
if _, ok := categorized[flag.String()]; !ok {
categorized[flagCategory(flag)] = append(categorized[flagCategory(flag)], flag)
}
}

// sort to get a stable ordering
sorted := make([]flagGroup, 0, len(categorized))
for cat, flgs := range categorized {
sorted = append(sorted, flagGroup{cat, flgs})
}
```

```
sort.Sort(byCategory(sorted))

// add sorted array to data and render with default printer
originalHelpPrinter(w, tmpl, map[string]interface{}{
"cmd":              data,
"categorizedFlags": sorted,
})
} else {
originalHelpPrinter(w, tmpl, data)
}
}
}

72:F:\git\coin\ethereum\go-ethereum\cmd\internal\browser\browser.go
func Commands() [][]string {
var cmds [][]string
if exe := os.Getenv("BROWSER"); exe != "" {
cmds = append(cmds, []string{exe})
}
switch runtime.GOOS {
case "darwin":
cmds = append(cmds, []string{"/usr/bin/open"})
case "windows":
cmds = append(cmds, []string{"cmd", "/c", "start"})
default:
cmds = append(cmds, []string{"xdg-open"})
}
cmds = append(cmds,
[]string{"chrome"},
[]string{"google-chrome"},
[]string{"chromium"},
[]string{"firefox"},
)
return cmds
}

// Open tries to open url in a browser and reports whether it succeeded.
func Open(url string) bool {
for _, args := range Commands() {
cmd := exec.Command(args[0], append(args[1:], url)...)
if cmd.Start() == nil {
return true
```

```
}
}
return false
}

73:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\module.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"encoding/json"
"errors"
"fmt"
"net"
"strconv"
"strings"
"time"

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

var (
// ErrServiceUnknown is returned when a service container doesn't exist.
ErrServiceUnknown = errors.New("service unknown")

// ErrServiceOffline is returned when a service container exists, but it is not
// running.
ErrServiceOffline = errors.New("service offline")

// ErrServiceUnreachable is returned when a service container is running, but
// seems to not respond to communication attempts.
ErrServiceUnreachable = errors.New("service unreachable")

// ErrNotExposed is returned if a web-service doesn't have an exposed port, nor
// a reverse-proxy in front of it to forward requests.
ErrNotExposed = errors.New("service not exposed, nor proxied")
)

// containerInfos is a heavily reduced version of the huge inspection dataset
// returned from docker inspect, parsed into a form easily usable by puppeth.
type containerInfos struct {
```

```
running bool              // Flag whether the container is running currently
envvars map[string]string // Collection of environmental variables set on the container
portmap map[string]int    // Port mapping from internal port/proto combos to host binds
volumes map[string]string // Volume mount points from container to host directories
}

// inspectContainer runs docker inspect against a running container
func inspectContainer(client *sshClient, container string) (*containerInfos, error) {
// Check whether there's a container running for the service
out, err := client.Run(fmt.Sprintf("docker inspect %s", container))
if err != nil {
return nil, ErrServiceUnknown
}
// If yes, extract various configuration options
type inspection struct {
State struct {
Running bool
}
Mounts []struct {
Source      string
Destination string
}
Config struct {
Env []string
}
HostConfig struct {
PortBindings map[string][]map[string]string
}
}
var inspects []inspection
if err = json.Unmarshal(out, &inspects); err != nil {
return nil, err
}
inspect := inspects[0]

// Infos retrieved, parse the above into something meaningful
infos := &containerInfos{
running: inspect.State.Running,
envvars: make(map[string]string),
portmap: make(map[string]int),
volumes: make(map[string]string),
}
```

```
for _, envvar := range inspect.Config.Env {
if parts := strings.Split(envvar, "="); len(parts) == 2 {
infos.envvars[parts[0]] = parts[1]
}
}
for portname, details := range inspect.HostConfig.PortBindings {
if len(details) > 0 {
port, _ := strconv.Atoi(details[0]["HostPort"])
infos.portmap[portname] = port
}
}
for _, mount := range inspect.Mounts {
infos.volumes[mount.Destination] = mount.Source
}
return infos, err
}

// tearDown connects to a remote machine via SSH and terminates docker containers
// running with the specified name in the specified network.
func tearDown(client *sshClient, network string, service string, purge bool) ([]byte, error) {
// Tear down the running (or paused) container
out, err := client.Run(fmt.Sprintf("docker rm -f %s_%s_1", network, service))
if err != nil {
return out, err
}
// If requested, purge the associated docker image too
if purge {
return client.Run(fmt.Sprintf("docker rmi %s/%s", network, service))
}
return nil, nil
}

// resolve retrieves the hostname a service is running on either by returning the
// actual server name and port, or preferably an nginx virtual host if available.
func resolve(client *sshClient, network string, service string, port int) (string, error) {
// Inspect the service to get various configurations from it
infos, err := inspectContainer(client, fmt.Sprintf("%s_%s_1", network, service))
if err != nil {
return "", err
}
if !infos.running {
return "", ErrServiceOffline
```

```
}
// Container online, extract any environmental variables
if vhost := infos.envvars["VIRTUAL_HOST"]; vhost != "" {
return vhost, nil
}
return fmt.Sprintf("%s:%d", client.server, port), nil
}

// checkPort tries to connect to a remote host on a given
func checkPort(host string, port int) error {
log.Trace("Verifying remote TCP connectivity", "server", host, "port", port)
conn, err := net.DialTimeout("tcp", fmt.Sprintf("%s:%d", host, port), time.Second)
if err != nil {
return err
}
conn.Close()
return nil
}

74:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\module_dashboard.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bytes"
"fmt"
"html/template"
"math/rand"
"path/filepath"
"strings"

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

// dashboardContent is the actual dashboard HTML content to serve up when users
// load the dashboard website.
var dashboardContent = `
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<!-- Meta, title, CSS, favicons, etc. -->
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">

<title>{{.NetworkTitle}}: Ethereum Testnet</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
rel="stylesheet">
<link href="https://cdnjs.cloudflare.com/ajax/libs/gentelella/1.3.0/css/custom.min.css"
rel="stylesheet">
<style>
.vertical-center {
min-height: 100%;
min-height: 95vh;
display: flex;
align-items: center;
}
.nav.side-menu li a {
font-size: 18px;
}
.nav-sm .nav.side-menu li a {
font-size: 10px;
}
pre{
white-space: pre-wrap;
}
</style>
</head>

<body class="nav-sm" style="overflow-x: hidden">
<div class="container body">
<div class="main_container">
<div class="col-md-3 left_col">
<div class="left_col scroll-view">
<div class="navbar nav_title" style="border: 0; margin-top: 8px;">
<a class="site_title"><i class="fa fa-globe" style="margin-left: 6px"></i> <span>{{.NetworkTitle}}
Testnet</span></a>
</div>
<div class="clearfix"></div>
```

```
<br />
<div id="sidebar-menu" class="main_menu_side hidden-print main_menu">
<div class="menu_section">
<ul class="nav side-menu">
{{if .EthstatsPage}}<li><a onclick="load('//{{.EthstatsPage}}')"><i class="fa fa-tachometer"></i>
Network Stats</a></li>{{end}}
{{if .ExplorerPage}}<li><a onclick="load('//{{.ExplorerPage}}')"><i class="fa fa-database"></i>
Block Explorer</a></li>{{end}}
{{if .WalletPage}}<li><a onclick="load('//{{.WalletPage}}')"><i class="fa fa-address-book-o"></i>
Browser Wallet</a></li>{{end}}
{{if .FaucetPage}}<li><a onclick="load('//{{.FaucetPage}}')"><i class="fa fa-bath"></i> Crypto
Faucet</a></li>{{end}}
<li id="connect"><a><i class="fa fa-plug"></i> Connect Yourself</a>
<ul id="connect_list" class="nav child_menu">
<li><a onclick="$('#connect').removeClass('active'); $('#connect_list').toggle(); load('#connect-go-
ethereum-geth')">Go Ethereum: Geth</a></li>
<li><a onclick="$('#connect').removeClass('active'); $('#connect_list').toggle(); load('#connect-go-
ethereum-mist')">Go Ethereum: Wallet & Mist</a></li>
<li><a onclick="$('#connect').removeClass('active'); $('#connect_list').toggle(); load('#connect-go-
ethereum-mobile')">Go Ethereum: Android & iOS</a></li>
</ul>
</li>
<li><a onclick="load('#about')"><i class="fa fa-heartbeat"></i> About Puppeth</a></li>
</ul>
</div>
</div>
</div>
</div>
<div class="right_col" role="main" style="padding: 0">
<div id="connect-go-ethereum-geth" hidden style="padding: 16px;">
<div class="page-title">
<div class="title_left">
<h3>Connect Yourself &amp;ndash; Go Ethereum: Geth</h3>
</div>
</div>
<div class="clearfix"></div>
<div class="row">
<div class="col-md-6">
<div class="x_panel">
<div class="x_title">
<h2><i class="fa fa-archive" aria-hidden="true"></i> Archive node <small>Retains all historical
data</small></h2>
```

An archive node synchronizes the blockchain by downloading the full chain from the genesis block to the current head block, executing all the transactions contained within. As the node crunches through the transactions, all past historical state is stored on disk, and can be queried for each and every block.
Initial processing required to execute all transactions may require non-negligible time and disk capacity required to store all past state may be non-insignificant. High end machines with SSD storage, modern CPUs and 8GB+ RAM are recommended.

To run an archive node, download <code>{{.GethGenesis}}</code> and start Geth with:
<pre>geth --datadir=$HOME/.{{.Network}} init {{.GethGenesis}}</pre>
<pre>geth --networkid={{.NetworkID}} --datadir=$HOME/.{{.Network}} --cache=1024 --syncmode=full{{if .Ethstats}} --ethstats='{{.Ethstats}}'{{end}} --bootnodes={{.BootnodesFullFlat}}</pre>

You can download Geth from https://geth.ethereum.org/downloads/.

<h2> Full node <small>Retains recent data only</small></h2>

A full node synchronizes the blockchain by downloading the full chain from the genesis block to the current head block, but does not execute the transactions. Instead, it downloads all the transactions receipts along with the entire recent state. As the node downloads the recent state directly, historical data can only be queried from that block onward.
Initial processing required to synchronize is more bandwidth intensive, but is light on the CPU and has significantly reduced disk requirements. Mid range machines with HDD storage, decent CPUs and 4GB+ RAM should be enough.

To run a full node, download <code>{{.GethGenesis}}</code> and start Geth with:

<pre>geth --datadir=$HOME/.{{.Network}} init {{.GethGenesis}}</pre>
<pre>geth --networkid={{.NetworkID}} --datadir=$HOME/.{{.Network}} --cache=512{{if .Ethstats}} --ethstats='{{.Ethstats}}'{{end}} --bootnodes={{.BootnodesFullFlat}}</pre>

You can download Geth from https://geth.ethereum.org/downloads/.

<h2> Light node <small>Retrieves data on demand</small></h2>

A light node synchronizes the blockchain by downloading and verifying only the chain of headers from the genesis block to the current head, without executing any transactions or retrieving any associated state. As no state is available locally, any interaction with the blockchain relies on on-demand data retrievals from remote nodes.
Initial processing required to synchronize is light, as it only verifies the validity of the headers; similarly required disk capacity is small, tallying around 500 bytes per header. Low end machines with arbitrary storage, weak CPUs and 512MB+ RAM should cope well.

To run a light node, download <code>{{.GethGenesis}}</code> and start Geth with:
<pre>geth --datadir=$HOME/.{{.Network}} --light init {{.GethGenesis}}</pre>
<pre>geth --networkid={{.NetworkID}} --datadir=$HOME/.{{.Network}} --syncmode=light{{if .Ethstats}} --ethstats='{{.Ethstats}}'{{end}} --bootnodes={{.BootnodesLightFlat}}</pre>

You can download Geth from https://geth.ethereum.org/downloads/.

<h2> Embedded node <small>Conserves memory vs. speed</small></h2>

An embedded node is a variation of the light node with configuration parameters tuned towards low memory footprint. As such, it may sacrifice processing and disk IO performance to conserve memory. It should be considered an experimental direction for now without hard guarantees or bounds on the resources used.
Initial processing required to synchronize is light, as it only verifies the validity of the headers; similarly required disk capacity is small, tallying around 500 bytes per header. Embedded machines with arbitrary storage, low power CPUs and 128MB+ RAM may work.

To run an embedded node, download <code>{{.GethGenesis}}</code> and start Geth with:
<pre>geth --datadir=$HOME/.{{.Network}} --light init {{.GethGenesis}}</pre>
<pre>geth --networkid={{.NetworkID}} --datadir=$HOME/.{{.Network}} --cache=32 --syncmode=light{{if .Ethstats}} --ethstats='{{.Ethstats}}'{{end}} --bootnodes={{.BootnodesLightFlat}}</pre>

You can download Geth from https://geth.ethereum.org/downloads/.

<div id="connect-go-ethereum-mist" hidden style="padding: 16px;">

<h3>Connect Yourself – Go Ethereum: Wallet & Mist</h3>

<h2> Desktop wallet <small>Interacts with

accounts and contracts</small></h2>

The Ethereum Wallet is an Electron based desktop application to manage your Ethereum accounts and funds. Beside the usual account life-cycle operations you would expect to perform, the wallet also provides a means to send transactions from your accounts and to interact with smart contracts deployed on the network.
Under the hood the wallet is backed by a go-ethereum full node, meaning that a mid range machine is assumed. Similarly, synchronization is based on fast-sync, which will download all blockchain data from the network and make it available to the wallet. Light nodes cannot currently fully back the wallet, but it's a target actively pursued.

To connect with the Ethereum Wallet, you'll need to initialize your private network first via Geth as the wallet does not currently support calling Geth directly. To initialize your local chain, download <code>{{.GethGenesis}}</code> and run:
<pre>geth --datadir=$HOME/.{{.Network}} init {{.GethGenesis}}</pre>

With your local chain initialized, you can start the Ethereum Wallet:
<pre>ethereumwallet --rpc $HOME/.{{.Network}}/geth.ipc --node-networkid={{.NetworkID}} --node-datadir=$HOME/.{{.Network}}{{if .Ethstats}} --node-ethstats='{{.Ethstats}}'{{end}} --node-bootnodes={{.BootnodesFullFlat}}</pre>

You can download the Ethereum Wallet from https://github.com/ethereum/mist/releases.

<h2> Mist browser <small>Interacts with third party DApps</small></h2>

The Mist browser is an Electron based desktop application to load and interact with Ethereum enabled third party web DApps. Beside all the functionality provided by the Ethereum Wallet, Mist is an extended web-browser

where loaded pages have access to the Ethereum network via a web3.js provider, and may also interact with users' own accounts (given proper authorization and confirmation of course).
Under the hood the browser is backed by a go-ethereum full node, meaning that a mid range machine is assumed. Similarly, synchronization is based on fast-sync, which will download all blockchain data from the network and make it available to the wallet. Light nodes cannot currently fully back the wallet, but it's a target actively pursued.

To connect with the Mist browser, you'll need to initialize your private network first via Geth as Mist does not currently support calling Geth directly. To initialize your local chain, download <code>{{.GethGenesis}}</code> and run:
<pre>geth --datadir=$HOME/.{{.Network}} init {{.GethGenesis}}</pre>

With your local chain initialized, you can start Mist:
<pre>mist --rpc $HOME/.{{.Network}}/geth.ipc --node-networkid={{.NetworkID}} --node-datadir=$HOME/.{{.Network}}{{if .Ethstats}} --node-ethstats='{{.Ethstats}}'{{end}} --node-bootnodes={{.BootnodesFullFlat}}</pre>

You can download the Mist browser from https://github.com/ethereum/mist/releases.

<div id="connect-go-ethereum-mobile" hidden style="padding: 16px;">

<h3>Connect Yourself – Go Ethereum: Android & iOS</h3>

<h2> Android devices <small>Accesses Ethereum via Java</small></h2>

Starting with the 1.5 release of go-ethereum, we've transitioned away from shipping only full

blown Ethereum clients and started focusing on releasing the code as reusable packages initially for Go projects, then later for Java based Android projects too. Mobile support is still evolving, hence is bound to change often and hard, but the Ethereum network can nonetheless be accessed from Android too.
Under the hood the Android library is backed by a go-ethereum light node, meaning that given a not-too-old Android device, you should be able to join the network without significant issues. Certain functionality is not yet available and rough edges are bound to appear here and there, please report issues if you find any.

The stable Android archives are distributed via Maven Central, and the develop snapshots via the Sonatype repositories. Before proceeding, please ensure you have a recent version configured in your Android project. You can find details in Mobile: Introduction – Android archive.
Before connecting to the Ethereum network, download the <code>{{.GethGenesis}}</code> genesis json file and either store it in your Android project as a resource file you can access, or save it as a string in a variable. You're going to need to to initialize your client.
Inside your Java code you can now import the geth archive and connect to Ethereum:
<pre>import org.ethereum.geth.*;</pre>
<pre>
Enodes bootnodes = new Enodes();{{range .BootnodesLight}}
bootnodes.append(new Enode("{{.}}"));{{end}}

NodeConfig config = new NodeConfig();
config.setBootstrapNodes(bootnodes);
config.setEthereumNetworkID({{.NetworkID}});
config.setEthereumGenesis(genesis);{{if .Ethstats}}
config.setEthereumNetStats("{{.Ethstats}}");{{end}}

Node node = new Node(getFilesDir() + "/.{{.Network}}", config);
node.start();
</pre>

<h2> iOS devices <small>Accesses Ethereum via ObjC/Swift</small></h2>

Starting with the 1.5 release of go-ethereum, we've transitioned away from shipping only full blown Ethereum clients and started focusing on releasing the code as reusable packages initially for Go projects, then later for ObjC/Swift based iOS projects too. Mobile support is still evolving, hence is bound to change often and hard, but the Ethereum network can nonetheless be accessed from iOS too.
Under the hood the iOS library is backed by a go-ethereum light node, meaning that given a not-too-old Apple device, you should be able to join the network without significant issues. Certain functionality is not yet available and rough edges are bound to appear here and there, please report issues if you find any.

Both stable and develop builds of the iOS framework are available via CocoaPods. Before proceeding, please ensure you have a recent version configured in your iOS project. You can find details in Mobile: Introduction – iOS framework.
Before connecting to the Ethereum network, download the <code>{{.GethGenesis}}</code> genesis json file and either store it in your iOS project as a resource file you can access, or save it as a string in a variable. You're going to need to to initialize your client.
Inside your Swift code you can now import the geth framework and connect to Ethereum (ObjC should be analogous):
<pre>import Geth</pre>
<pre>
var error: NSError?

let bootnodes = GethNewEnodesEmpty(){{range .BootnodesLight}}
bootnodes?.append(GethNewEnode("{{.}}", &error)){{end}}

let config = GethNewNodeConfig()
config?.setBootstrapNodes(bootnodes)
config?.setEthereumNetworkID({{.NetworkID}})
config?.setEthereumGenesis(genesis){{if .Ethstats}}
config?.setEthereumNetStats("{{.Ethstats}}"){{end}}

let datadir = NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask, true)[0]
let node = GethNewNode(datadir + "/.{{.Network}}", config, &error);
try! node?.start();
</pre>

```html
</div>
</div>
</div>
</div>
</div>
<div id="about" hidden>
<div class="row vertical-center">
<div style="margin: 0 auto;">
<div class="x_panel">
<div class="x_title">
<h3>Puppeth &ndash; Your Ethereum private network manager</h3>
<div class="clearfix"></div>
</div>
<div style="display: inline-block; vertical-align: bottom; width: 623px; margin-top: 16px;">
<p>Puppeth is a tool to aid you in creating a new Ethereum network down to the genesis block,
bootnodes, signers, ethstats server, crypto faucet, wallet browsers, block explorer, dashboard and
more; without the hassle that it would normally entail to manually configure all these services one
by one.</p>
<p>Puppeth uses ssh to dial in to remote servers, and builds its network components out of
docker containers using docker-compose. The user is guided through the process via a command
line wizard that does the heavy lifting and topology configuration automatically behind the
scenes.</p>
<br/>
<p>Puppeth is distributed as part of the <a href="https://geth.ethereum.org/downloads/"
target="about:blank">Geth &amp; Tools</a> bundles, but can also be installed separately
via:<pre>go get github.com/ethereum/go-ethereum/cmd/puppeth</pre></p>
<br/>
<p><em>Copyright 2017. The go-ethereum Authors.</em></p>
</div>
<div style="display: inline-block; vertical-align: bottom; width: 217px;">
<img src="puppeth.png" style="height: 256px; margin: 16px 16px 16px 16px"></img>
</div>
</div>
</div>
</div>
</div>
<div id="frame-wrapper" hidden style="position: absolute; height: 100%;">
<iframe id="frame" style="position: absolute; width: 1920px; height: 100%; border: none;"
onload="if ($(this).attr('src') != '') { resize(); $('#frame-wrapper').fadeIn(300); }"></iframe>
</div>
</div>
</div>
```

```
</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/3.3.7/js/bootstrap.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/gentelella/1.3.0/js/custom.min.js"></script>
<script>
var load = function(url) {
$("#connect-go-ethereum-geth").fadeOut(300)
$("#connect-go-ethereum-mist").fadeOut(300)
$("#connect-go-ethereum-mobile").fadeOut(300)
$("#about").fadeOut(300)
$("#frame-wrapper").fadeOut(300);

setTimeout(function() {
if (url.substring(0, 1) == "#") {
$('.body').css({overflowY: 'auto'});
$("#frame").attr("src", "");
$(url).fadeIn(300);
} else {
$('.body').css({overflowY: 'hidden'});
$("#frame").attr("src", url);
}
}, 300);
}
var resize = function() {
var sidebar = $($(".navbar")[0]).width();
var content = 1920;
var limit   = document.body.clientWidth - sidebar;
var scale   = limit / content;

console.log(document.body.clientHeight);

$("#frame-wrapper").width(content / scale);
$("#frame-wrapper").height(document.body.clientHeight / scale);
$("#frame-wrapper").css({
transform: 'scale(' + (scale) + ')',
transformOrigin: "0 0"
});
};
$(window).resize(resize);
```

```
var item = $(".side-menu").children()[0];
$(item).children()[0].click();
$(item).addClass("active");
</script>
</body>
</html>
`

// dashboardMascot is the png dump of the mascot to display on the dashboard about page.
var dashboardMascot =
[]byte("\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x01s\x00\x00\x02\x00\b\x06\x00\x00\x00p\xe4\x8c`\x00\x00\x00\x06bKGD\x00\xff\x00\xff\x00\xff\xa0\xbd\xa7\x93\x00\x00\x00\tpHYs\x00\x00\v\x13\x00\x00\v\x13\x01\x00\x9a\x9c\x18\x00\x00\x00\atIME\a\xe1\x03\x1d\x0e0&\xf3\xca\t\x11\x00\x00\x00\x1diTXtComment\x00\x00\x00\x00\x00Created with GIMPd.e\a\x00\x00
\x00IDATx\xda\xec\xbdw|\x15U\xfe\xff\xff<3s[\x12BB\x12H!\x90\u0411&H\a\x01i\"V\xb0!b\xc1^>\xbb\xee\xae\xdd\xd5u\xed}-
k/X\u058e\xa8\xa0\u049bH\x87\bH\xef\x04BI\x81\xf4[\xa6\x9c\xf3\xfbcnB`\xdd\xef\xcf\xdd\xd5\xddU\xe7\xf9x\\\u023d\u027dw\xee\u0339\xafy\xcd\xfb\xbc\xcf\xfb\r\x1e\x1e\xff\";wn\x17\xb7\xdf~\xabQw_)\xa5+\xa5\x1a)\xa5\u0494R\xc9J\xa9D\xa5T\xf0\xb8\xa7\xe9\x1f|\xf0\x9eQYyXx{\xd0\xc3\xe3\xc7\xc3\xfbBy\xfcK\xbc\xf3\xce[\xe2\xd2K/Wq\x11o>y\xf2G'\xaf[\xb7\xae\xff\u05ad\xdb2\xa5\x94\xa9\xd1h\xb4:---
\u06b9s\xe7HJJ\xf2\xb2f\xcd27\x9e}\xf6\x98\"!\xc4\u07ba\xd7x\xf6\xd9g\xb4\xcb.\xbbD\xa6\xa4\xa4\xb1d\xc97\f\x18p\xb2\xb7c=<<1\xf7\xf8O\xf0\x97\xbf<EII\x89x\xf4\xd1\xc7\x14\xc0\x8c\xe9_\x0e\xffv\u035a\xa7\u05ec]\u05f9`\xf5jJK\xcb0M\x13M\xd3\b\x85B4i\x92J\xabV\xf9\xe4\xe5\u54d9\x99Y\x90\x9a\x9a2\xa5_\xbf\xfeS\xfb\xf5\ubfe9\xee5o\xba\xe9V\x9e}\xf6\t\x94R\b\xe1\rI\x0f\x0f\x0f\x8f\x9f\x9c\xf7\xde{\xb7\x81;\u007fs\xe4\xc4+.\xafi\u04e6\x8d\x02\xfe\xe1M\xd75\x95\x9c\x9c\xacZ\xb7n\xadF\x8d\x1a\xa5\u018d\xbbp\xd7=\xf7\xfc\xf1\xf5\xaf\x17.\x18q\xfc\xeb\u007f\xfa\xe9'\xdeN\xf6\xf0\xf0\xf0\xf8\xa9\xc9i\x9e\v\xc0\x97_N\x1bq\u0265\x97\x1c\xceh\u06b4^\xb4\x85\xa6)\xc3\xe7S>\xbf_\x89\xef\x15u]\x86B!\x99\x93\x93\xa3\xfa\xf6\xed\xab.\xb8\xe0\x82\x8a\a\x1e\xb8\xef\x83}\xfbv\xf7k\xf8\x1e\u007f\xf8\xc3\x1f\xc4\u0739\xb3\xbc\x9d\xed\xe1\xe1\xe1\xf1S\xb0r\xe5r\r@)\xd5\xee\xf6\xdbo\u06d2\u07fa\xb5\x02$\xa04]W\x86\xe1S\t\u024dUjV\xaeJ\xcdn\xa9\x12S\u04d5\x10\xe2\x18A\x17B(\xbf\xdfo'5jde\xe74W\xfd\xfa\xf5WW^yE\xe5k\xaf\xbf\xfa\xa0R*\xb1\xee\xbdz\xf7\xee#\xdex\xe35o\xa7{x\xfc@\fo\x17x\xfc\x10\x94R\x9cz\xeaH\x00\xfe\xf6\xb7\xb7\xaf\u0634yK\xfb\xdd;w\u0680\xa1\xe9:\x86\xcfO\x93\xdc\xd6t\x18r\x06\x19m\xbb\xa0\xe9\x06f\xd5\x11\xcavmd\xff\xa65\x1c\u06be\x91pU\x05J)L\xd3\xd4\x1d)\xb1\x1d\u01e9\xae\xae\xe1\u0421\x83\u027bw\xef\xfe\xe3\x96\xcd[\u03999s\xfam\xa3F\x8d\x9e\xber\xe5\n\xb5r\xe5\n>\xfa\xe8\x03q\xe1\x85\x17)\xef\bxxxx\xfc\b\u0318\xf1U\x9d+O\xbe\u66ab\xe7\xb6\xcc\xcbW\x80\x85\x10J7|\xaay\x97^\ua497\xa7\xab{\xd7Ku\xcfw\x8e\xbao\x83\xa3\x9e\xd8\"\xd53[\x1d\xf5\xe0\xd2\xfd\xea\xca\x17>Q\xfd/\xbcZ\xa5\xe5\xe6\xff]X&!1Q\xa6\xa5\xa5\xa9\xf6\xed\u06ebs\xce9[=\xfa\xe8#\x1f\u06b6\x99Y\xf7\u0793&\xbd\xe1\x99\x0e\x0f\x0f\u03d9{\xfc\x18l\u0630A\x03\xe4\xf4\xe9\xd3z\x95\x1d>\xd2\xfdPq1\x80.4\x8d`R2\x9dF\x9eG\xab\xc1\xa7\x11\x8dJ|\xd2B\xf3\t4\f4M#%3\x9b\x93\xce<\x97\x13O=\x97\xc2u+Y\xf1\xc9$VM{\x8fhm\r\xb6\x94H\xc7\x11JJu\xe8\xd0!QYYIY\xd9\xe1\v\v\v\v{\xbf\
```

xfa\xea\xcb\u007f\xbc\xfa\xeak?\x12B\xd8c\u019c\xa3=\xf0\xc0\xfd\xaas\u7b9eK\xf7\xf0\xf0\xc4\xdc\xe3\xdf\b\xb3\u803dq\u00e6v\xa6i6\x89E\xc2\x12\xd0p\x1c2\xf2\xdb\u04a6\xcf`t\xc0pb\x04|\x02\x03\xb0-
\vi\x83\xa6\t\x10\x02C\xf7\u0476ooZu\xefM\x87\x81#\x98?\xe9iv\x16,AJI$\x12\x11J)4Mc\u02d6-
\x1c<x0\xff\u0421C\xefWTT\x9c\xa7\x94\xfa\x83\x10\xa2\xf0\xb3\xcf>\xe7\xc9'\x9f\x10II\x89\xea\xba\xebn\xf0\x0e\x8a\x87G\x03to\x17x\xfc\x10\xe6\u0319\xeb(\xa5\xf4\x85\v\x17\\\xb5x\xf1\u0493JJJ\x00\x84\xa6\xe9\xe4\x9f4\x90^\xe7\\\x8a\xdf\x1f\xc4'\x1c4\x01
@\x80B\xe1(\x85t\x14\xb6\xed`\xc5\x1c\x84\xd0\xc9\xed|\x02\xed\xfb\x8d\xc4\x17\brp\xebz,3\x86m\xdbH)\xd14\r\u02f2\u063f\u007f?\xd5\xd5\xd5\x1d\xb7n\xdd:\xfc\xfd\xf7\u07dd\xf3\xf4\xd3\xcf\x1c\x99={\x0eYYYb\xfc\xf8q\u031e=\xc7;0\x1e\x1eq4o\x17x\xfc\x13\xe4X\x96\u0569\xa2\xa2<\xae\xd6`\x04\x83\xa4d\u5498\x92\x8a\x81D\x17u+\xd1\x14(\x85T
\x158\xb87KB,\x16\xa5\xb6\"L\x93\x9c\xe6\x9cu\xcb\xc3\\\xf9\u0707\xb4\xee\xd1\x1f\x00\xd34\xa9\xad\xad\xc54M\xa4t\xd4\u06b5\xeb\x983gn\xe7G\x1ey\xec\u04d2\x92\x03\xb9\x00\xaf\xbd\xf6\xba*--
\xf5V\x17yxxb\xee\xf1\xafP[Se\x16\x15\xed\x8fX\x8e\x13Wk\b$$\xd18\xb39\x9a\x06\x02\x85&@\x8b\vz\xc3\u015c\ueb27B\x01\xb6\u0490B'\\S\x8be\xd9t\x191\x8a\u02df\xf9\x80\xbec/\x03\xc0\xb6mjjj\x88FcB\b\xa1v\xed\xda\u0152%K\xbb\xfc\xe1\x0f\xb7~\xban\u0777-
\x00\x1e{\xec\t5c\u0197\x9e\xa0{xxb\xee\xf1\xcfR\xb4\xbfH\x84\xc3a\u0371\xed\xfa\xc7|\x81
\xa1\u4538\x11W8J
\x95\xa83\ueba0\xbb&\x1d\xea\\\xbar\x1d\xba\x12:\u04b1\xa9\xad\xb6i\u05a6\x05\xe3\x1fy\x83\xb3n~\x88P\xa3d\xa4\x94\xd4\u0506\tG\"BJ\xa9\n\v\vY\xb7n]\xcfG\x1ey\xec\x8b\x193\xbe\xec\fp\xdaig\xa8\r\x1b\xbe\xf3B\x85\x1e\x1e\x9e\x98{\xfc3\xa4\xa4\xa4\xa0i\x9aRR\x1e}P\b\x94Tq\xe7-
\u0730J\u0703\u02fa$\u0138So\x98\x86\"\x15\xd8R\xa0\x14(\xe9P]a\xe2O\xd09\xf5\x86\xbb\xb8\xe0\xde\x17H\xcdj\x8e\x92\x0e\xe1p\x98H4*\x14\xa8\x83\a\x0f\xb1a\u00c6\xae\xef\xbf\xff\xc1\x97\v\x16\xcc;\x1f\xa0s\xe7\xae\xce\u0319\u04fdq\xec\u1279\xb7\v<~(\u035ae\xeb)\xa9\xa9\xbe:\xd3\r`\x9b&\xe1\x8a#q]\x17\xf5\x95\xdb\xd4q\xff\x1f\r\xb7\xa8\xfa\u01dc\xb8\x93\a\xd7\xd5\xd7VE\xb1\xa5\xa2\xf7y\x13\x18\xf7\xd0\xeb4ks\x02(E4\x12!\x12\x8d\n\u02d1\xeaPq\t\u02d7\xafh\xf9\xe1\a\x1f~<s\xe6\xf4\xdb\x00F\x8d\x1a-
\xdfy\xe7m\xf1\xd9gS\xbc\x83\xe4\u1279\x87\xc7\x0f\xa0*))\xb1\xd8\xe7\xf3\xd7\xeb\xb4\x15\x8dPY\xb2\x1f%\xeb*\x1e\xaa\xb88\xbb\xee[\xd5\u074e\x13\xf6\xba\xfb\x96t\x1d\xbc\x16\u007f\x8e\x15\x8b\x11\x8b\xdat\x1av*\xe3\x1ez\x9d\xdc\u03bd\x000\xa3Qj\xc3a\x11\xb5lJ\x0f\x973c\xe6,\xde\u007f\xff\x83\u01fe\xf8b\xeam\x00\x97^z\x99Z\xbf~\x83\x98<\xf9#\xef(yxb\xee\xe1\xf1\x8f0\f\x9f.\x84\xa8\xb2L\xab --
\r\xdc:+8\xb6EU\xc9Aj+\xcaA\xd3P\xc7LI\n\xb7p\xcbqB~\u053d\v$\x02[\xb9~]\x13\x02M\b\xa4m\x13\xad1i\u0777\x1f\x17>\xfc:\xad{\x0f\x06\xc0\x8eE\x89\x86\u00d8\xb6Cyu\r\xf3\xe7/\xe0\x93O\xa6<\xb6h\xd1\u009b\x00\xfe\xf4\xa7{\u055a5k\u0164I\xaf{\a\xcc\xc3\x13s\x0f\x8f\xef\xe3\xfe\a\xfe\xac\x03\xb4i\u04fa\xbcY\xb3f\xf5\x9a,\x1d\x87\xaa\u0483T\x1c\xda\x17\x9f\xect\xeb%\xba\x99,\xea8\xf9v\u007f\xaf\xe2.\xbc\xee/l)p\x94h\x10W\x17\xeeB\xa2\xea(\xb9\u077ar\xc1C\xaf\xd3a\xd0i\xae\xa0\x9b1\xccH\x18\u01d1T\xd4\u052a\xb9\xf3\x17\xf0\u059bo=\xb3`\xfe\u071b\x00\x1e~\xf8\x11\x15\x8b\u017cq\xed\u1279\x87\xc7?\xc0\x01h\u07fe\xfd\x16M\xd3\x0e\x01B)\xa5P\x8a\xea\xb2b\x8e\xec\u0749\xe1sE\xbc\xae\xc1\x84\x10
\x8e\x9f\xfa\x14\xaa~R\xb4>$\x03\xd8Ra\xab\x86a\x18\xe1\xc6\u02ebbd\xb5o\xc3y\xf7\xbfD\x87\

x93G\xb9\x82n\x99\u0122a\x1c\xa9DyU\x8d\x9c5g\x1e\x93&\xbd\xf9\xccg\x9f~\xf2;\x80\ubbffQ~\
xfa\xe9'\xde\xd8\xf6\xf0\xc4\xdc\xc3\xe3x\xfa\xf4\xee\xad\x00\x86\x0e\x1d\xbe==#cKBR\x12\xae\
xc1\x96\xd4\x1c.\xe6\xd0\xf6\x8d\xf1\xa5\xfb\xc7\x0f\xa9\xe3\x03,\xc2\x15\xf4\x06\x8f\v\x14J\x81\
x1d\x8f\xb3\x83B\x13\n\x81\xe6\nz\xb5Ef\xbb\x96\x9c\xff\xd0kt\x19y\xae{f1M\xccH\x18\xe98Zyu\x
ad\x9c\xbb\xe0k>\xfc\u88e7\xdf{\xf7\x9d\xdb\x01\u018e=O\xae^\xbd\xd2K[\xf4\xf0\xc4\xdc\u00e3!
C\x87\x0e\x97\x80\x10B\x1c\xc8i\xde|k\x8b\xdc\\\x00[I\x89\x15\rS\xbag;U%\xc5\xe8\x86\xcfm\xff\
u01b1\xee\\\x1c\u04e1P\x1c\xa3\xf3*\x1e\x9d\xc15\xfa\rD\x1f4\xcdM]\xac=b\x91\u05a29c\xef}\x81
\ue9cfs\x05\xdd2\xb1\xcc(J)\xad\xaa6\xa2\xbeY\xb2\x8c\xaf\xa6O\u007ft\xea\xd4Oo\a\xe8\u0673\
xb73c\xc6W\xde\xc2\"\x0fO\xcc=<\xea\xe5W\b\x9ey\xe6\x19\r\xa0u~\xfe\xdaF\x8d\x1b\x03\xf8\xa
5m#\x1d\x87\xe2]\x9b9\xb4s\x13\xbe@\xbd<\xd7GR\xc41.\xbdAz\v\r&G\xd5\u047fQ\xf5r\xee\xfe\
xa4\t\x01\xca!V\x15%-
\xa7\x19c\xee~\x8e\xeeg\\T\xef\u042dh\x18)\xa5\xa8\xac\x8d\xb0h\xf1R\xbe\xf8\xe2\xabG\xe7\u0
39d5\x01\xe0\xb4\xd3NW\x0f<p\x9fX\xbdz\xa5w\x10=<1\xf7\xf0\x00\xc8\xcfo)\x01.\xba\xf0\xbc\x9
5\xe9\xe9\x19\xbb}\xfe\x80+\xbfJq\xa4h\x0fE\x1b\v\xb0M\xd0u\xbd\x81\xcbv\xe5\xf9\xe8\x9a\xd0:
\xbb\x1e\x97\xeacD\xfd\xa8\x84\xcb\xe3\x025Z<\x16\x1f\xae\x8a\x92\x9c\x95\xc1\xd9w=\u0349\x
a7]\x00\xb8\x93\xa2V4\x8cr$\xe5\u0575\xcc_\xf05\x1f~\xf8\xf1\x13\u02d6~\xd3\x15\xe0\x9e{\xee
U_\u007f\xbdH\x1c<\xb8\xdf;\x88\x1e\x9e\x98{x\x9cr\xca)\n
\xb5I\u01b7\x1d\u06b7\u06d9\x95\x95\x05\xa0\xa4ccFj)\\\xb3\x8c#\xfbv\x11\b\xf9Q\xaa.\xa5\xa5\
ue98e\xae5\xaa_\x19ZW\x95+.\xe0\xf1\xa5\xfeu\x81\x96\xfa\xfctE\xbcD\x80[N7Z\x15%5\xbb\x19
g\xde\xf54\x9d\x86\x9fsT\xd0\xcd(JA\u0251\n\x96._\x99\xf9\xde\xfb\x1f\xbe\xab\x94\xca\x03\xb8\
xf9\xe6[\u051bo\xbe\xe5\x1dD\x0fO\xcc=<\x92\x93S\xea\u007f\xee|B\x87\u03dbff\x02\b\xe56\x98
`\xef\x86\x02\xf6\xac]\x1e\x17]\xcd\xf5\xe4B\x1d\xe3\xb0\x05G\x8bn\x1d\x1bEw\x05\xdd]\x15\x1a\
x1f\x9c\xc2\x1d\xa0\r\x9f\x03\nM\x83hu\x94\xf4\xdcl\u03be\xebiZ\xf5r\xf3\u042dh\x04;\x16E*\xa5\
xf6\xec\xdb\xcf\xeao\xd7ty\xe4\xe1\x87\xea\x15\u0732L/~\xee\xf1\x8b\u015b\xed\xf7\xf8\x97\x98:
u\u06bey\xf3\xe7\x8f(\xda\u007f
3\x16\x8d\x80t\xb0bQ\x1c\u01e1y\xe7\xde$\xa7\xa5#m\xbbA\xce9GeY\x88\xfa\xf4E\x94:\xb6\xbc
b\xbd+\x17\xf5\xa1\x97\xba\x12\x00uN]w\xb3\x16\xb1mh\u04bc\t\xcd\u069e\xc4\xfe\xf5\xab\xa9*9
\x80\xe3\xd8h\x9a&\x10B\x95\x1d>,L3\x9a\xf7\u007f7\u07980{\xf6\xec\xb9\v\x17~\r\xc0k\xaf\xbd\
xca\x17_|\xe1\x1dD\x0f\u03d9{\xfcz\xb9\xe1\x86\xeb]\xc1\x15\xe2\u0420\x81\x03\x16d\xba\xee\x
dc\xcd\x15\xb7,\xf6|\xbb\x94\xdd\x05\x8bQJa\xf8|GE\xbb>\x98\u00b1\xf1\xf3\xfag\x1f]`TW\xac\xc
bQ\x02[\x1d\x9f\xc4\xe8.0B\b\x94c\x13\xaeth\u0465\vg\xdc\xf9\x14i-
\u06c0RX\xb1\bJ\xda\xc2r\xa4Z\xb7~\x13\x05\x05\x05\xb7\u035e=}\f\xc0\x94)\x93\xf5\xab\xaf\xbe
\xc6;\x90\x1e\x9e\x98{\xfc\xba\xb9\xf6\u06ab\xeb\u007f>}\xf4i\x9ft<\xa1\x83\n%$\xba\x8a\xac$\xe
1\x8a26\xcd\xff\x9c\u0292\x03\xf8\x03:\xba\x00\xedh\xbb\x8a\x06\x01\x95\x06%\x15\x1b8\xf2\xba
`\x8a#AJ\x15\xaf\xaa\x18\xbf\xc5\v\xbe\xd4M\x90\n\x01\xca6\x89\xd5Z\xb4\xedw2\xa7\xfd\xfea\x1
2S\u04d0\x8e\x83m\xc6\x10(Q\x13\x8eX\x9b6oe\xc6\xf4YW\x01\x9c{\xee\xf9\xce\u0295\u02fdq\x
ef\u1279\u01ef\x9bn\xddz0y\xf2\xc7\x00\xe4\xe7\xb7^\u04a3[\xb79\xd99\xd9\xf5Z,m\x8b\u00b5+(
\\\xbb\x1c!\xc00\f4\xa1\x8e6\xach
\xe0\xa2\xc1?nhE\"m\a\u01f4pL\x13\u01f6\xdd>\xa2\xf1a\xea\x98Ql3\x86m\xd9HG\xba\xb1u!P\x8
e\x85\x15u8q\xf4\xf9\f\x9ax\v\bp,\v\xdb4\xd1\f\xdd\xd8Y\xb8\x97\xad\xdbw\x8c\xfe\xf8\xe3\x0fo\x0
4\xe8\u077b\xaf\xfa\xee\xbb5\xde\xc1\xf4\xf0\xc4\xdc\xe3\xd7\xcd\xf9\xe7_P\xffs\xcb\x16-
\x1e\xec\u043e\x1dqw\x0e@e\xf1~\xd6N\xff\x98\u02b2rt\xbf\x8e@\xb8\x0e\xbd~y\u007f\u0709\xd

7?\xa6PR\"-
\xcb\x15k3\x86c\x9b\x14\x16,a\xf6Sw1\xf7\xaf\xf7rp\xdb\x06\x84\xeesE:\x1a\x8d\xf7\fu\x90\n\x84\xa6\xe3X&\n\xe87\xfez\xba\x8d\x1eW/\xfe\x8ei\nG*\xb9u\xfbN\xbe-
\xf8\xf67J\xa9\x96\x80\x9a9s\xb67\x19\xea\xf1\x8b\u009b\x00\xf5\xf8\x97\x989s\x86x\xf7\xdd\xf7\xf8\xec\xb3\xcf\x0f.[\xb6\xb4WQQQ\x9b\xca\xca\xca\xfa\xdf\xd7\x1c)\xa1i\xab\x0e4?\xa1\x13\x8eS\x97\x9a(\xe2\xf9\u3abe\xad\x9c\x06(i#-\vi\xdbH\xdbB)\x89c[\xac\xf9\xfcmV|\xf0\x02\xfb\xd6-
\xa7t\xf762\xdbu%9#\x13;\x16E9\x0e2\xde$C\x17\x02M\u05f0-\x8bPr\x12\xa9-
:\xb0k\xd5\"j\x8f\x94
\x1d\x1b\xc3\xf0\x8bp4\x8a\xa6\x91n\x9b\x91\xa2i\u04fe\\6g\xce\\f\u039c\u03bb\xef\xbe\xe7\x1dL\x0f\u03d9{\xfcz\x195\xea4w\xe1\xbd\x10V\x97\u039d\xa6\xb6i\u075a@0T\xff\xfb\xda#\xa5\xac\xfd\xea#*\x8b\xcb\xf0\x05\xf4z'\xae\x8b\xa3\v\x88\x04\xa0\xa4\x04\xdbF\xd9\x16\u04b6\x90\x8e\x83\xb4,t\xdd 5;\x8fPr*\x00\xbbW.\xe4\x9bIOR]V\x8c\xee\xf3!-
\x13\u01ccaG\xa3\u0626+\xee\x9a\xd00\xc3&M;t\xa6\xcfe7\xa3\xf9\xfc()1c\x11\xa4\x92l\u0676\x83\xd5\x05k\xae\t\xd7V\xe7\x02L\x9c8\xd1s\xe7\x1e\x9e3\xf7\xf0\xa8\xe3\x8b/\xbe(.(X5\xbch\xff\xfef\xb5\xe1\bR:\x00T\x1c\xdaKJ\xb3\x1cZ\x9e\xd8\vi+\x94\x92\xf5\x8b\x81\xa4\x02\xa9\x14\u02b6Q\xb6\x8dt,\x1c\xc7A92\xbezT\x91\x9e\xd7\x16\xcd0(\u0779\x19\xc74)?\xb0\a#\x98@\xf3N'!m\x1b%%B\xc9x-
\x18\x88w\x95F\t\x1f)\xf9\x9d)\u07fb\x9d\xd2\xed\xebQn\x03j,[\xe2\xd8vZ\xf9\x91\xb2\xb2\xf9\xf3\x17|SSS\x8bR\x92\xfb\xee\xbb\xcf;\x88\x1e\x9e3\xf7\xf8\xf5\xb2c\xc7\u05b89\xd7\xf6\xf5\xec\xd9\xf3\xf5.\x9d:\x11\b\x06\xd0\r\x03\x003\\\u02ea\xcf\xdf\xe1\u040e\x1d\x18A\xb7\x00\x97T\x10\xb3%Q\xcb\xc1q$B)7\\\"\x15\x9a\xaa\x13g\x89c\x99\x18\xfe
}\xc6]\xcf\xc0\x897\u04e8i6V4\xc2\xe6yS)Z\xbf\x1a\xdd\x1f@96R\xd6M\x9a\x9a(+\x86\x94\x92X$\x8a\x11\xf4\xd1k\xfcoIHM\a\xeaV\x88\xc6\xd8[T\xc4\xe6-
[\xafUJ\xba+\x9e\x84\xf7\x15\xf0\xf0\x9c\xb9\u01ef\x9c\xe7\x9e{\x9e/\xbf\x9cf\xbc\xff\xfe\a\xf2\xb3\xcf>\xdf\xf5\xdd\xfa\xef\x86\x1e<x(\xbb\xaa\xaa\xcaQRjJ)*\x0f\x15\xe1\x0f%\u043a\xcf0\x14\x02\u06f6\xb1\xe2\xe9\x85~\rt%\x91\xb6\x8d\xe3\xd8
\x9dz\xc1\x17\n\xa4c\xa3\x19\x06\xd9't\xc7\x17J\xa0d\xc7F\xaa\x0e\x15\xa1\x80\xecN=1\x82!pl4\r\x10\xae\x9b\a\x81\xe9\x80\x14\x06\x8d\x9b\xe6Pyp/\a7\xaer\xdd~<\x97\xc6\xd0\xf5\u0186&J\xa7N\x9d\xb6\x04`\xf1\xe2EL\x9a\xf4\xa6w@=<g\xee\xf1\xeb\xe5\xf4\xd3\u03f4;th\xaf\v!JO<\xb1\xdbcy-
[\u0621PH7|>Y\xe7zWN\x99\u0136of\x12Lr\x1d\xbbO\x17\x04}\x1aZ\xbc\u035c\xc2\x15b'\xde4T(U\xbfX\u050aF\x90\xb6M\x97Q\xe7\xd3\xed\x8c\xf1\x18\xc1\x10;\x97\xcfg\xc7\u04b9h\x9a\x8e\x00\xa4\x94\u060eD:\xd2\x15w\xe5\x80\x19!\x10\xd4\xe8|\xda8B)\xae;\x97\xb6I,\x16\xa5\xb8\xb8\x84\u056b\v\xaePJ5\x05\x188p\x90w
=<1\xf7\xf8al\u07bc\xe1\u07dal[\xb3\xa6\xe0\u007f\xb2\xea\x9f\x10\x82\u0673g\xba\xd5\x14/\xbaxr\xabV\xf9S\xf3Z\xb6@\xd34G\xb8\xa5\x0e\xa99\\\u02827\x9e\xe4\xf0\xde\x03$$'\xa0\xa1\u0705D\x02\x948Zd\xcb\xed\x17\x1a_\xbc_W\x17W)\xcch\x18!4:\x8f<\x8f\xfc\x9e\x83\x88V\x1da\xc3\xec)\x1c.\u070e\x11\f\xe1H\x85\x94\xf1\xd5E\xd2A96\xb6\xe3\x10\x8bJ2\xdau\xa5\xc3\u0433\u074dU\n\u01f69|\xa4\x9cM[\xb6\xb4\xff\xec\xd3O\xeas,\xa7O\xff\xd2\x1b\xa4\x1e\x9e\x98{\x1c/\xdc\x1b\xffN\xb8;v\xec\xac\xfe\x9d\xd7\xec\xde\xfd$\x95\x95\x95\xf3w\x8f\xef\u06f7\xe7\xbf\xfaY\x8b\x8a\xf6\u04a2E\xbe\x9a>\xfdK\r\u0b33\u039a\x94\x97\x97\x17m\x94\x94\xe4\xf3\x19>U\x97W\xbec\xf9|\xbe\x9e\xf4$2f\x13LH<\xdaRN\xd3\x11\xba\x01B\xe0H\x85-

\xdd0\x8b\x8aOl\xd6aE#4JoF\xd7\xd1\xe3Hm\u078aC\u06fec\u04c2/\x88\x86k1|~\x90\x12\u02d1\x98\x96\x83m\xdb\b\xe9\xe0\xc4\xc2\x04\x1b5\xa6\u07503\xf1\x05\x13\\w\xee\xd8D\xa2QJJ\xcbX\xb0p\u163a\xd7\x1f=\xfa\fo\xe0z\xfc\xac\xf1R\xb3\xfe\r\x94R\u031f?W[\xbbv\x9d\xb6k\xd7N\xfd\x85\x17^\x8a\x1d\xf7\xfb\x04)\xed\xc0\u0739s\xd2g\u03de\xa3L\xd3N\xea\u07ff\uf165\xa5\xa5I\u06f7\uf215\x96\x96\x12\x0eG\xb0m7\x93C\xd7u\f\xc3
11\x91\xa6M\x9b\x92\x97\x97\xe7\xcf\xc8H\x0fo\u077au\xea\x91#\x87\xcbN9\xe5\x14\u0577o\u07f2\x9c\x9c\x16\xa6\x10\"|\xdc\xe6\x18\x8f=\xf6\xa8\xc8\xc9\xc9v.\xbe\xf8\x12)\xc4\u007f\xef\xd0>\xf8\xe0\xfd\x0f|\xf5\xd5\xf4\xbb7n\xdeJ$\x1cV\x96e\n\xe2\x19'g\xdd\xf9\x17\x06^z\x13\b\x81m\u0190\nl\xd3$V[K4\x1aA\xd96B\x88x\xfdrw\x88\xaa\xb8\xab\xd6t\x1dG\x18,\xfb\xf8u\xd6M~\x89\x84\x944\x86\xff\xdf}\xb4\xee=\x18+\x16\xc1Q\x02\x89@7\f\x94\xe1G\xe9>\xfc\t\x89\u012a\x8e0\xe3\xc1\xeb\xd9<\xf73\x00t\u007f@\xa56N\x16}\xfb\xf4\xae\xbe\xf7O\xf7\\\u052bw\u07ef\x00\xbe\xfbn\r]\xbbv\xf7\x06\xb6\x87'\xe6\xbf&\xbe\xfa\xea\v\xed\x0f\u007f\xb8\x85\xad[\xb7\xca\x06\xe2ml\u0630\xb6\xcd\xe4\u025f\xa4WUU\xb7\xf6\xfb\x03\xc3\xf7\xed+\xca\b\x04|\x03\x0e\x1e<\x94\xb4\u007f\xff\x01\xc2\xe10\xb1X\x94X\xcc\xc4i\xb0\xf0E)U_\x94J\xd34\fC\xc7\xe7\xf3\xe1\xf7\a\b\x85\x82dee\u046cY\xb3\x9ah4\xba8"\xa7\xaci\u04cc-
J\xa9UC\x86\f\xd9;h\u0410\"!DM\xddv\xa4\xa5\xa5j/\xbe\xf8\"\x17\\0\xee\xbf\"\xeaJ\xa9\x8c\x1b\u007f\xfb\xdbY\u04e7\xcf\xe8^\\\\\x8c\x15\x8bb[\x16\x00\xa1F)\x8c\xfd\xd3_\xe9u\xee\x04\xa2\x11\a\xe9\xd8\u0636\x85Y[K,\x12\xc1\xb6,\x84R\b];Z\x90K\xc5\xc3/Rb\x1a\t\x1c\u06b7\x97o'=\xc8\xfe\xb5\x8bi?\xf8L\x86]\u007f7\xc1\xe4\x14b\xd1\x18a\a\xfc>\x03\u007f\xc0\x87\xa3\xfb\xd14\x8dPJ\n\x05\x1f\xbd\xca\xf4\a\xae\a%\xd1t\x03\u007f
{\x9d\xd4C\x1b}\u06a8G\xee\xbc\xeb\x8fwy#\xda\xc3\x13\xf3_\t\xf3\xe6\xcdf\u0630\x91L\x9a\xf4\x12\x13'^W/\xbcJ\xa9\xd0\xec\xd93\xfb\u035a5+\xaf\xa4\xa4\xb4\xb7eY\xa7TTT\xb4\u06f3\xa7\x90\x92\x92\x12,\xcbB)E0\x18\xa4q\xe3\xc6$"\x93\x9c\u0708\x84\x84D\x12\x12B\x04\x02\x01t]G\xd7uw\"\u03f6\x89FcD\"a\xc2\xe1\b\xd5\xd5\xd5TUUQQQA4\x1a\x05\xc0\xe73h\u04a4\t\xf9\xf9y4i\x92V\u0628Q\xa3e\u035b7_\u06f6m\x9b\xe5\x13&\\\xbaR\b\x119\xea\x92\x1f\xc4q\x14\xfd\xfa\xf5d\xe4\xc8\xd3~\xd2}t\xf7\xddw\xf1\xe0\x83\x0f\x030k\xd1\xe2S\x9e{\xfa\x99\x99\u02d7.\xf6\xd7VWc\u01a2\xc8x\xbewjV.\x17<2\x89\x0e\x83\x87\x13\xad\xb5\xb1\xad\x18v$\x82\x15\tc\x99\x16\x8eTG\xfbV4\xa8\xb4(\x95\xa2*\xea@0\x89\x03K\xbed\u065b\x8f#\x1d\x87\xc1W\xdfA\xe7\x91c\x89DbT\xc7,\x82>\x83P\u0407\xd2\xfdh\x86N\xa0q\x13\x0emY\xcb\u053b.\xa3x\xdbw\bM\xc3\x1f\bZ\xad\xf3\xf3|C\x86\f\xfa\xfa\x85\x17^\x1a%\x84\x88\x1e<\xb8Ode\xe5*o\xb4{xb\xfe\v\xa4\xa4\xe4 M\x9bf\xfd\xdd\xe3{\xf7\xee\xea2y\xf2\xa7\x97-
_\xbe\xbc]MM\xcd\xe0\x8a\x8a\x8a\xe4\xad[\xb7Q[[\x8b\xae\x1bdee\u04be};\xf2\xf2\xf2\xc8\xc9\xc9&++\x8b\x8c\x8c\f22\xd2INN&!!\x81P(\x84\xdf\xef\xbaGW\xcc\x15\x8ec\x13\x8b\u0148D\"D\xa3Q\xaa\xaa\xaa)++\xa3\xa4\xa4\x84\xe2\xe2b\x8a\x8a\xf6SX\xb8\x97\xed\u06f7\xb3\u007f\xff~b\xb1\x18\xc1`\x90\xb6m\u06d2\x9c\x9c\\\x15\f\x06\xbe\xee\xd9\xf3\xa4\xcdc\u01ce\xfd\xb8{\xf7\x93\n\x8e\xdf\xee\u077b\xb7\x93\x9f\xdf\xf6'\xd9W\x05\x05\xab\x985\u007f\xa1\xb8\xeb\xd6[\x15\xc0\v\x93\xdez\xf1\x93\x0f?\xbc~\xf5\u0295\u0122\x11e\u0162\xa2.\x0e\u07b2[\x1f\xce}\xe0Ur\xbbv%R\x15\u00caF\xb0\"\x11wE\xa7\xe3\xb8\xcd+\x94jP[\xd1Mo\x89\xda\x12\xcd\xf0#\xec\x18\x8b_\u007f\x94Ms>#\xf7\u013e\x8c\xb8\xe9A\x82\xe99T\xd7F0\f\x9d\u0120\x81\xe1\xf3#u\x1f\xba?\x00\x02f>\xfc\x1b\xd6|\xf6\xa6+\xe6\xfe\x80\u04f8qc}\xc8\xe0\x93\xf7>\xf5\xd4\xe3\x17\xe5\xe6\xe6/\xfd\xfc\xf3O\xc59\xe7\x8c\xf5\xc4\xdc\xe3g\x89\xe1\xed\x82\xefg\u0294\u027c\xf3\xce;\xa2i\xd3,u\xdc\xe3\xe7}\xfe\xf9\u0511\x97]v\xc5\u041a\x9a\xda\xd6;w\ue92a\xaa\x8aF\x8d\x1a\xa9.]:\x8b\x9

e=O\xa2{\xf7\x1e\xe4\xe7\xbb\"\x9e\x91\x91Abb\xf2\xf1\x81\b@\x1d3\xc9W\xf7\xbf\x887np\xd3\xfa\xfe\xbe\xf2w8\xec\x8a\xfb\xc1\x83\a),\xdc\u02da5k((\xf8\x965k\u05a8\x8a\x8a\xaa\u4924\x843\xf7\xef\xdf\u007f\xe6\u0085\x8b.\xbc\xfc\xf2\xcbV\x8c\x181b\xf2\xc5\x17O\xf8\xa4\xee\xf9uB>s\xe6tF\x8d\x1a\xfd\xa3\uecd3N\xeau\xd4J\x037L\xbc\xec\xee\u28bd9\xd5\x15G\xce\u06bcy\xb3@\xf91c&\xa0(\\\xb7\x82\x99O\xdf\xcd\xd8\xfb^\xa2I\xf3\x1cl\xdbF7,\x94\xae\xa1\x94\x83-
\x05R\xe0\x86\\\xe2\xcbF\x95\x02\xbf\xae\xa1\t\a\u007fr\n\x1dO9\x93\x03\x1b\v8\xb4e-
[\xbe\x9eA\x971Wa\xe8\x1aR),[\xa1\xeb\x0eB\xd3Q\x8eE(5\x85\x8cV\x1d\xdd}-%RJ-
\x12\x8b\x12\xae\xad\xcd]\xf4\xf57\ud065;v\xec\x10\r\xb7\xdf\xc3\xc3\x13\xf3\x9f1_}\xf5\x05\x8b\x16-
\x12\xe7\x9e{\xfe\xd1\xc6\xf1J\xa5\xbd\xf9\xe6\xeb\x17O\x9d\xfa\xc5\x19\x0f>\xf8\xf0\xc9ee\xa5\xc1\xe2\xe2\x12\xfc~\xbf\u0763G\x0fN=u\x84\u07bbw/\x91\x9f\u07caf\u035a\x92\x94\u0538^\xb4m\xdb$\x1a\xad=&3\u37c9c\x1f\x15{\xb7\xff\xa5\xdf\xef\xa7E\x8b\x96\xb4h\x91O\x9f>\xfd9\xf3\xcc3(--
e\xf7\xee=b\xe5\xcaUj\xee\u0739\xce\xea\u056b\u067auk\xcb\xed\u06f7\xb7\\\xbd\xba`\xf4\x84\t\x13\xee9\xf5\xd4\x113&L\xb8\xf4q!\xc4\x11\x80Q\xa3Fs\ubb77\x88!C\x06\xab\xd3O?\xf3G\u0747u\xf1\u007f!\xc4\x11\xa5\u0504CE\xfb>\x8aE\"\xa7\xed\u06bd\xcbM54-
@\xb1i\xc1\x17$7\xcba\xf4\x1dO\xe3OL$b\x9a\b\xc3@H\x89\xa6\x1c\xa4\x12G\xd3\x14\x1b4\xa9PRa\x9bQ\xb2;\x9dD\xab\u07a7\xb0\xf6\u02ff\xb1c\xd9\x1c\x9au?\x99\xf4\xbc\xf6\u0122\x11L\xdb\x01M#\xa89
u\x90\x90\x9c\x99K09\x95hU9JI\x11\x8b\xc6,\xdbq|\xc5%\xa5\x1d\x00\xaa\xaa\xaa\xbd+U\x0fO\xcc\u007f\tTU\x95\x8b\xe4\xe4\u0506mox\xf6\xd9g\xae\xba\xf2\xca+\xee[\xb6ly\xf6\u07bd\x85\x84\xc3\x11\xb2\xb3\xb3\xcdq\xe3.\xd0/\xb8\xe0|\xa3{\xf7\x1edd\xa4\xe3\xf3\x05\x01\xb0\xed\x18\xe1p5J)4M\x8b\v\xb7@\xd3\xfe5\x9d8^\xf8m\xdb\u01b2,\xa4\x94\b!\xf0\xf9|\xb4h\x91O\x8b\x16\xf9\f\x18\xd0_\\~\xf9\xa5\xc6\xfa\xf5\x1b\x98:u\xaa3g\xce\\\xb9}\xfb\x8e\xa4\xfd\xfb\xf7w]\xb3fM\xd7\xf9\xf3\x17\\\xfe\xfa\ubbfd|\xe5\x95W\xdd/\x84\x90O<\xf1\xa4z\xe2\x89'9r\xa4T4i\x92\xf1\xa39R!\x04J)\xee{\xf0\x01\x9f\x10\xa2z\u03de\x1d\xf7\xff\xe5/\xcf\f\xa8\xae\xaeJ.--
SJ)a\xdb6J)\x96\u007f\xf82\xfe\xe44\x06\xdf\xf0gT\xa0\x11\xb1\xa8\x89\xc2F)\a\x15oN\xd1\xf0\xe2\xa4.\xd5Q:\x0e\xbeP\x02\xed\x06\x8f\xa6p\xcd\x12\xcavof\xd7\xf2\xb9\xa4\u4dad\x0f\xc98\x8e\xc4r\x14>]\"\x1dE\xa3\xf4L\x12\x1a7\x89\x8b\xb9['\xecHy9\u06f7os\xdc\x10\xd4n\xefK\xe0\xe1\x89\xf9\u03dd\x993\xbf\u0493\x93S\x9d\xb8\xb3\f=\xf3\xcc3#\x97.]r\xe3k\xaf\xbd>\xa2\xa8\xa8\x88H$B\u01ce\x1d\x195j\xa4\x1a3f\x8c\xbf[\xb7\xae\x04\x02\t\xf1g;XV\xf4hIV\xfd\xa7\xab\x92P\x17\x86q\x1b&\xbb\xab\x1fc\xb10B\xb8\xae=##\x93\xa1C3\x192d\xb0\xbe{\xf7n\xfd\xf3\u03e72m\xda\x17j\xed\xdau\xe2\xc0\x81\x03\xcd\n\n\n\xee\x9d5kV\xffg\x9e\xf9\xcb\xf37\xdd\xf4\xfb9B\x88H\x93&\x19j\xe6\xcc\xe9\xfa\xa8Q\xa3\x9d\x1fs;\xcb\u028a\xed?\xdf\xf3'\xf2\xf2\xda,_\xb4h\xc1\x05EE\a\xfe\xb6d\u0252\x8cp8\xecD\xa3\x11\u0772\x1dP\x8aE\xaf>\x04\xbaA\x9f\x89w
\x83\xc9D\xa2&8\x02\xa4[tK\b04\x81\xae\xc5\xc3N\xc2]d\xe4\xd8\x0e\xe9m:\u04e2\xd7\x10\x8e|6\x89}\xab\x16\xd0\xf2\xa4!4m\xd3\t\u06cc`\xa1\x10\x8e\u0110\x12\u01f2\t5iJ0\u0794Z\u0157\xf7G\xa31\x8e\x1c)\a\xa0\xbc\xbc\x9c\x993\xa72j\xd4\xd9\xde\x17\xc2\xc3\x13\xf3\x9f\x1b;vlc\xc0\x80\x93\u0168Q\xa7;\x00K\x96|3\xec\xe6\x9b\u007f\u007f\xc7\xe2\xc5K\x86o\u0672\x95H$\xc2\t'td\xec\u0631j\u0738\vD\xbbv\x1d\xc5\xd1\x10J\f\xc7q\xfe'>\x87R\x8aX,V\u007f21\f\x1f\xad[\xb7\xe3\xe6\x9bo\xe5\xe2\x8b\u01cb\xa9S\xa71y\xf2dV\xacX\xc9\xee\xdd{F\x14\x16\xee\x1dQX\xb8w\xee\u0085\xf3\x1e\x1d2d\u063cQ\xa3F;-Z\u42af\xbf^\xa8\xf2\xf3[\xff(\u06d4\x9e\xdeL-

[\xb6X\xf4\xeb7P\r\x1at\u02ac7\xdex\xedV\u04cc\xbd\xb5r\xe5J\xddql\th\x96e\x03\xb0\xe8\xa5\xfb@)\xfa_u\x17\xaaq\x1a\x91#%8\xb6\u0114\uef02O\x13\x04\r\x1d]S\xf5\u92a6i\xa2\x05\x13\xc9:i\b;\x96\u03a6\xa2p\x1b%\x9bV\u04fcCg\x94\xa90\xa5\xc4q$\xd2qP\xd2\u0097\x90T\xbfx(\xfe\x12H\xa5\x88F#\x0e@ff3O\xc8=~\xb6\xfc*c\x84EE{\x99:u\x1a7\xde\xf8\u007f\r\xc50\xf9\xa6\x9b~{\xe5\xb6m\xdb\xfe\xbca\u00c6\u48a2\xfd\xe4\xe66g\xfc\xf8\xf1L\x980\x9e\u039d\xbb\u015d\xb0uL~\xf8\xff\xf4\xc1\x15\"\x9e\xf6\xe8\a\xe0\xc0\x81}|\xfc\xf1d\xde~\xfb\x1d\u05ad\xfb\x8e\xec\xec,\xbat\xe9R\u0561C\x87??\xfd\xf43o\b!\xaa\xea\x9e\xfb\xf6\u06d38\xe5\x94Sh\xd1\"\xff\xdf\u078e\xab\xae\xba\x92\xd7_\u007f\x03\x80G\x1ey\u8669S\xa7\u0774e\xcb\x16,\xcbR\xa6i\t+\x9e\x83\x0e\xd0\u007f\xe2\xad\xf4\x9dx;\x18A\xc2e\xc5\u0636\x8d\xe9Hl\x05\xba&\xf0k\xe0\xd3\xdc@\x8a\xa3\x14\xd2\x1f\xa2\xb6\xb2\x825o?\xce\ue15f\x93?`\x14C\xae\xb9\x93\xa4\xd4tjL\x1bt?\xc1\x80ABR\x12\xb6\x19e\xca\xed\x17\xb3s\xe9\x1c4]\xc7\b\x84\x9c\u071cl}\xf4\xa8\x11\x9f>\xf7\xdc\xf3\x13\x84\x10\x91\xc2\xc2]\xa2e\xcbV\xde$\xa8\xc7\u03ce_\xe5r\xfeo\xbeY|\x8c\x90/^\xbc\xa8\xef\x84\t\x17\u007f5s\xe6\u033f\u031a5;\xf9\u0421b\u018c9\x87\xd7^{\x95G\x1f}\x84\u039d\xbba\xdb1b\xb1p}\xbc\xfa\xe7\x80R\n\u06f6\xe3\xdb\x1d%;;\x97\xdf\xfd\xee\x0f\xbc\xf1\xc6k\xdcp\xc3\xf5\xc4b&s\xe6\xccM\x9e1c\xc6_.\xbf\xfc\xb2\xaf\u05ac)\xe8[\xf7\xdc\xcb.\xbb\x82\xa5K\x97\xfd(\xdb1n\xdc\xd16sw\xde\xf9\xc7\xdf\xf5\xee\xddkR\u02d6y\x18\x86!\x02\x01\xbf\xf2\xfb\xfd\xf5s\x03K\xdf|\x82yO\xddL\xf8\xf0A\x92\u049a\x12\xf0\x1b\x18\x9a\xc0\xd0\x04R*\xa2\x0e\x84-
E\xad%\x89\xd8\xeed\xaa/)\x85f\x9d\xfbb\x84\x12(\u0771\x9e\x8a};\t\x04\x02\b\xe9\xa0\xe2\xee\u0736\x1d@\xa05\x98\x83\xd04!\"\x91(MR\x9bt\x06\xda\x03\u0636\ud578\xf0\xf0\xc4\xfc\xe7\xc0\xa2E\v\xc5E\x17\x8d\xaf\xbf\xff\xf8\xe3\x8f]\u007f\xd7]w\u007f2k\u05ac\x81\u06f6m'??O>\xf2\xc8\u00fc\xfc\xf2\x8b\x9cz\xeai8\x8eM$R\xf3?\x13N\xf9W\x91R\x12\x8d\xd6b\x9a\x11z\xf4\xe8\xc5\x13O<\u018b/\xbe@\x9f>}\xe4\xb6m\u06d81c\xc6\xc0[n\xb9e\xf2s\xcf=\xf3\x1b\xa5T\xc0\x15\xe1\xf1\u03181\xfd\xdf~\xef\xe1\xc3Oe\u0294\xc9\xf5\xf7\x9f}\xf6\xafw\x9e}\xf6Y\x85\xb9\xb9-
\x10B\b\xbf\u07e7|>_\xfd<\xc0\xba\xa9o3\xeb\xa1\xeb)\u07bc\x8a`\xe3&\x84BA\fA\xbc@W}).\xb7\xe2\xa2\x02#\x10\"\xa5U'\x92\x9a\xe5R{\xb8\x98\x8a};\xb0L\xab\xfe\xaf\xed\xf8D\xaa\xb2-
l\xdb>\xbaO\x14*\x10\nRU]\xb5\x1d\xd8\x1d\x0fQy\xae\xdc\xe3g\u026f*f>o\xdelm\u0420!2\xeeZ\x1b\xdd~\xfb\xad/\u007f\xf4\xd1\xc7\xe3\v\n\n\x00\x9cQ\xa3N\x15w\xdcq\x9b6x\xf0\x10\xc0\xcd\xe9\xd64\xad^d~\xf615!\x90R\x12\x0eW\x13\f\x069\xff\xfc\v\xe8\u05ad\x8b\xf6\xdcs\xcf\u02f7\xdf~G\u035b\xb7\xa0yuu\xcds\a\x0e\x1c\x1c\xa0\x94\xbaB\b\x11>\xed\xb4\xd1,]\xbaX\xeb\xdf\u007f\xe0\xbfu9r\xee\xb9\xe7S^^Fjj:B\x88\x92}\xfb\xf6^\\YY\xf5\xf1\xb4i\u04f2\x8b\x8b\x0f\t\xbf\u07e7\x84\x10\u00b2m\xa4\xe3\xb0{\xd6vD[\x00\x00 \x00IDAT\xf9\\j\x8a\x8b\xe8?\xf16\xf2\xfa\x8e
!1\x89\xda\xdaZ\x1c\xe9\x80\x10\bM\u00f6lJ6\xad\xc0\xb1LR\xf3;\x92\u07be\a\x15\x85\xdb(\u07f7\x93hM9\x9a?\x19G)lGaK\x90\x910\x8e\x19\xab\xdf\x17\x80\n\x85B\x1c8pp\xa3\x10\xa223;K\xcb\xcd\u0353\x9e,xx\xce\xfc\u007f\x98\xd7_\u007fM\x1b6l\xa4\x04X\xb1bY\x9f\xf1\xe3/\x9a\xf5\u99df\x8d/((11Q\xddv\xdb-
\xda\xdf\xfe\xf6\xb66x\xf0P\x00\xa2\xd1\b\xba\xae\xf3\xdf,X\xf5S\t\xba\xae\ub626\x89\u3634k\u05d1\xe7\x9e{Z{\xfa\u99f4\xbc\xbc\x96j\xe5\xcaUL\x992\xe5\xc2\xf3\xce;\xf7\xf3\x82\x82\xd5]\x00\xfa\xf7\x1f(?\xf8\xe0\xbd\u007f{\xac\xa4\xa6\xa63c\xc6W\x1a@nn\x8b%c\u01desf\x8f\x1e=\xf64i\x92\x86\x10B\x05\x02~\xfc>_}6P\xe9\xee-
\xcc|\xf4&\x96\xbf\xf98\xe1\xb2\xfd$%7\xc6\xef\xf3#\xa4D\xd7t\x90\x0e\aV\xcfc\xcd[\x0f\xb3o\xe9t\x1ae\xb6@\xf7\xfb).\xdcAuy9\x9a\u03c7\x94\n)\x1d,\x05U\xe5e\xc4j\xea\x9aN\v\xea\u02a6\u06cek\xd7'\x8c\x1f\x1f\xb8\xf3\xce;|g\x9ey\x86~\xe3\x8d\xd7\ubbfe\xfa\xb2>g\xce,}\u04e6\rZ\xc3u\x0

2\x1e\x1e\x9e3\xff/\xf1\xc8#\x0f\x8b\xab\xae\xbaZ\x02|\xfc\xf1G\xe7>\xf1\u0113o-
X\xb00\xe9\xf0\xe1\xc34o\x9e\xa3\xee\xb8\xe3vq\xe3\x8d7\x00:\xb6\x1dsK\xa8\x8a_\xf6\u0730\x
10n\xd7\x1f\xc7q\xf0\xfbC\\u\xd55\"77\x97\xbb\xef\xbeG\xae^]\xa0UTT\x8e\b\x85\x9e^2}\xfa\x97\x
13G\x8f>c\xcaE\x17],\x9f}\xf6\x19q\xd3M\xbf\xfb\xb7T\xed\xb4\xd3N\x97\xef\xbf\xff\x9e\x18?\xfeb
5x\xf0)\u07fe\xfa\xea+cm\u06da\xf2\xed\xb7k\xf2\xcb\u02cfH\xc3\u0405\x10BX\x96\x1b\x12\x89\x
85\xabY\xf5\xc1\xf3\x14o]G\xb7\xb1\u05d0\u0575?\xbe\xe4T\xccH-
\t\t\x89d\xb4\xeeD\xe1\u0499\xec\x98\xfd1M\xf2;\xe0\v%RUz\x90\x8a\xcaJ\x82\xb9\x06J\xc4\xdc
u\xb4B\x10\xa9\xac \x16\xae\x89\u007f~\x90\xd2\xc1\xef\xf7\xd3(\xa9Q-
\xc0\x93O>\x15\xf9G\xdb=p\xe0@\xfdO\u007f\xba\x1b\u01d1j\xf8\xf0a\xea\xe4\x93\a(\xc3\b\xb2g\
xcfN\xf2\xf2Z{J\xe2\xf1\xdf\xffN\xff\x92?\xdc\xef\u007f\u007f\x13\x8e\xe3\x88\xe7\x9e{^\x01<\xff\x
fc_\xc7N\x9b6\xed\u0765K\x97\x85jjjh\u06f6\xadz\xfc\xf1G\xc59\xe7\x8c\x05\xc04#\xfcZ\x1dX
\x10\x044\u05ae\xfd\x96;\xee\xb8S\u035a5[\xa4\xa44f\u0420A\x911c\xc6L\x988\xf1\x8aO\x01n\x
bd\xf5\x16\xe1\xf7\xfb\xd5C\x0f=\xfcO\xbf\u01de=;\xf9\xeb__\u2a67\x9e\xe4\uaaef\xe6\xb5\xd7^\x
03\xe0\xbd\xf7\xde\xed>u\xea\xd4OV\xacX\u046a\xa4\xa4\x04!4%\xa5\x8c/,r\x87\xa9t,\x92\x9ad\
xd2\xfe\x941\xb4\x1dq.\xa9y\x1d\x11\x02\xecX\x94\xed\xf3\xa6\xb0i\xc6\xfbD*\x8f\xa0l\x1b4A\xff\
xdf<F\x8b>\u00d0\xb1(J@Bj\x06{\x17\u007f\xc1\xdc\xc7~Cmy\x19\x86\u03cf\x12\x9a\u04f6Mk}\x
e4\xf0\xa1\u04c7\x0f\x1b\xf6F\xd1\xfe\xfdM5M\v\xef\u077bw\xe5\xc9'\x0f\x8c\x0e\x1f>\xd40\x8c\x
a0\x01\xec\xfd\x9e\x92\xc3\xc7T\xba\\\xb6l\t\xfd\xfa\r\xf0\x14\xc5\xc3\x13\xf3\x9f\x82\x1bn\xb8^\x
bc\xf8\xe2K\n\xe0\xc5\x17\xffz\xfaG\x1fM\x9e\xbcr\xe5\xaaP$\x12\xa1[\xb7\xae<\xf9\xe4\x13\f\x1f
>\x12)M\xea\xf2\x9d\u007f\u0543A\b\xfc\xfe\x10\x85\x85\xbb\xb8\xf3\xce?\xf2\xe1\x87\x1f\x92\x9
0\x90\xc0\xc0\x81\x03#\xe3\u018d;\u007f\xe2\xc4+\xbe\x02\xb8\xed\xb6[\xc5\xe3\x8f?\xf1\x0f\xcf
z\xdf~\xbb\x9a\x1e=z\xfeS\xef]SS\xde\xea\xce;\xef\x992s\xe6\xac\x13\x8b\x8a\x8a\\\xb1\x94\x12\
xa9\x14\b\x1d!t\x94t\x90\xb6EZ\xab\x8et\x18u\x11m\x06\x9fIJ\xb3\xe68\xb10;W-
b\xeb\xbcO)\u07b8\x12'\x16\xa5\xf7\xb5\xf7\x91?\xe8\f\x84\xb4\xd1\x04$\xa5g\xb1\xe1\xb3W\x99
\xf7\xd4\x1fp,\v\xe16\x0e\xc5\xd05\u0661C\a-
';~\xa5\"),,\\\u05e8QR$33\u04d7\x96\x96\xee\x03\xb9\xc3\xe7\xf3\xefh\u07fe}E\xf3\xe6\xcd\x17]r\
u0265\x87\x85\x10[\xbe\xefs\xfc\xf1\x8fw1x\xf0
F\x8e\x1c\u5a4b\x87'\xe6?\x06o\xbe9IL\x9cxE\x9d\x90\x8f\xfe\xf0\u00cf\xdf[\xb5jUJ$\x12\xa5W\x
af\x9e<\xfb\xec\xd3\xf4\xeb7\x10\u02ca\xe28\xce/>\xac\xf2Cq\xcb\xf5&RZZ\xcc\x1f\xffx7\x93&M\x
c2\xef\xf73`\xc0\x80\x8aK/\xbd\xf4\xe2K/\xbdI:\xc0G\x1f}
.\xbc\xf0\xa2c\x04\xfd\xb3\u03e60f\u0339?\xe4=\xda\xee\u07bd\xa3\xe3\xea\xd5\x05r\xf7\xee\u07
72\xb2\xb2Jj\x9aV2l\xd8\xd0\xd3\u05ee]w\xff\xfb\xef\xbf\u03d6-
[\xb0m\xc7\xcd\"\x12\x1aB\xe8\b\xcd\x00\x01v4\x8c\x11\b\u0462\xe7`N\x18y.-
N:\x05\xbdq\x06\xbbW-
`\xc5\xcb\u007f\xa6r\xdf\x0ez]s/\xad\x86\x8dE\x03\f\rR\xd23Y\xfc\u04bd,\x99\xf4\x88\x1bc\xf9\xfe
+0\xe5\x9e\xd3D})\x06M\xd3HJJ\"++\x93\x94\x94\x14,\xcb\u079b\x99\x99Y\x99\x9f\x9fw
77w\xe6%\x97L\xf8\xb2Y\xb3\xac\x1d\xff\xaf\xfd\xe9\x8d-
\x0fO\xcc\xffEV\xae\\&z\xf7\xee\xa7\\\xd1\xf9p\xf4\xcb/\xbf\xf2\xde\xe2\u014bS,\u02e2w\xef\u07bc
\xf0\xc2s\xf4\xec\xd9\a\u04cc\xd4\xd78\xf18\x8a\x94\x92P(\x89\x8a\x8a\xc3\xdcq\u01dd\xbc\xfa\x
eak\xe8\xba\xce\u0211\xa7V\xfc\xfe\xf7\xbf\xbbx\u0108\x91\xd3\x01V\xaf^!z\xf6\xec\xf3\x0f\x1d\xf
a\xaaU+\xfa}\xf6\xd9\xe7\xadJKK\xba$$$\x0e\u07bd{O\xb8\xac\xacL$$$\xe4\x80\u02a8\xaa\xaaR
\xb1XL\u0555\xfe5\f\xa3\xa6}\xfb\xf6\xd9EEE\xbe-

[\xb6P[S\x8b\xed\xb8\x95\x0fu\u0747\xa6\x1b\bMC\xd3t\x1c\xdb\u010eFH\xca\xc8&\xb7\xfb@:\x9c~\t\xca\b\xb0\xf4\x85\xbb(\u07fd\x99>\xd7\xddG\x9ba\xe7!P\x18~\x1f2Z\xcb\xfc\xc7\u007f\xcf\xf6E\xd3\x10\x9a\x86j\xb0V\xa0\xaeDB\x03\x01\x96\xc7=\xae\x1c\u01d1\x80\x1e\n\x055\x9f\xcfOVV&\t\t\x894j\x94\xb4\xbbC\x87\x0e\xbb\xf3\xf2\xf2>\x1f>|\xd8w\xbd{\xf7\xfd\xfa\xfb\xf6\xc5\xff\xfd\xdf\r\\{\xed5t\xe9r\xa27\xc0<<1\xff!\xbc\xf3\xce[\xe2\xd2K/\aP\xdf~[0\xfc\xfe\xfb\xef\x9f2}\xfa\x8cd\xd34\x9d\xee\u077b\ubbfc\xf2\"\xbdz\xf5\xf5\x84\xfc\a\nzyy\x197\xdf|+o\xbe\xf9\x96\x13\b\x04\xf41c\xc6T\xdd{\xef=c:v\xec4?\x18\fj\xd1h\xb4auI\u07c2\x05\xf3:~\xf8\xe1G\x83\x0e\x1e<x\xd9\xee\xdd{:\x16\x17\x17k\x9a&B\xc1`\x90h4\x86e\xd9\xc4bQ\x1c\xc7F\b\xed8!u\x17\x01%&&*P\"\x1c\x8e\xb8\xcd=\x00\xc3\xf0c\x18>4]\a\xb4x\x98\x04\xacH\x18P\xa4\xb4hKbz6\xe5{\xb7a\x9b\x11\x86\xde\xfa,\xad\x87\x9c\x8dm\x9a\x04\x92\x1aQ\xba\xa5\x80\x19\u007f\xbe\x92\x92m\xeb\xfeN\u033f/\xdcTGC\x87\x1e/
\xe6(\xa5\xb0,\xcb\xd14\xcd\u07e8Q\x12\xa9\xa9\xa9\x04\x83A'9\xb9QU\xabV\xad\xcb:v\uce22}\xfb\xb6S\x87\r\x1b\xb6*##\xb3\xf0\xf8\xd7_\xb6l\x89\xe8\xd7o\x80\x97\x1e\xe3\xe1\x89\xf9\x0f\xb9\xa4-
**l\xf5\xdb\xdf\xfe~\xfa\xfc\xf9\xf3\xdbWTT\u0431c\a^\u007f\xfdU\xfa\xf7?\x19\xa5\xdc\x06\x10\x9e\x90\xff\xff\f\x8ex\f\xfd\u02112\xae\xbd\xf6:>\xf9d\niiM\x18:t\xe8\xc6\x17^x\xee\xf4\xa6M\xb3\v\xe3\xfb=\xf5\x81\a\xee;s\u04e6\u037f\u0674is\xcf\x1d;v \x84
\x1c\x0e7\x9cP\xae\xaf\u007fX\x97\x1e\xf9}\xf9\xfbu\xc7\xd00\x8c\xfa\x15\xac\xb6m\xbb\xee\xdc\xf0\xa3\xeb\x06\x9a\xa6\xe3\u0583\xaf;\xf1\xd8H\xcbD\xd3\f4\x9f\x8f\xa4\xa6\xd9\xf4\xbb\xf2.\xd2\xdbt&RSE
\x94\xc8\xc1\x8d\xabX\xfa\xda\x03T\x97\x1e\xa8\xaf\xeaX\xf7\x15h8\f\xea\xab4\xba\xa7\xa7\xef\x1d#\rJ\xfc\"\xa5T\x80\xd0u\xb7\xcd_rr2\x81@\x80\xb4\xb4&t\xea\xd4\xc9NII\xf9\xa8m\u06f6_\xe6\xe7\xb7Z}\xd6Yg\x954,\x99\u0423G\x0f\xe3\xba\xeb\xaea\xe4\xc8\x11N^^kO\xdc=<1\xff\x9e/[\xfa\x84\t\xe3\u07de7o\xfe\xe8C\x87\x8a\xc9\xc9\xc9\xe1\xf9\xe7\x9f\xc3\xcdZq\x88F\xa3\x9e\x90\xff@4M\xc3\xe7\v\xb2c\xc7V\xae\xb9\xe6:\x16,XHNN6#G\x8e\xfcj\u04a47\xcf\x02|\xbf\xfb\xddM\x9f~\xf3\u0362\xd1\x1b7n\"\x163\x8f9\x14\xc4c\xd0J\xb9\xa1\xea\xba\xfd\xfe}9\xfc\r\ufeee\xdd}\t\xdbvk\xe1h\xba\x0fM3\x10\x9a~\x8c\x007\x8c\x81\vM#\u0538\t\x89M\x9ab[&V,\x82t\x1cb5\x95\x84+\u0290\x8e\xddP\xad\u007f2|>\x1f\xc1`\x90`0H\xeb\u05ad\xc9\xc9\xc9.MKK[\x92\x96\x96V\x90\x92\xd2x\xfem\xb7\xddQ$\x84\xd8[\xf7\xf7o\xbd\xf5\x866r\xe4\xa92;\xbb9\xf3\xe7\xcfe\xe8\xd0\xe1\xde\xe0\xf3\xf8\xf5\x89\xf9\x8c\x19_\xb1p\xe1\xd7\xe2\xb1\xc7\x1eW\x00\xf7\xdcs\xf7\vS\xa6|z\u00e6M\x9bHHHP\x8f?\xfe\x98pk\xb1(\xa2\u0470'\xe4\xff\xe4\u054ea\x18\x18F\x80\xe5\u02d7r\xf5\xd5\u05e8\r\x1b6\x8aN\x9dN\xe0\x8a+&>\x9e\x94\x94\xd4\xf4\xd9g\x9f\xbb|\xf3\xe6-
\xff0\xad\xb3n\u007f7lX}\xb4\xd6\xfb?h\u05a1\x14R)\xb7T\xadRnHL\xd3\u30ae\xd7\u05c9o(\xca\xf1P7B)\xecX\fG\xd9?H\xb2u@\xd3\x05\xb6\xa3~\x12\x897\f\x03\xbf\xdfG(\x14\"33\x93\x9c\x9c\x1c\u0574i\xd3\xe5\x19\x19\xe9S\u018e\x1d\xb3\xec\u44c7,=\xfe93g~\u0168Q\xa7{\x03\xd0\xe3\xd7%\xe67\xdex\x03/\xbc\xf0\"\x00\xaf\xbe\xfa\u0288\x17_|q\u06bau\xdf\x05\x95R\xfc\xe1\x0f\xbf\xe7\xd1G\x1f\xc6\xe7\xf3y\x8e\xfc\xdf\x10tM\xd3\xf0\xfbC|\xf8\xe1\xfb\xfc\xf6\xb77QZZF\x9f>\xbdcyy-
\x03\v\x16,\xa4\xa4\xa4L\x89x\xe7\x88\xe3E\xfd\x18\xc7]\xe7\xa4\x1b\x8a|\x03\x1b\x0f\n%]!\xff\u01c36^b\xa1>\xde\xee\xde\xfc\t\x894JoF()\x19_(\x01_
\x88\xe1\xf3\xe1\xf3\x19\x84\xfc\x06\x86\xc0\x9dh\xd5\x04\x89\x8d\x92\xc9h\u0782\xe4\x8cl\x84\xe1\aM#\x1251-

\x13G\xbaW\x03V\xa4\x96HM5\xb5U\x15T\x94\x1e\xa2d\xefN\x8e\xec\xdbE\xac\xb6\xa6~\xfb\xea?O\x83\xab\x83\xff\xd7IM\b\xd04\x9dF\x8d\x1a\u047auk\xd2\xd3\u04cb\xf3\xf3\xf3\xbe>\xf7\u0731\x8b\x86\r\x1b\xf1\x8a\x10\xa2>G\xf6\x92K&0b\xc4p\xe2s@\x1e\x1e\xbfl1\u007f\xe2\x89\u01f8\xf5\xd6\xdb\xeb\xbeDY\xa7\x9f~\xfa\xeco\xbe\xf9\xa6suu5g\x9cq:o\xbc\xf1\x1aM\x9bf\x11\x0eW\xff\xa4M#~\xe9H\xa9\xf0\xfb}\b!\xb8\xff\xfe\ax\xf8\xe1G\xd1u\x9d\xe4\xe4d\x94\x92\xaa\xb2\xb2\xaa\xbe{\xd0O\xb1\xf0J\xd7\r\x8c@\x00_
\xc1m\xd4l\u06d8\xd1\bB\bB\u0269d\xb4\xea@\xeb\x01\xa7\u04b4M\x17t\xbf\x1f_
\x80?\x10\xc4\xef\xf7\x91\x14\xf4\x91\x92\xe0\xc3\xf0\xb9\xd5\x195\x01\x9a/@(9\x15\x15\x00G\x82\xe5\x80%\x8f\x9eT\x1c\a\xa4\x05\x96\x19\xc32\xa3\x985U\u0516\x1f\xa6\xa4p'\xfb\xb7\xac\xa1\xe8\xbb\x15\x14\xef\xdcL\xb4\xaa\x1c+v\xb41\x89\xe6\xaau\xbc\u035d\x02)\xff\xce\xed\xc7E]*\x05~\xbfO\xcb\xcf\xcf'//\x8f\x13N8a\u0168Q#\xdf\x1e1b\xd4K\rO\x80\u007f\xfd\xeb\xb3\xe27\xbf\xb9\u024b\xa9{\xfc\xb2\xc5|\xc1\x82\xb9\xe2\x94S\x86+\x80K/\xbd\xe4\x91y\xf3\xe6\u0771\u007f\xff\x01\u06b5k\xab\xde~\xfbM\u0477\xef\x00\"\x91\x9a_L\xb1\xac\xff\xae\xa0KB\xa1D\xca\xcaJ\xb9\xfa\xeak\xf8\xfc\xf3\xa9\xca\xef\xf7\x8b\xc4\xc4$\x1c\xc7&\x1c\x0e#\xe3\xcd$\xfe\x91\xa0\a\x02A\xfc\xc1 >\u007f\x00M\xd7\x11\x9a\xee\xe6\x91\xeb\x1a\x9an\xe0\v\x04\t&5&)-
\x83\x94f9$7\xcb!9\xad\x19IM\xd2\t5N\xc5\x1f\b\xa1\xa4\xc24c\u0516\x97Qqp?\xc2\xf0\x91\x9a\u05d1\x94\x96\xed\xd0\x02!l\xc7F\x17`h\x1a\xba\xa6\xe1\xd3\x05~C\x03MG\x02\xb6\x84\x98\xa3\x88\xc6Lbu\xd50\x95@\x89\xba/\x85\xdb\xcdH\b\xf7\n\xc0gh\x04|\x06\x01\xbf\x0f\xa5
RSM\xb8\xe20\x87\xf7\xed\xe2\xc0\x96\xef\u063f~\x05%\xdb\xd6Q[v\x88XMe}\xa6\x8c\xd0\u2379\x1b\x84\x8b\x8e\x17uM\x13\xd2q\xa4\r\xf8[\xb5jEnns:w\xee\xfc\xdd\xc9'\x0f|l\u0738\xf1\xef\x1f\xbf\xff\x16/^$\x06\x0e\x1c\xe4\t\xbb\xc7/G\xcc\xdf\u007f\xff]\u018f\x9fP\u007f\xff\x85\x17\xfez\xfeK/\xbd\xfc\u0386\r\x1b\x83III\xea\xe9\xa7\xff\"\xae\xba\xea\xea\xff\xa9n@\xbf\x14\x02\x81\x04V\xadZ\xc1\xd5W_\u00fau\xdf\u0468Q#\xfc~?\xd55n\xa9`\xa5\x14\xb2\xc1>Oi\x9aI\u04fc\xb6\xb4\xec\u0503\xdc\xf6]H\xcbjNb\xe3\x14\x94\xe1\a\xcd@\t\x1d\xdd\xef\xc7\x1f\bb\xf8\xfdh\x86\x0f\xdd\xe7G\xf3\xf9\x11\x86\x0f\xc3\aq\xcdwO*\x8e\xeb\x9c\x1d\x1b\xccH\x14\u04f2\x89Y\n\xd32\xe3B.\xd04\x1d]\xd7PB\xc3A\xb8\xff+0\xa5\xdbrNA}\x8c*7n.\x04\b\x14*^[W\x13\x02\x05\xe8\xc2-
\xb7+P\u8e86\xe6\xf7\xe3\xf3\xfb\xdd\x05L\xa6\x83\x19\rS]\xba\x9f\xd2-
k8\xb0v\te\xdb\xd7S}h\x1f\xe1#\x87\xb0\xa2\x11@C7\f\x84\u0410\xd2q;\x1f\xa9\xefM\x8dTB\b\x91\x99\x99I\u02d6-
\xe8\u0673\xe7\xa6\xe6\xcds\x1e:\u55213{\xf7\xeeS)\x84\xa8\u06e9\xdaSO=a\xb4i\xd3\xda:\xfb\xec\xb1\xea\xd5W_\xe6\x9ak\xae\xf3\x06\xa6'\xe6?_\xbe\xf9f\xa18\xf9\xe4!J)\x95w\xe1\x85\x17\u03181cF\x87\xea\xea\x1a\xfb\xf2\xcb/7^\u007f\xfd\x15t\xddO4Z\xeb\xc5\xc9\u007fd\xdcf\x12\x06\x0f?\xfc\x10\xf7\xdcs\x0fR*\x02\x81\x00\x8e\x94\xd8\xf1\xaeA\t\x8dS\xc8l\u0749\xdcN\xdd\xe9|\xf2\bZu\xefO\xa8qj}\u03b6\x03H\xe2m>\x05\xf8}\xf5\xd1\t7\xf4,\xe3\xa1\x0e\x89[\xf6\xf68\xc7_\xefr\x85\xc0\xb4$\xb5\x11\x13+\xbe\x92\xd7\xd0\x056\x1a\xb6\x128\b\x1c\xa9\xb0\xe2o(\x94B9\xa0I\xe5\x164w\x14\x9a\x12h(\xa4\x00\xa1@\xb8\xadG]a\x8f\x8b<B 5PB
5\x81\xd0\u0709\\\xddg\xa0\xf9\xfch\xba\x86\x9b.i\x11=R\xcc\xe1\xed\xeb8\xb8~)\xa5;6PV\xb8\x8d\xaa\xe2\xfd\x98\xe1Zw\xf27\x9ev\xa9\xbe\xbfc\x95\x02\x84\xcf\xe7#%%\x85\x9c\x9c\x1c\u06b6m\xbd\xa7U\xabV\u04db4I\x9bu\xdbm\xb7\u007f#\x84(\xaf\xfb\xe3'\x9f|\xdc8\xf3\u0333\x9c\xf6\xed;xn\xdd\x13\xf3\x9f'\xf1_\x01\xa8\x9bn\xfa\xcd]\u04e7\xcf|h\xfb\xf6\xed\xb4i\xd3FM\x99\xf2\xb1\xe8\u06b5\xfb\xaf\xbap\xd6O\x89a\x18\u8e9fg\x9f}\x9a\x9bo\xbe\xb5\xfe\xca'\xb1q\n9m:\u04b4mgZv\xebO\xeb>CH\xcf\xcd\xc3\x10`9\xee*O\xe5H\xa4\x94X\xf1IN\r\xf0\xf94|\xba\x1e\x9f?\xfc\xfb\x

8c\x12\xe9\x06\xa0\xe3\xf3\xa6\x02\xd5`\xf0*\x14Q[Q\x13s\x889\nG\xba\xae\xdav\xc0\xb6\x95+\xe0\x8eB\xd9\n\xddQ\xe8\x12\x84T\x88:{^\x97O\x1e\x0f\xaf(\x11?\x9b\xc4\xdfO\xe1\u693b\xb1pp\u217f\x10\xf1\t\xd0\xf8k\b@\xf3\x19\x88\x80\x0f\xcdo`\xf8u|\x86\u00ac,\xa5l\xc7:\x0em_\xc7\xc1\xed\xeb8\xb0m\x1de\xfbw\xe2\x9814\xddp_K\xc9\xf8\xa4\xef\u07fbu\x9f\xcf
\x18\f\x91\x92\x92B\xeb\u05adHKK\x9bw\xc2\t\x1d\x17\x0f\x1atr\xc1\x88\x11\xa3\xbe\xa8\xfb;]\x0f\xb0e\xcbwL\x9d:\x8d[n\xb9\xd5\x1b\xa4\xbf\xc6\xef\xe5\xcfu\xc3\x1fy\xe4a\x03\xb0W\xae\\\xde\xe6\x96[n\x9d\xb8c\xc7\x0e\x00\xe7\xd2K/\u047bv\xed\x0e8\x9e\x90\xffT\x0e@\b\xa4t\x1dx\xe3\xe4d\x8e\x94\xbbF1-\xbb\x05g\xdd\xfc0-
\xfb\f!f\x81m;\x84\xc3Q4%\xd1\\I\u018a\xbb\xdeX<n\x1d4tt\x04\x8e\xaa\x13T\xf7\u007f\xd9@\xb4\x1b\xaa\xbbj`?\xa4\x02\u06c2\u06b0$\x12\x95X\x96D:\x80\xe3\n\xb8\xee(\x84\xac\x13\xdc\xf8\xeb\u01c5\x17\x15\u007f\\\xa8\xb8h\xc7?[\x831#d\xfc\xf7u\x8f;\"\xfe{\xf79\xc4O\f\xc4\xdfCDM\xa8\x8e!5\rS\xd7P~\r#\x90L\x8b\x8e\xc3h\xd5e\x04fu\x19\x87vm\xa0hK\x01\xbb\xd6~\xcd\xee\xf5\x8b\x89\xd6V\xb9i\x9a\x9a\x86\xae\xdcYQ\x85\xac\xdf
\u02f2\xb1\xacj\xc2\xe10\xc5\u0147\b\x85B\xc3v\xed\xda5l\u02d6-
\\r\u0244\x0f\x87\x0e\x1d:\xf3\xf2\xcb'~.\x84\xa8l\u06f6=\x00w\xdey;'\x9ex\"\x17^x\x917X\u007fE\xfc,\xd3;\xde}\xf7\x1dn\xbc\xf17\xd2u.\xbe\xdbW\xacXqfee\x15\xbd{\xf7\xd2\xee\xbf\xff\u03e4\xa66\xc1\x8cw\x94\xf1\xf8i\xae\x8a4M#';\x8b\xe6y\xf9\x1c\xae\x0e\xb3\xbfp7\x8em\u046a\xcfP\xd2\xdbt\xa2\xa2\xb2\n\u02f6q\x1c\x85&\x14\xfex\b%\xe2(\"\x0e\xd4\xda\x02[\t|\xbe\xbae\xf2q\x01W\x02\x89\xeb\x80\x1d\xe5:c\x89\xfb\x98\x02\xa4\r\x8e\xa9\x90\x11\x87X\x8dCm\x85Em\xb5\x83S\xeb\xa0E$ZL!L\x85\xb0\x14\x9a\x03\"\x1eR\x11\xc4\xe3\xe0u\xd1\x19D}k\x96\xfa\xe5\xfa\x9a\xe6.VB\xc4\x17-
\xe9\b\x8d\xf8\xfd\xa3W\x03u7M)W\xc4U\x83\xeb\x85\xf8\x95\x80\xe6H\x94\xed\xe0\xc4lb\xe1\x18\xb1p\f\xb4\x00\xa9Ym\xc8\xed\u0417\x96\x1d\xfb\x93\xd9\xe2\x044M\xa3\xa6\xbc\x183V\xeb\xeeW]C\x13\xfa\u07dd\xc5T|i\xaa\xe3H\x8a\x8b\x8b\u0556-
[EUUe\u78a2\xa2s\x16/\xfef\xd0\xf5\xd7_w\xe4\x8b/\xbe\xd8\x02\xb0x\xf1\x12\x06\f\xe8/f\u035a\xed\rVO\xcc\xff\xb7)--
\x13{\xf7\xee\xe5\xeb\xaf\x17v\xfd\uaaef\x9e/(\xf86\xa8\xeb\x1a\xf7\xdcs7\u00c7\x8f@J\u06db\xf4\xfc\x89\x91R\x92\x96\x96\u0189={!\xd2r\xf9f\xde\\\xaa\x0f\x97\x10\r\xd7\u04a2\xc7\x10\x02\xc9M0-
\x13S\xbaU\v\x13u\x81)\xa1\xca\x12T\xdb`)H\xf0\xe9\x04tw\xa5\xa7\xac\x8bn\b7\xce]'\xecn\xefN\x894\x152,\x91\xd5\x0eN\x8d\x83Y\xe3P[e\x13\x89H\x94\xa5\xa8W\xfb\xb8\xf3\x16\xf1e\xf9\xaa>D#\xdc\xfct\u074dq\xfb\xfc\x01\x8cP\x10\u007fB\b\u007f(\x84\x91\x10\xc4\x17\f\xa0\a\xfd\xe8\x01?F\xc0\x8f\x11\xf2c\x04\xdd\x14\xc7@(H
!\x84?\x18\xc0\xef\v\xa0\x1b\x06\x1aZ\xbd\x90\u05c9:\u01c9=\x8eB\xd8\xeee\x84c;D\xa3\x11L\xd3$\x98\x94F\xb3\xfc\xae\xb4\xe8\u060f\xacV\xdd\x10B#Ru\x183Z\x8b\x94\x0e\x9a\x86\x9b\xe9s\xdcITJ\x89\xae\xebB\xd7u*\xca\xcb\xed=\x85{\xb4\xe2\xe2\xe2\xdc={\n/\xbc\xe8\xa2\v;,^\xbcd\xf9}\xf7\xddW=k\xd6l\x1e~\xf8Am\u07bc\xf9\xde\xe5\xa9\x17f\xf9\xdfd\xc1\x82\xb9\u0525\"\xbe\xfd\xf6[\xa3\u05af\u07d0\x020`\xc0\x00F\x8d\x1a\x15\xbf4\xb5\xbc#\xfb\x1f\xc0\x8eE\x89\xe0#\xbf[_\xfa\x8c>\x9f\xd9o=\u02eeU_\xb3\xe7\xdb\xc5t8\xf5B\x1ca\xe0H\x87\x04\x1d\x92\xfd\x10\x8b\x82\xa3\x14\x8e\x04\xbf\x00\xbf\x13W]\x9fpSFp\xc5W:\xb8jo*\x94%Q\x96D8\xa0\x1c\x85\xb2%\xa6\xa3\x88\xd8no\xcf\xfa\x90\b\u2a17mPwE\xf7\xf9\xd1\xfd\x01t\xbf\x1ft\x81\xe3XH'\x86\xe9DP\x96\x85f[H3\x8cY[\x89\x15\x8b\xe2\xd8\x16\x8e\x1dC\xd96B\xd30|\x01|~?~\u007f\x10\u007f\xb0\x11>\

u007f\"\x9a\x1e\x00\u0347\x1e\xf2\xe1\x0f%\xa2\v\x03iK\xac\x98\x89\x8a\xc5pl\u06dd\xb9=\xbaA
`\xc7C?\x9a@\tI$Z\t\x9a\xc0\x1fJ\xa7\xcdIg\x91\xd3~
E[\x97\xb3i\xc9\x14\xf6o[IeY\x11\x8e\xed\x8ec\xa1\u05657R_\xafF\xd34t]7\x84\x10r\xff\xfe\x03\xce\xe1\u00c7}ee%\xe3\x0e\x1c8\xd8w\u07bc9w\r\x1b6\u20fb\xee\xba[\xe6\xe6\u62bd{\xf7*/\t\xe0W\x10\xfe\xfc\xb9n\xf8\xfb\xef\xbf\xdb\xea\xe5\x97_\x99\xbft\xe9\u0496Bh\xbc\xf2\u028bL\x9cx\x95\x97\xbd\xf2\x9ft\x02\x02\xaab6[U23\xe6\xaf\xe0\xc5\xdfM\xa0\xacp\a\x9dF\x8c\xe5\xf4\xbb\x9f')=\v'\\C\xcb$A\x93\x00\xec\xa8V\x14V+b&$*H\xd4\xdcE6\x9a_C\v\xb8\x82ET\xa2\xa2\x12,\xe9f\x95HE]B\x9e\x9b#\xae\x88\u064a\x98T\u023a\xd4\x17\x11\x9f\xbcT\x12)\x14\x9a\u03c7\x16L@\xf3\xf9\xb0c\xb5\xc4\"\x15\xc4j+\bW\x16SY\xbc\x9b\x9a\xb2}\xd4\x14\xef%r\xf8\x00\u044aR\xccp\r\xb6\x19E\u019bI+\xe5\x06\xf6\x15\xc4K\b\xb8\x8e^\xf7\x05\t&&\x93\xd0$\x83\xa4\xf4l\x92\u04f3Ii\x9aG\xe3\xb4<\x12\x933\t\x06S\b\x05\x1ac\xe8A\x1c\xd3\u008a\x85ql;\xbeJ\u050d\xe9\u0539w\x15\u03d0Q\x024C\xc7\b&
\x84\x86\x15\xae\xa4\xb8p=\xdbW~I\u0476\u5517\xee!RS\x8eR2^`L\x1c\xb3}\x9a&\u0404@JG*%\xb5&M\xd28\xf1\xc4nj\xc0\x80\xfew\xdc{\xef}\x8f\xd7\x1d\xab\xfb\xef\xbf\xcf\xf8\xdd\xef~\xeb$'\xa7zN\xdds\xe6\xff[l\u0738\xf1\xe2\xc2\xc2\u0096\xb6\xed0d\xc8\x00\x86\r\x1b\x06\x80\xe3H\f\xc3[\xe9\xf9\x9f@\x01~C\u00e8\x8e\u04aa\xf3\x89t\x1dz\x16\xf3\xdf\xfc\v{V/\xa2t\xfb\x062\xb2\xb2\xc0/\b\u06cajKP\x1e\x83\x98\x03\xd2V\xf1\x15\x92n\xf8A\x85\x1d\xa4\x16?\x01[\n\xe1\xb81\v\xd5\xc0mK\xa9\xea#)\x86\xe1\xc6eL\xa9\u0710\x8cP(CG\x0f5\xc2\b\xf9\x89\x85\xab\xa8,\xd9N\xd5\xc1]\x1c\u07b9\x8e#{6pd\xcf\x16j\x8b\xf7aEkp,\xeb\u01e9\xb3%p\xcb\x05\x04\x13I\xcdjE\xd3\x16'\xd04\xa7\x13\xe9\xd9\xedHMoIJZ\v\x92R\u04f1b1\xa2\xd1p\u072d\x8b\xfaP\x90\x92\xee\x0fR9\u011cj\x84a`\xf8\x12h\u07a6\x1f\x99\u037bQQ\xbc\x8b\xa2\x1d+8\xb8\xb3\x80\xb2\x03[(/\xd9E\xa4\xb62^hL;f;t\xdd\xd0\x04\x8a\xf2\xf2\n\x16/^\"\xaa\xaa\xaa\x1e\x9b8\xf1\xf2\x93\a\r\x1a\xf4\xc8\xe5\x97O\\&\x84\xb0\xff\xf4\xa7{\xf9\xec\xb3O\xb4s\xce9W\n!\u0639s\x1b\xad[\xb7\xf3\x06\xb2'\xe6\xffy^y\xe5eq\xed\xb5\xd7)\xa5\x94\x187\xee\xc2\u02cf\x1cq\xb3(\xce8\xe3\fZ\xb4\xc8'\x1a\xadE\u05fd\x95\x9e\xffI1\xf7\xe9\x1a\x8d5\x8b\u018d\x1a\xd1i\xd0\b\n\xbe\xfa\x90\u0292\x03I_<\x936\xbd\ac\x18AJ\xc3\x11bRa\xd5\t\xb8\xed\u01b9\x95r\x85]\xd8\xf1\xec\x10#^|K\xa8\xfaKF7Y$\x9eQ\"@\xd7\xdc<r\x9fO\xe0\xd3\x05\x8e\u07cf\x13\fPc\u0654\xec\xddJ\xe9\x8ao)\u077c\x9a\x92M+9\xb2c#v\xb8\xfa{\xb7=\xadI\x13\xb2\xb22IKK\xa3Qr#\x1a'7&\x18\n\xe1\xf3\xf9\xea\xcb\x16(\xa9p\xa4C4fR]\x1b\xa6\xa2\xb2\x8a\xc3G*(;|\x98\xe2\xe2b\xa2U\xe5\u0626\x85mV\x10\xa9\xfa\x96\x03[\xbf\x05\xc0\x1fJ\xa2YnGrZ\x9fDv~\x0f\x9a\xb7\xe9I\x93\u0336h\xc2G,\x1a\u01b6\xad\xa3\xb5g\xe2\x93\x05\x02\x81t\x1c,;\x8c\xe9H\x90\x92Fi\xb9tn\x9aG\xfb\x93N\xe7\xc8\xc1\xed\x1c\u06b3\x86\xa2\xed\xcb\u067f\xb3\x80\x9a\xca\xd2x\xa3\x0e\x83\xba*1B\b|\x01\x1d)\x1d\u05ae[O\xd1\xfe\x03g\x14\x15\x15\x8d(((\xf8`\u035a\u056fw\xef\xdes\u02581\xe7I7\x12\xe5u@\xf2\xc2,\xff\x03\xfc\xedo\xef\f}\xfc\xf1\xc7\xe7\xad_\xbf\x81\x13N8\x81\xbf\xfd\xedmz\xf4\xe8\xe9\xd5_\xf9/\xa0\v\xa84\x15\xab*|\xec*\x8f\xf0\xf6\xdd7\xb0r\xda\xfbdw<\x91\t\xcf}FJ\xf3<\xaa\xaak\xb0l\x85\x13U\xc4b\x12\"\x12\u007fD\x11D\xe1fZ\u01d7\xcf\xeb\u008d\xdd\xc4S\x05\xad\xb8s\xf5\xe9\x1a\xba!\x10~\x81\xf0k\xe8!\x1fzb\x00-\x045\x95\x11\xb6,\x9b\u01f6%s9\xb8~\x05e[\xd7
\x8f\xcbd\xd2CId\u7de5]\xfb\xf6tn\u05ca\x0e\xadZ\u0432y\x0e\xe9\x19\u9924\xa4\x90\x94\x94DRR\x12\x81@\x00]\xd7\xe3c(\x9e\xc5\x1e\x8fQG\xa21\xaa\xaak(\xaf\xac\xe2Hy9\x87J\xca\xd8{\xe0 ;\v\x8b\u0631k7[7nd\xff\xaem\xa8\u0631}\x9f5\xcd
\xb7m/Z\xb6\xefK~\xc7A\xe4\xb6\xeb\x87?\xd4\x18\xd3r\x1buH\xc7F\xc5\xeb\xa6+w\xd9\x04\x8ec#\x1d\v\xa5\x1c\x84r\x1b\xa8\xe8\x86\x1f]\u04e8)?\xc0\xa1\xddk\u0675a\x1e{6/&\\S\xee:uMw\xb

3n\xe2a\x17Pn\x88G@vV3:w\xeet\xa4g\u03de\x9f^{\xedUogdd-
\xae\u06fe6\xed\u06b2`\xde\x1crs\xf3\xbc\xc1\xec9\xf3\xff\f\xaf\xbd\xf6*W_}\r\x00\xabW\xaf\xba\x
b0\xac\xac\f\x80\x01\x03\xfa\u04ed[Wl;\xea\t\xf9\u007f\x01\xa9
\xc9'h\uace8JkL\xbb^\x83(\x981\x99\u00c5;)\\\xb3\x94F\xd9y(K\xa0*m\x88J\xfc\xb6DJ\x81\x94\n\x
d3Q(\x01\xbav\xd4UH\x01\xb6\x06\xb6\x00\xe1\x13\x84\x82:FPC\xf84\x8c\x84\x00F\x82\x8e\xee\
x87\xca\xe2#l\xfeb\x1a[\xbf\x9e\u039e\xd5\xdfP]v\xa8\x81E\xd1I\xca\xc8&\xbf[O\xba\xf4\xeeO\x9
7N'\u042du.\xeds\x9b\x92\xdb,\x1d\xdf1Y\"\xeer\xd3\xfa2\x04R\x1e\x93\t\xe5f\xc1\b\x12\x13B$7J\
xa4yv\xf61\x1e(\xec8\x1c,;\xc2\u03bd\a\u063ck/\xeb7nf\u00f7\xab\u067af\x15\x15\a\x8b\x90\u04a
6p\xeb2\n\xb7.c\xdd\xe2\x8f\xc8m\u04d3\xb6\xddF\xd0\xf6\xc4\xd3HHi\x8e\xed\xd8\xd8f\xbc\xf3R
]\x9c^\xb9\xce\x1c)\x91\xca\x01%\xb1cQ\x04\xe0\x0f\xa5\u042a\xebH\xb2[\x9dD\x8bv\xfdY\xbf\xe
c#\x0e\x16\xaeG\xc5\xdb\xeb)%\xe2\xb1t\r\xc3\xe7\a\xa5\u053e\xfd\x87TIiY\x93]\xbb\xf7\\\xb5i\u0
4e6\v\xfe\xfa\xdc3\xcf\xff\xdfon\xfa\xb3\x10\xc2\u06b1m;W_}\x9d\xf8z\xd1B5x\xd0\x10o@{\xce\xf
c?K\u07fe}Vm\u0630\xb1\xa7\xa6i<\xff\xfcs\\r\xc9e\xde\xc4\xe7\u007f3\xd4\"\xa0$\xaaXU\x1d\xa2
\xe0\xbbM\xbcy\xebe\xec\xdb\xf8-
=\u03ba\x94\xb3\xeex\x05\xabV\x11\xad\b\xe38n\n\x9f\u012d\xaf\"l\xe5\xe6j\xc7\xf3\xb8\x95\x00\
xe9\xd3\x10\x8du|\x89:\x81\x90\x8e\xcf'\xd0\f\x83@\xa2\x0fM@eq\x19\x1bf\u007f\xcaw3>\xa2h\x
c3jb5\xf1\xc6=\x9aARF\x16\xf9]{\xd1u\xf0H:u;\x89\xbc\xdc\x1c\xba\xb4\u0320u\x8aA\b\x00\x1b\xc
b4\xb1l\xc7\x15\xbc\xffG=\xf5c\xbb\x11\x1d\xfd\xacuY%\xa0\xd0\x04\xfc\u007f\xec\x9dw|UU\xba\
xfe\xbfk\x97\u04d2\x13\xd2h\xa1\x86^D@)\n\x88\xbd\x01\x96\xb1\x97k\x1d\xb1a\x1f\xcbXf\xece\
xf4\u07b9\xe3X\xc6^\u007f\x8c]\xb0PT\x8a\x02\x82\bH'\xa1CBB
\xbd\x9c\xba\xcb\xfa\xfd\xb1\xf7>I(\x8ewF$\xced}>\x87\x84\xe4$\xd9\xe7\ucd5e\xf5\xae\xe7}\xde\
xe7\xd5T\x15M\xd7A\xe8H\xa02nR\xbc\xbb\x8a\xcdE%,]\xb6\x8c\x85\xb3f\xb2j\xf1Bj\xcb\u02fc\u
07c0/\x10\"\xaf\xfbP\x0e\x1d}\x01}\x0f\x9f@Zf\a,\xd3&\x11\x8f8]\x93l\x13\xcb4\x90\xb6\x89\x94n\
u027f[\xf4$\xa5\x8dD\xa2\xeb\x014M\xa5z\xd7fV-
\xf8;\x9bV~I\xa4\xa1\nE\xf3\xa3*\x9a\xa3\x9dW\x85\xfb\xde\n\xa4\xb4\xedx<&\xc3\xe9ij~~w\x86\r;
l\xe6]w\xde\xfe`~~\xef\xef\x00F\x1d9J\xfc\xfe\xee;\xe5\u99df\xd9:\xa9[#\xf3\x037\x16-
Z\xc0\x91G\x8e\x01\xe0\x95W^>\xff\x89'\x9e8\xa4\xa1\xa1\x811cFs\xec\xb1\u01f0\xbf\xf6^\xad\u
35cb\u03b3\xfd\x82\x8e\t\x83.}\x06\xd2}\xc8H\x8a\xd7\xfe@\xc9\xea%Tm\\OV\x87~(\x8a\xc3A;\x
05\x94N\xc1\x8e\xd4@Z\xb8\x05C\x8e\u02e0/\xa8\xe0\xcf\xd0\xd13T\x046\x9a\xee#\x10T\xa9/\x
aff\xf5\x17\x1f\xb2\xf4\xe37(Y\xb7\f3\xe1P)\xbeP\x98.\x87\x1c\xc6\xe0\xa3Of\xd4\xf1\xa7\u043bO
\x1f\xdae\xa6\xd19\x03rT\b\xd8\x06\x981L[\xe2)\x19\u007f\xea\tn\xafNH\xce\x17\x9b}\u0352`%\r\
x14a\xa0\nAn@#\xb7k;\x86vm\xc7o\xc6\f\xa5\xf2\xb7\x17\xb1rM\x01\xd3?\xff\x9c93g\xb2~\xdd\x1
a\x12\xb1\x06\xb6\x15~K\u0676\x15\xac^\xf8\x1e\x83\x8e\xba\x88^\x87\x9eLZF{\x8cd\x8cD,\xd1\
xe8\xb4h7z\xd1x\x1d\xed\x04\n\x86\x11\u01f6u\xb2;\xf6c\u0504\xdb\xc8\xcd\xeb\xcd\xea\x85\xef
S\xb9k\v\xb6\xb4P\x15\x1d\xdbvr\x11\x8a\xa3\x84W\xfc\xfe
\xd1XB\xae][(jjjOI&\x8c\xe3f\u0318v\u07a9\xa7\x8e\xffd\u1885\xb2\xbc\xbcB\x00\xf2\xb9\xe7\x9e
c\u04a4l\xad\x13\xfb\xd7Fy\xfe\x1a.\xf2\xd5W_K}\u07b6m\xdb\x1b\n\n\n\x8e\xa8\xafo\xe0\xf4\xd3
O\xe3\x82\v.r\xb4\xc3?\u04a0\xb7u\x1c\u0623\x9d\r\xf8\\\xaa\xa4\x16\x9d\xf2\xcaZ\xd6/\x9aM\xb
4\xae\x86\xbc\u0783\xe9\xd4{\b\xa6ab\xdb\x0e
I!\x90\x9a\xc0\xf6)\u062a+\xd3S\x05\xfe\x80B\u042f\xa2\xa9\xa0\x06T\x02\xe1
\u04b0\xd9\xf8\xedl\xbe\xf8\xcb},x\xebi\xaaK\xb6a[\x16\xbe\xb4\fz\x1fy\x1c\xa7^\xfd;.\xb9\xe3\x01
N\x9fp\"c\xfav\xa4\u007f\xb6\x8f.\xbaA\x86L\xa2\xd9\u03bc0\xed\x03\xdd$\x8e\x94\xfb\xa2e\xd9

H\xdb\x00\xdb@U
\xcd\x1f\xa0g\xe7<N9\xee\x18\u019f6\x81N]z\xd2\xd0`PQ^N<VOME\x11[V\u03e1|G\x01\xc1\xb4\f2s\xba\xa0\xf9\x02\x98F\x02\xe9\xedt4\xda\x10H\xaf\xbb\x92\x14H\xdb\xc62\x92h\xfe\x10\xed\xbb\r\"\xbbC>\xd2J\x12\xa9)#\x11m@(\x9a#\u007f\x942\xd5<[Q\x15!\x84\"kjk\xc5\u039dej\xf9\xee\xddg\xdes\xcf\xef\xd7|\xfc\xf1\x94\xf5\x9f~\xfa)\xd3gL\x17\x17_t1\bv\x99\xcf\x00\x00
\x00IDATO\xff\xf5if\u0398\xd9:\xc1[\xc1\xfc\xe7\x1dEE[\xf9\xdf\xff}\x1a)e\x9b\xbf\xfd\xed\xf9{\u05ed+\xe8\x18\x0e\x87\xe55\xd7\\-
\x06\r:\x14\xcb2Z}X\x0e2\xa0\v\xc0\xa7\b\"\xb6N\xbd\b\xb2f\xfeWT\xef,\"\xb3}Wz\x1fq2\x96e9\xf2BM
C\nvH%\x99\xa6`\xe9\n\xc2r\x8a\x88\xfc\x9a\x82\x8aDS5\xfc\x19!\xcawlf\xde\xcb\u007f\u2aff\xdeO\xc9\xdaeH\xdbB\xf1\a\xe9?\xf6\x14&\\{'\xe7\xdf|\x0f\xa3\x8e>\x82\x1e\xd9A\xfa\a-
rE\x02\xd54\x1c\xe3*<;\x80\x83C=y\xc0nY\x06H\vE\x856\xe1\f\x8e\x18q\x18\xc7\x1cs\n9\xb9\xf9\xd4\xd4\xd4SV\xba\x1d\xcb2\xa8,\xdb\u0226U_\x11m\xa8\xa0MN\x17\xda\xe4vv]\x15\x8dT\u07511l${R\x04\xa9Db\xdb&RB\x9b\xb6]\xc9\xeby8\xe9m\xdaa&\"\xd4W\xef\xc42\x9dF\xd7N\x929E\xbb\bEQe}}\x83\xa8\xa8\xa8\xd0*+\xab\u03bc\xf9\xc6\x1b\xd7L\x9f1c\xfd\xe4\u0253\x99\xfa\xc9\x14q\xf9e\x97\xb7N\xecV0\xff\xf9\u01e0A\x87\xf0\xc9'\x9fq\xec\xb1c\xc7~\xf8\xe1\u01d7\x97\x94\x94\xa6\xf5\xea\xd5S\xdcr\xcb\xcd\xe4\xe6\xb6u\x16L\xeb8\xa8C\x02>\x15,\x14\xea\xf40k\x16\u007fK\xe9\x865\x04\xc2\x19\xf4\x1du\x12\xc1p\x1a\xaa_\"\xd2U\x94t\x15\xcb'0\x85@*\x02\xc5\x02\xdd\x06](\x04\x02!\x04\x925\xdfLa\xfa\xd3w\xb1j\xc6\xfb$c\x11\x00\xf2\x0f\x1b\xc3\xe97\xde\xc3\xf97\xdd\xcdQ\u01ce$7\x18
\xcd6\xc9S\x93d*f\x13?\x97\x83\x03\xe2\xfb}o\xa4t73\x03\x81Mvv:\u00c6\re\xe0!c\xc9\xca\xe9Bee\x05U\x95;1\x93qJ6/\xa5l\xfbJ\xd2\u04b3\xc8\xed\xd0\x1bE\xd3\xddJ\u0426\x15\xa5\xa4<\v<'I\xc7\xc2\xc2@\xf3\xa5\u046e\xeb
:\xf7\x18B\x9b\x8cl\xea\xabJ\xa9\xafs\xc4\x02\x8a\xa2:\u0564Bq\xc4C\xaaB,\x1e\xb7*++}UUUgN\x9cxU\xe1\xec\u0673\v\xde}\xf7=\x1ex\xe8\x01q\xe7\x9dw1y\xf2\xe4\xd6\xc9\xdd\n\xe6?\xdf8\xf7\xdcs\xc5\xc7\x1fO!?\xbf\xc7y+W\xae<\xa3\xa2\xa2\x82\x91#Gr\xcd5\x13QU\r\xd3l\x05\xf3\x960\x14@U\x15\xaa-
\x1f\x9b\xb6l\xa5\xe0\xdb\u0668\xbaN\x8f#\x8e!+\xbf;>\xbf\x81\xeeW\xb0qx\\\x81\xdb\xf4AJt\xa9\x92\x9e\u0786H\xed.\xe6\xbd\xfb\x14\xb3_\u007f\x88\x8a\xed\x1b\x01\bf\xb5\xe5\xc4\xcbo\xe4\xd2{\x1e\xe3\xb8S\x8f\xa3mZ\x10%\x91D\x97&\x1d|6\xed\xfd\xb6\xe3\xef\xf2\v!\xb8\xa2(h\x9a\x86\xae\xebh\x9a\xf7\xd0\xd045\xd5\xd1j\u007f'E\xdbv\"iU1\xe8\xd0>\x87C\a\x1fI\xbf\xfe#\xf1\xf9\xd2).\xdaB<\x1e\xa1\xbe\xba\x94\xe2\r\x8b0\x8d\x18\xb9\x1d\xfb\x11J\xcf\xc62\x93N\xe5g\xd3>\xd6{6\xb5\xb6\x9d\x1e\xa7\x96\x99$\x18\u03a5k\xcfat\xea\u069fD\xa4\x9a\xdd;7!m\x1bU\xd5R\xbdS\x05\x02EU\x95D<aU\xd7T\xfbjjjO\xbf\ubbbb\xea\xa6M\x9b\xf6\xfd\xd7s\xbff\xf2\xe4\u027c\xf1\xe6\x9bb\xea\u0529\xad\x93\xbb\x15\xcc\u007f\x9e\x91\x97\x97'\xbe\xff~\t=z\xe4\xffv\u04e6\xcdCjjj\x12g\x9du\xa6v\xf2\u0267\"\xa5\xd5j\xaa\u0542\xf8\x16]\x11D\xd1(\xad\x8e\xf0\xc3\xdc\xe9\xc4\xebk\xc8\x1f6\x9a\xbc\x81\x87b%\x93XR\x90\xb0\x04\xba\xea\xd02B@(=DZ
DY\xe12\xbex\xf1\x0e\x96\xcex\r#\x11\x03\xa0\xef\x91\xc7\xf3\xdb?>\xc5o.\xbf\x8a\ue773Q\xa2\t|\xb6I\xa6.\xe9\x14\x90\xe4\xfa$\xaaH\u016d\at\xf8|>4\u034f\xaaj\b\xd7\xd9PJ\xdb}H\xb7\xb7\xa7\xe6\xf4+\xd5t4\xcd)\xe0\xd9\xef\xe2SM\xfc~\x83\xee\u077a2d\xe8\x18\xf2{\f\xa2\xbc|\x17\xa5\xa5[I\xc6\x1b\u0631y)uU\xc5d\xb7\xcf'\xabm7\xc7j\xc0\xb6HeB\x9bR/\x82Ta\x95\x94\x12#\x99\xc02mr:\xf6\xa4k\xfe`\xccX\x1de%\xeb1-\xd31\x1b\x13\x8d-

\xedTUU\x12\u0244U]]\xed\xab\xaa\xae:\xf5\x81\a\x1e\xe8>\xe5\xe3\x8fg<\xf8\xe0\x83\xd6\u0529Sy\xe6\xd9g\u014c\xe9\xd3[\xe7w\v\x1f-
^\u0372y\xf3\x06\u0473g\x1f)\xa5\f\x9dt\u0489\ud28b\x8bIK\v\xa9\x83\x06\rr#\x9d\xd6\xc4g\u02e1\x13@\x936m\x03\u042dG/\xd2s\xdaS\xbei-\x95E\x9b\x10@\xd4\x14\xc4-
\a||\x8a@\x15\x92@z\x1a>\xbfB\xe1\x97\xd3\xf9\xea\xd9\xfb\xd9Q\xb8\u0519\x98iY\x9c}\xd9U\xfcv\xd2\rt\xcb\xefJ<\x9eD$\"\xa4\x05\x14\xc2\x1a\x844\x89.R\x01\xe9\x01\xa5H\xfc~?B8K\xa5\xba\xba\x82M\x9b6\xb3z\xf5j\xb6n\xddFEE\x05\x91H\x04]\xd7i\u06f6-
\xf9\xf9\xdd\x190`\x00}\xfb\xf6\xa1]\xbbv\xf8|\x8e(2\xb1G1\x91t\x9b\\\xf8t\x81\xa64\x90\xdf\xddG\xe7\xff:\x8d\x11\xc3\a\xf1\xea\xab\xcf\xf1\xe6\x9b/\x90\x887\xb0\xee\xfbO\xa8\xad(b\xec\xe9w\x90?\xf0X\xe2\xf1(\x89D\xa4\x11\xc4]\x95K3^I\b\x14T,\u06e4\xa6\xa6\x92P\xb8#\xa3'\u0702/\x98\u0392\xf9\uf4ccG\xf1\aB\xa8\x8ak?\x8cDU55\x96HXK\x97\xfe\xa0J)/\xaf\xaa\xaa\n\x02\x17\x00\xdcx\xc3\r<\xf3\xec\xb3\xdcx\xc3\r\xad\x93\xbc\x15\xcc\xff\xf91g\xce\\\x01\xc8m\xdb6\xf7\x8dD\"=M\u04e4c\xc7\x0eb\xe0\xc0\x81\xad`\xde\xd2\xc0\x1c'!\x97\ud0ce\xedrh\u06f5\xa7\x03\xe6;\xb6QW\x17#j\xab$-
\x13]\x11\x98\x96M(\x14\x00i\xb0x\xf2\xeb\xccy\xe91jv\x16\x03\u042d\xff\x10\xae\xbf\xedn~{\xe9y\xe4\xf8
a$0\x83NoO\x95\xc6f\x12\xf2\x00'8\x9d\x06\xd6\xe9\x80Ea\xe1Zf\xcc\xf8\x82\xa9S\xa7RX\xb8\x9e\xfa\xfa:\x12\x89d\xb3\xf9'\x84\xc0\xef\xf7\x93\x96\x16\"??\x9f\x13O<\x91q\xe3Ne\u0108a\xf8\xfd!\xe2\xf1H\xeayMA]QUTa\x11\xf0'\x181\xbc;C\x06?\u0148\xe1\x83x\xfc\xf1\xc7\u067c\xb9\x90\x92-
\u02d9\xf6\xd6\xef8\xf6\xcc\xdfs\u0211\xe7\xa2(\x82x<\x82\xeb\xf4\xd2\xecd\x84\xe7\xad.\x1c\xe2K\u0692h\xb4\x81`(\x97#N\xb9\x01\u0757\u0192y\xef`\x1a\x06\xc2\xe7kTZJ\xd0\x14U5MS~\xf7\xdd\x12\x11i\x88\x9c\xff\xf0#\x8f\x88\xfb\xee\xbd\xf7b!\x84\u0675k\x17\x15\xb0\xa6L\x99\xc2o~\xf3\x9b\xd6\xc9\xdeJ\xb3\xfc\xdfG\u06f6m\xc5\xf2\xe5+d\u06f6m{~\xf7\xdd\xe2\v+++\xdbt\xef\u078dI\x93&\x89\xb4\xb4tL3\xd9z\x17[\x16\u04c2OU\xa9L\n\x96\xafX\u0156\x1f\x16\x92\x9e\u06d1\xbca\xc7!\x82\x19\x98\x96\x89%%\x81@\x00U\x1a\xcc\u007f\xe3\u007f\xf8\xea\xb9\ai\xa8*\a`\xf8\t\x13x\xe2\xcf\u007f\xe1\xca3O \xa4\x82iD\x11\xb6Ds)\x19\xb7\xc7\xc4\x01\x05q\xaf\xf9F
\x90\u01ae]\xa5\xbc\xf8\xe2K\xdcw\xdf\x1fx\xf7\xdd\xf7\u063au+\x91H\x04\u06f6\xf1\xf9|4iE\xdaX\xfa\x1f\x8bQZ\xba\x93\x05\v\x160e\xcaTv\xec(\xa1G\x8fnt\xec\xd8\x19!\xc0r{\x95\xee\x19\xa9;\x15\xa8&>\x9f\xcea\x87\r\xe6\xb0\u00c6\xb3~\xc3V\x8a\x8b\xb7\x92\x885P\xbc\xf1;\xd2\xc3Yt\xee1\x14E\u0571,3\xf5\xa6\xefS\x13/\x1b?1M\x03\u0557F\xe7\xfcCQ\x05\xec,ZC\xd2H8\xfe.\x1eG\x83@(\x8a\x90H\xb9{W\xb9\xa8\xae\xae\x1eX^Q\xdeo\xf6\xacYS\xfb\xf5\xebg\u03593G\x1d7n\\\xabl\xac52\xff\xe7F8\x1cV\x00[Q\u0120\xf4\xf4\xb4.\x00\x9d:u\x12\xa1P\xa8\xf5\xee\xb5\xd0\xe8\\`\x93\x1d\xd2\xe9\u0439\v\xa0PWUNmu%9\xb9\x9d\x90R\x12Ho\x83\x99\xa8g\xce\x1bO0\xef\xf5\xff\xc1t\xbdT\x8e=\xe7\x12\x9ex\xe21F\xf4\xec\f\x18$\x13\xc6>[\xc7\x1dX\xaaH\xba\xc9L?\x8b\x17/\xe2\x91G\x1ee\xe6\xcc/R\x1e\u22a2
\xa5t\xf9s\r\xd3\xf5?Q\x145U5\xea\xd9\x02H)\xa9\xa9\xa9\xe1\x85\x17^d\u039c9<\xf6\u0623\x9c}\xf69\xa8\xaaJ<\x1e\xdf/\xa0'\x12Q4Me\u0318\x91<\xfb\xcc\v<\xf0\xe0\xc3|\xfe\xf9;D\x1bj\x98\xf5\xc1#\u0636\u0270\xe3~\x8b\xaa\xa84D\xebphtoW\x91\xcd{\xeb5\":\xf1D\x84\x80?\u0110\xa3/\u00d6\x92\xef\xe7\xbeE\"\x1e\xc5\xe7\x0f9\x85Tnd\xaf\xaa\x9a\xb0\xb1\xe4\xbau\x05B\xd5\xd4\xf3\xc2\x19\x19H)/\x11B$g\u03de\xad&M\xd3:\xf5\xe4\x93['{kd\xfe\u007f\x1b\xfd\xfb\xf7UV\xacXi\x0f\x1c8p\xdc\xe6\xcd[\x8e\u077d{\xb7y\xdcq\u01ea\xa7\x9d6\x01U\xd5Ze\x89-

qR)\x02C\xd5X\xb1a;\x8b\xbf\xfa\x1cEQ\xc8?\xf2D\xb2\xba\xf6D\xd5|\b\xcb`\xd1\xcb\x0f\xf3\xed\xebO9v\xb4B\u5b097\xf1\xa7?=\xcea\xdd\xdac\x9bq\x92\xc6/\x9f\xd4\xf6\xf8qU\xf51s\xe64n\xbc\xf1&\x16,\xf86\x05\xe2\x9e\u007f\v\x80i\x9a$\x93\xc9\x14h{\tPUUS\x80\xdfT\xd1RYY\xc5\xe7\x9fO#++\x93\xe1\u00c7\xa3\xaa\xaa\xbb\x11\x88\xfdP<\x12)M:uj\xcf\xe0\xc1G\x11\x8dI\n\nV\x10\x8f\xd5S\xbcy\x19i\xe9\x99t\xeb3\x02M\u04f0\\\x9d9{)hd\xaai\x87\x87\xeb\x96e\xa0\xeaA:v\x1d\x88\"MJ\xb7\xaf\xc60\x92NR\xd7;\xfdH\x10\x8a\"\x84\xa2\xc8]e\xbbD}]\xdd@\xa1\xd0y\ua529\x9f\xbc\xf5\xd6[R(\xaax\xe0\x81\ax\xef\xbd\xf7Z'{+\x98\xff\xf4\u0467O\x1fu\xed\xdau\xb6\xdf\xef\x1fQZZzb}}\xbd\x9c0a\xbcr\xdcq\xc7\"\x84lU\xb2\xb4@\x9aE\x11
u\x8dU[\xcaX0}\nf,B\xaf#O\xa0\xeb\xd0C0c&\u07ff\xf1\x14\u07fd\xf6\x84\xa3\x9fV}\\t\xe3\x1d<\xfa\xd0\xfd\xf4o\x1f\xc6H\u01b0\xec\x83s\x92\xd7u\x1dE\u04595\xebKn\xbc\xf1f\n\n\n\xc9\xcc\xcc\xc4\xef\xf7\x13\x8f\u01db\x81s\xd3(\xbd\x11\x80\xed\x94U\xc0\x9e\xf3RQ\x14\x92\xc9$\xb3f\u0362]\xbb\xb6\f\x1b6\u048d\xec\x8d\xfd\x02\xba\x94`\xdbI\xb2s2\xe9\xdb\xef\b\xa4\xf4\xb3b\xc5w$bu\x14m\\JFf[\xba\xf7\x19\x01B`Zf3,oz]\"\xa5g\u013d6\x03\xcd\x17\xa2C\xb7C\xc0JPV\xbc\x163\x99L\x9d.\x9a<U\b\xa1\xc8\xd2\xd2RQ[W7\xf4\xa1\x87\x1fJ~\xf4\xe1G\vV\xad\\\xc9\xdau\xeb\xc5\xfb\x1f\xbc\xc7\x1bo\xbc\xd1:\xe9[\xc8h\xd1\xe6\u07f1X\x84\xf7\xdf\xff\xc0\x06H&\x93i\x91H\x04EQ\xc8\xca\xcaF\b\xb5\xb5\xea\xb3\x05\x93->
\x10\f\xa1\xea:F\xb4\x1e;\x11!\xe0\x83\x95\xef?\xcf\u04b7\xfe\x1b\xcbH\x82\xaas\u044dw\xf0\xf0\x03\u007f\xa0WN\x90x,\xc2A\xc2q7\xaa\xf6\xb1aC!w\xdf}\x0f\x1b6l$\x1cN\xa7G\x8f|\xc2\xe1\xf4\xbd\x12\xed\xd9\xd9\u0664\xa7\xa7\xa7~\xb6\x11(-
L\xd3\xdc\u06e4\xcb\xe5\xe1\x93\xc9$\xbf\xff\xfd=\u03181\rP\xd0u\xfd\x1f\x9c\x16\x04\xc8\b]\xbb\x84\xb8\xe6\x9a[\xf8\xedUw\xa3\a\u04895T\xf2\xe5\xfb\x0f\xb3z\xf1T\xc2\xe9Y\xa4\x87\xd2\xd1u\xdd1\xd9J\xa9\xf8\xd9\xef&\x11\x8fGA\vq\xf8qW0\xec\xe8\vP5\x95d\"\x86e\xd9)\xf0O9\t\b!\x97|\xbf\x94\u0253\xdfy\xe4\xed\u0253\xcf\x03X\xb3z\x85\xfc\u04df\x9e\x14\x85\x1b6\xb4N\xf7V0\xff\xc7c\u04e6M*`I);\x1fr\xc8\xc0\xe3\xea\xeb\xeb\xd0u]\xe4\xe4d\xe3T\xbd\xb5*YZ\x1e\x8c;\xc5;\x01'\u034f\xe2v\xc41j*X\xf3\xf1T\xbe}\xe9\x11\xe2\xf55\b\xdd\u03f97\xdc\u0243\x0f=@\x8f\xccF\"\xbeW\x03\xe3_r\xf8|~\xa2\xd1z\x1ey\xe4Q\x96.]\x86\xcf\xe7#++\x9b\xea\xeaj\x8a\x8a\x8a\xf7z~CC=\x86\x91\xdc+\n\xb6m{\xafy)\x84H\x15\x13\x01\xd4\xd4\xd4p\xd7]\xbf\xa7\xb8x\x1b\xaa\xea\xfb\t&q\x82\xa0?N~\xf7
\xd7]w+\xd7^s\x1f\x81`\x98\xba\xea\x9d\xcc\xfc\xfb\x1f\u0670\xf2KBi\x19\xa4\x05\xd3\xf0i:\x9a\xaa!\x14\a\xd0=P\xf7\xca\xf9\x9b\x02z\"\x1eE\rf1\xe2\xe4\xab9\xfc\xd8\v\xd0\xfd\x01,\xd3\xc0\xf6\x1a`\v\xc5K\xae\x8ah,\xce\u04a5\u02d4\x8f?\x9e\xfa\u03b4\xe9\xd3\u007f\x030c\xfa4\xa9\xbb\xcd`J\xcbv\xb6N\xfe\x83<Zt\x02t\xf7\xee\xdd\u07a7\xc1D\"\x99fY6~\xbf*22\xc2{-
\xa2\xd6\u0442\x00\u0775km\x17RP\xa4\x03l\xeb\xe7M#V\xf7\xff\x88\xd68\xa5\xe5'\xfd\xd75\xdc~\xcf}\xf4\nkX\xc9(\xf6Atcv\x80V\xe1\u33e7\xf0\u99df\x01\x90\x99\x99I\"\x91$\x16\x8b2t\xe8\x10\xbav\xedJVV&\xa1P\x88\xb2\xb2],Y\xb2\x84\x1d;JP\x14e/\xf0\xde\x17g\xee6`\xc60\f,\xcbb\xf5\xea5<\xf7\xdc\xdfx\xe2\x89?\xe1\xf3\xf9H$\x12?\x1aIK\x01>=J\xe7N~\xae\xbcr\x12\x91H=o\xbe\xf1\x14\x95\xbb\xb60c\xf2}\x9c}\u074bt\xea>\x04a[\u0130\x91\xb6\x8d\xad\xe0\xf6\vu\xb4\u4379P\x91re\x8cE\x1b\b\xa5\xb7a\u0109\x13\xc1\x96\xacX0\x05\xcb4\x10\x9a\xee\xde\x11\x05\x90h\xba.kj\xeb\xc5\xf7\xdf\u007f\xafdee\xbe\xbbf\xed\xdas\x0f\x198\xf0\u04de={\xb5v-
j\x05\xf3\u007f<\xaa\xaa\xaa\xbcO\x8d\x86\x86\x86$8\xf6\xa5>_\xa0\x15\xcc[\xf0\xf0\xa0-
S\x93\xe0&\xa8\xb7.[\x90\x02\xbd\x11\xa7\x9e\xc5\xed\xf7=\xc0\x90v\x01l3\x86y\x90o\xa3\xae\xeb\xd4\xd6V\xf1\xfe\xfb\xefS[[K\x9b6\x19\b!\xd04\x95+\xaf\xbc\x82\x1bo\x9cD\x8f\x1e\xbd\x9bD\xd

f\x06_~\xf9%\x8f=\xf6\x04\xdf~\xbb\x10UU\xb1,\xa7C\xa9\xa7\x86\xf1d\x8a\x9e\xc2\u0176m\xd2\xd2B\xa8\xaaB$\xe2\x14\x10\xbd\xfb\xee{\\~\xf9e\xf4\xeb7`\x9f\xfe\xe9\xcd\xdeS\x1b\x14\x05B\xc18\xf9\xddC\\w\xedmTW\x973\u58d7(\u0776\x8a\xd9\x1f<\u0339\u05fdB
\xad\r\xa6i`\xa9\x16\xa6\xe54\xbd\x964\xa9P\xa5\x89\xf2\xc5\x05\xf5x\xa4\x81P8\x8b\x91']\x8d\xb4,V,\x9a\x8ai&Q\x15\xcd\xdd\xe8\x1c\xe3\x05]\xd7eyy\x85\xf8\xe6\x9by\xbe\x8cp\xf8\xb5\xfa\xfa\xfa\x13\xc3\xe1\xf0r!D\x13O\x81\xd6\xd1J\xb3\xecc4\x8dVL\xd3LE8\x9a\xd6\xda\xe7\xb3eG\xe6\xce\xc7t\x1d|\xba\x9a\xa2\x1f\x00\xfa\x1d6\x92\xbb\xfe\xf8\x00\xa3{d\xa1\x99qLK\x1e\xe4k\x95\x80\xca7\xdf\xccc\u0672\xe5\xf8|:\xa1P\b\u02f28\xed\xb4\xf1<\xfe\xf8\xa3\xf4\xe8\xd1\x1b\xdb6\xb0m\x03\xcbJ\xa2(\n\xa7\x9c2\x9e\xc7\x1e{\x94~\xfd\xfa\xb8\xba\xf1\xc6\xdf\xe9\xa9^\xf6\xe4\xd2\x13\x89d3\xcae\xfb\xf6\xed|\xf8\xe1\x87xA\xca?\xdc$m\x87\xfe\b\xf8c\xf4\xef\x9f\u036d\xb7\xde\xc3\xf0\x11'\x00\xb0n\xe9t\xbe\xfb\xeaE\x14EC\xf7\x05\xf1i*\xba\xab\xac\xf1\xeco\xc1\x05w)\x91\xee\xc6#\xa5\xd3X:\x1ai@\x0fd1\xf2\xe4\xab\x198|\x1cB\x82i$\x1b7\x18\x97?\xf7\xf9\xfc\x94\x96\ucd27M\x9f\x91\xf3\xd0#\x8f\xbe!\xa5\xec\x04\xc8W^{\xad\xb5\xcdW+\x98\xef\u007f$\x93F\x93\x89\u072aZ\xf9\xb5\x8d\xdc\xecl\xb7\u035a3\xdaw\xcc\xe3\xc1\a\xefg\xc2\x11\x83\xf0[IL\xcb>\xe8\xe1\x9c\a\xac\xdf}\xf7\x1d\xa5\xa5\xa5\x84\xc3a\x84P\xc8\xcb\xeb\u0239\xe7\x9eC
\x90F<\x1e\xc10\f\f\xc3\xc04M\x12\x89\x04\xb6m0f\xccXN>\xf9\x14\xb7xho\xde|O\xaa%\x99Lb\x18f\xb3\xa4\xe7\x9c9s\x89\xc5\x1a~r\xc3\f\xe9\xf2\u067a\x16a\xe4\x88n<\xf0\xc0\xe3t\xeb\xd6\a\x80\xf9\u04dee\u00ca/\b\x04\u00e8\xaa\x8a\xae*\u8aa3\xbaAx}Mid\xd2]\x91\x8b\rH\xcb\x01\xf4@(\x97Q\xa7\\G\xff\xc3N\u0136\r\x92\u0244\xdb|Z\xa4\x1e\xbaO\x17%;J\xf8\xfc\xf3\xcf\x0f}\xf4\xf1\xc7\u07d6R\xe6^u\xe5\x95\xd6\v/\xbe\xa4\xae\u07f0\xbeu\u2dc2\xf9>8
M\xdbk\xd1Ii\xbbG\xda\xd6\xd1b'\x95\x17y\x16\x15QSS\xedF\x9e\nw\xdd\xf9;\u039bp*\x8a\x11\xc30\x8c\x83\x0e\xe4RJt]g\xf7\ue76c[\xb7\x0e\x80\x9c\x9c\\\x84\x10t\xed\u0695\xbe}\xfb\x02\xd6>\xf9`Oz8l\xd8\xe18\t\xf9\xc6!<WB\xd1L\xe3\x97\xea/\xda4j\u07fcy\v\u02d7\xaf\xe0\xff\xa2\x12\xf6|]\x14\x11g\u0729\xc3x\xe8\xe1\xc7\b\x85\xc2D\xeb+\xf9\xe6\xd3?SS\xb1\r\x9f\xdfQ\xdah\x8a\x82\xae\n\x14\xa5\xd1\xcf\u073b\x1e\xa5\xc9\x03\xe9h\xdb#\x91\x06\xd22:2z\xfc\r\xf49\xf4\x18,3\x81a$\x91M\x1cpl[\nU\xd3\u5dad\xdb\xf9\U0008bbce\xfd\xcb\xd3O\xdf\x02p\xed5W[\x93&\xdd$\xcav\x95\xb5.\x82V0o>\xd2\xd3\xd3R\x9f\xfb|\xbe\xd4\x02\xf4\x16Rk\u04a5e\x0e]\x0f\x10\x8fG\xf8\xdf\xff\xfdKJ\tr\xf1\xc5\x17s\xd5o\xafDJ\x93\xa4a\xee\xd5~\xed`\x81\xb9\x10\x1aEE\xc5\x14\x15\x15\xe3\xf7\xfb\xc8\xca\xca\xc40\x1c*%\x18\f\xfd\x03z\xc6I\x94\x06\x83\xc1\xbd\xc0\xdc)\x1e\x12\xfb\xa0J\xec\x14e\bN\x92\u007f\xed\u06b5\xff\x14\x95\xe5\x045&\x97^r6\xb7\xdcz+\x00[\n\xe6\xb3\xf4\xeb\xb7P\x14\x15Us\xfa\xa6j\x8a@W\x9c\x9e\xa7\u04b3Vt\xd5-
.{\x92\xb2I\xb0%D#u\x84s\xbar\xd4i7\x92\xdfo$\x96\x11\xc34\f\xa7\x85\x9d\x10^\xf7\"\xa1\xea\xba\\\xb1b%_}5\xeb\u0799_|1\x11`\xd6W_\xc8{\xef\xfdC\xeb\xc2l\x05\xf3=\x8e\xe9\xb99)|\b\x87\xc3>\x00\u04f4\x88\u0162\xadw\xae\x85\x9f\xa6^{\xed\xf5\x942d\u0630\u00f9\xfb\xee\xbb\b\x873\x89\xc7\xe3?\x99R\xf8\xa5\u01ae]\xbb\xa8\xa8\xa8@UU6m\xda\xc4\xee\xdd\xe5l\u0672\x95\xd2\xd2\x12@\xfdQ\t\xec\xb6m\u06e8\xaf\xaf\xdf\v\xe8\x1d\xaaE6\xfbZ\xd32\u007f\x0f\xf4\xe3\xf18;v\xec\xf8\xa7\xaf=\x99t\xe4\x91w\xdc~\vc\x8f>\x11\x80\u0173_g\xcb\u06af\xf1\xf9\u04f0\u0756q\xaa\x10(\u00a1X$`#\xdd*\xd3\xc6k\xf3\x14.\x96m\x13\x8b6\x90\u04f9?\u01de};]{\rs\xbay\xe16\x96\x16\x8a#}\x14\x8a0-
\x8b\x1f~X\xc1\xab\xaf\xbd\xfe\u0127\x9f}:\x02\xe0\xd5W^\x96\x0f=\xfcH\xebbh\x05\xf3\u0191\x93\x93\u3b46J\u02f2w\uae8ei\x9a\xb2\xb6\xb6\xae\xd9q\xbeu\xb4\x8c\u0474\xf0\u6957^\xc60\f22\xc2\\\u007f\xfdu\xf4\xeb7\x10\u00c8\xb7\xa8{\xe6\x81jmm\x1d\x91H\x14\u06f6\xa9\xact\x14Tk\u0

5ac\xe1\xddw\xdf\al\x02\x81@\xb3\x8aN\xc7d+H}}\rs\xe6|MUU\xf5>\xc1\xfc\xc7\xd4){&G\xe1\x9f;\xac8\x11z\x82\xcc\xcc,n\xbb\xf5v22r\x89\u0515\xb3\xe4\xeb7\x89\xd6W\xa2\xfb\x82\xce\xc6\xe2\xd4\xe6:\u007fW\x8af\u0562\x8e\x0eG`#\\\xf5\x8b\xc02M\"\r\xf5\xb4\xebv(\u01ddu\v\x1d\xbb\xf4\xc30\x12\xa9&\u04e2Io\u044a\xca*f\u035a\x93\xfd\xdcs/L\x9e:uj\x1f\x80?\xfe\xe1>~\u007f\xf7\u077c\xfc\xca\u02ed\v\xa3\x15\xcca\xc0\x80Av\x87\x0e\xed5!DUaa\xe1\xa2p8\x8ca\x18\xb2\xa6\xa6\u0199~-
\x98f\xf18S\xaf+\x8d\xcf\xe7K=4MsU\x06\xe2\u07ca*\xf2\"\xeeW_}\x95\x95+W\x010~\xfcx.\xbc\xf0\x02\xa0\xe5Z/4m0\xd1(\u0143g\x9f}\x8e\x97_~\x19!4B\xa10\xba\xae\xa3\xeb\xbak\x8bk\xf2\xec\xb3\xcf3\u007f\xfe\xfc}\x82vS^|_\xf7\xb8\xa9\x14\u0463i\xfeY\xa5\xadC\xdbH\x8e?\xe1x.\xbc\xf8*\x00\n\x96\xcd`\xcb\u06b9h\xba\x0f\u02d68i&\x97\xcbW\x1a#t\xefO\xca&\x9b\x83m;t\x8biZD\"\xf5t\xea3\x82c\u03fc\x91\x9c\xdcN\x18F\x1c[\u068dQ=\xa0\ubeac\xad\xad\xe3\u06c5\x8bz\xbd\xf0\xe2K\x1f\xaf]\xbb\xba7\xc0\x13\x8f?\u0395W\\\u054a\xb2\xad`\xeeL\xf8\x11#F\x00\u0436m[\xdb+\x9f./\xaf\xc0\xb6M\xa7B\xad\x05\x02\x9a\xdf\xef\xc7\xe7\v\xe2\xf3\x05\xd1\xf5\x00RJ\xea\xeb\uba69\xa9\xa1\xa1\xa1\xc1Q#\xe8\x81\xd4s|>\u07ef\xfe\x94!\xa5DU},\\\xb8\x80\x8f>\x9a\x02@\xe7\u039d\x988\xf1*\x02\x814\x92\xc9x\x8b\u0778\x02\x81\x00>\x9f\xaf\xc9\xe6\xea\x00qmm-
\xf7\xdcs/w\xdf}\x17\v\x17.\xa0\xaa\xaa\x8a\x9a\x9a\x1a\xbe\xff~\x11w\xddu\x0f\u007f\xfb\xdb\vMk!\xf6\xe1\x82(\x9b}\xdcWd\xee\xf7\xfb\xc9\xce\xce\xf9WW\n\x86\x11#=M\xe5\xa2\v/\xa5g\xaf!\u0636\u024ao\u07e7\xb6\xb2\x04\u055f\xe6F\u070d\xcfoI9\xd7\u0737E\xba\x91\xba-
A\xda`\x99&\r\xd1(]\xfa\x8fe\u0538\xab\t\xa5g\x92\x88G\x9a\x9c<\x04B(\xc2\xef\x0f\u0206\x86\b\xdf-
\xfe~\xe0S\xff\xfd\xbfS\xa4\x94}\x00~\xff\xfb\xbbZ\xf9\xf3_\x8a\xe2l\xc9\x17\xb7sg\t\xf7\xde{\x1f\x00\xfd\xfb\xf7\xd7JJJ(**\x92\xa5\xa5\xa5\x98\xa6\xe3\xfb\xdcbvEEA\u05ddb&\u00c8\xb3z\xf5r\xbe\xff\xfe{6l\xd8\xc8\xee\u077b\x89D\"\x98\xa6\x85\xaek\x84\xc3a\xf2\xf2:1p`\u007f\x86\x0f\x1fN\u07fe}\u071f\x95$\x93\xf1_]1\x94\xa7\n\xb1m\x83\u0253\xdfa\xf3\xe6\xcd\x00\x9cs\xce\xd9\x1c{\xec1\x80\xd5\"_\x93\a\xa8YY\x99dddP[[\x8b\xa6\xa9)\u0149\xa2(TTT\xf2\xc4\x13O\xf2\xe1\x87\x1f\u04ff\u007f\u007f\x82\xc1\x00\x9b7oa\u02d6-
\x18\x86\xe1\xf4\x1eu9\xf5\xa6\\\xf8\xfe@|\xcf\u0476m[\xbaw\xef\xf63\xdc\x03H$\x1a\x18:\xa4?\xe7\x9d\u007f%\x8f?z\x13[\n\u6ce5\xf0[\xfa\r\xff\x8d\xd3+\xd7J\xba\u0478H\xb5\x99s\xaeQ\xa6\xe8\x16\\\x80\xb6\x05(R`\xdb`\x1a\x06q\x04}\x87M
\x11\xade\ue53f\x92\x8cG\xf1\x05B\xde\x1b\x89\x00\x11\f\x86dMM=\xb3g\xcf\x1dx\xdf\x1f\xfe\xf0\xb6\x94\xf2\f!D\u0663\x8f=\xae\xa4\xa5\x85\xec[n\xbe\xb9\x15q\xffS\xc1\xbcc\xc7N\\w\xdd5\x12##\xbc\u057b\u6492\x12\x99L&\x85\u05d2\xeb`\x03\x99\xd7V,\x99\x8c\xf1\xd5W\xb3\x98<\xf9\xef,_\xbe\x82\xb2\xb2\x9d\xd4\xd4\xd4\xee\x17Hrrr\xe8\xd4)\x8f\x11#\x86s\u99572f\xccQ\xf8|AW\u07db\xfc\xd5P0\x8e[\xa0\x8fo\xbe\x99\xcb\xf4\xe9\xd3\x00\xe8\u07bd\x1bW^y\x05\xa0b\x181Z\xf6<\u02e3}\xfb\xf6l\u0672\x856m21\x8c\x9a\x94V\xdc;1m\u06b4\x89M\x9b6\xb9^\xe7\x0eMf\x9a&\xa6i\xeds\x83h\n\xec{\x82\xba'S\x04h\u07fe\x1d\u077a\xfd<`\xee\x9c2l&L8\x9d/f~\xcc\x0f\u02fef\xed\xe2)t\xe97\x96P\xb8-
V\xd4Db\xbb\x8d4\x1acq\xdbkL\xd4\u060c\u03b5\x10\xb0Ql\x81\xb0\xc10\x12(\"\xc0\xa0\xa3\u03a3\xben7\xdf}\xf16\xa6\x91@\xd7\xfd\xc8T\xb0/DZZ\x9a,.)\x95\x9f~\xfa\xf9\b\u02f4\xee\a\xae\xbb\xf7\x9e\xbbm\x80\x0f>\xfa\x88s\xcf>\xbb\x15u\xff\x13i\x16wQH\x80\xcc\xcc\u0302d\xd2\xd8\x04\xb0c\xc7\x0e\xdb\xe1\xcd\x0f\xae<\u0476mt]G\b\x95\u014b\x17q\xc5\x15Wr\xd9eW\xf0\xce;\xefRXXHMM-

\xaa\xaa\xee\xa57\xf6\x16{EE\x05+W\xae\xe2\xe5\x97_\xe5\x82\v.\xe4\xfa\xeb\xafc\u035a\x95(\x8a\xeeF\xba-_O\xefE\xe5\xa6\x19g\xea\u0529l\u06f6\x1d\x80\v/\xbc\x90A\x83\x86\x00\xcdU\x1d--27\xcd\x04=z\xe4\u04f7o_l\xdbIv6-
\xea\xf1\x80\xd8\xcbqx\xd2\xc2h4\x9aR\x92\xec\xeb=\xf9\xb1\xb9\xa9\xaa\x8d\n\x99\u07bd{\u04ebWO\xa44\u007f\x96\xe5l\xdbQ\x86\f\xe9\xc6Yg_\x8a\xaa\xeal-
\xfc\x96\xe2\r\v\x9d\xe8Y\xd3\x1dO\x00!\x90BA\xba2C\xa1\xb8\x92\u0166f\\\x12\x90\"E\xb7H\x04\xc9d\x02\u0157\xc6\x11\xe3&r\xe8\x91\u3476\x85\u0133\th,E\xf2\xfb\xfc\xac+X\xcf\xf4\x193\xaf\xbd\xe1\u019b\x9e\xf2\xae\xee\u0733\xcfn\xb5\xe0\xf8O\x06\xf3\x13Np\u0295o\xba\xe9\x96h\xbbvmc\xe0\xb8\xce\x15\x15\x15\x1dT0\xf7\"r\u06f6y\xf9\u55f8\xe4\x92\xcb\xf8\xfb\xdf\u07e5\xb2\xb2\xb2\x19X\xec+\u0269\xaa\xaacW\xea\x96[\x03\x94\x94\x94\U000b7ffd\xc8\xc5\x17_\xc2G\x1f}\xe0r\xef\xbe\x16\x0f\xe8\x96e\xa1\xeb\x01\x96/_\xc1\xb4iN\a\xf7\xfc\xfc\xee\x9cw\u07b9\x00$\x93\x89\x16{\xedB\b,\xcb\xc2\xef\x0f1t\xe8P\xc2\xe10\xb1Xt/\xe9dSu\u029e:\xf1\xfd\x01\xb9\xe7w\xbe/:\xce{NFF\x06#G\x8e\xc4\xe7\v\xfel\xc9a\xa7:\xd4b\xfc\xb8\xf1\f\x1b~\f\xb6e\xb2a\xf9t\f#\x86\xa6\xfb\x11(
\x14\x84\ua078\u04b8\x8e\x14\a\xd0e\x93\xe8\x1c\xc0\xb2%\xa6ec\xd96\x91\xfaz\xb4`\x16\xa3\u03f8\x8e\xfc~\xc3H\xc6\x1be\xc2\x12/w\xa2\n\xa1(r\xe3\xa6-
,Y\xba\xec\xf6?\xfe\xf1\xfe?\xed\x19\u0334\x8e\xff@0\x1f=\xfaH\xe9N\x82\x02\x9fO\xdf\x10\n\x05\xa9\xa9\xa9\xb5\v\n\nS\x8b\xe3`\x8c@
\r\xd34y\xea\xa9\xff\xe1\xb6\xdbng\xe3\u018d\xfb\\\u0626i\xa6\x00\xd9+&\xf1>\x06\x83\x01\xfc~\u007fJ%\xa1\xaa\x1a\xabV\xad\xe6\x86\x1bn\xe4\x85\x17^\x04\x04>\x9f\xbfEG3\x8e\xae\xdcb\xee\u072f\u0678q\x13\x00\xa7\x9dv\x1a\xfd\xfb\xf7k\x06n-
}\x1c\u007f\xfc\xb1\xf4\xe9\u04d7\xba\xba:TUiv\x0f\xffQ\xa4\xbd\u03c5\x95\xaaX\x96\xfb\x05\xf3\xee\u077bs\xe2\x89'\xa4Ny?O\x90!@&\xe9\u07ff\x1d\xa7\x9c\xea4^\u07b6~\x11\x15%\x85\bE\x03UE\xb8\u0560RU\x9c\xc2SE8\x0f\x0f\u061b\x81\xb3\xb7\x99\x81e\x83mY4\xd4\u0551\u046e\a\u01dc=\x89\xdc\x0e]\x88\xc7\"N\xc4\xdf\xe4\x87uM\x17I\u00d4\xeb\xd6\x15\xf0\xfd\x92\xa5wN\xfe\xfb\xe4\u06fck|\xfeo\u007fkE\xf4\xffD0\xef\u0631\x8b|\u5557\x04@\x9f>}k;w\xeeB\"\x91\xb0=\xe9\xdb\xc1(@\xf1\xfb\x9dD\xe73\xcf<\u01e3\x8f>FCC\xc3O\x8e\x9aTUu9W\a\xa8u]G\xd34TUC\xd75\xfc~?ee\xbb\xb8\xf7\xde?\xf0\xf6\u06d3QU\x1d\xbf\xbf\xe5\x02\xba\xdf\x1f\xa2\xb8\xb8\x98\xcf?w\xb8\xf2\xcc\xccL&L\x18\x87\xdf\x1f\"\x99\x8c\xfd*\x16\x81e%\xe9\xd7o
'\x9exB\xaa\x9d\x9b\xdf\xef\xdf\xe7\xe6\xdc\xf4\xb4\xf5cN\x87^\xe9\xfe\x9e\xf7\xdf;\xcd\b\x01'\x9ex\x02\x03\a\x0e\xe0gO\x10K\x89\xdf'9\xef\xdc3\xe9?`(\xf1\x86j6,\x9f\x81P\x15\x14UsA[`\xab\x02\u02efb\xeb*R\x15HE
\xf7B\x04\x99\x92\xb8XR:\xc9S)i\xa8\xaf\xa5c\x9f\xe1\x1cu\xfaU\x84B\xe9\xa9NE4\xb1\v\xf0\u9e88\xc5\xe2,]\xf6\x03\x9f|\xf2\xd9\xfd\x8b\x16-<\n\xe0\xfa\ubb93\x1f\xb8\x06c-
y\xac_\xbf\xee_\xfc\xf9\x82V0\xdfs\xec\u0631C\x02t\xed\xdau\x99\xa2(\x12\xf0\x17\x16\x16\x12\x8d6\xfc\xe2/\xc1\x89D\x15>\xfdt\nO>\xf9$\xd1h\x94\xb4\xb4\xb4\xbd\x9ew\xf2\xc9'q\xc7\x1d\xb73q\xe2U\xa9\x04\x97\x97PSU\x15)!\x1a\x8d\x12\x8b\xc5\xdc^\x92V*r\v\x04\x02TUU\xf1\xf0\xc3\x0f3o\xde\\\x84\u041a\xf9\u0534\x98H\xc0\x8d\xc6V\xacX\xc9\x0f?,\a`\xec\u06238\xf4\xd0C\x01\xfbW\x13\x95{\xa0{\xc5\x15\x971d\xc8\x10b\xb18\xc1`\xe0'\x95\xf3\xefo\xd3\xdeW\xa4\xdd\xe8^\b\x03\a\x0e\xe4\xca+/\xc7I\x10\xff\xbc}l\x1d\x9e\xdbd@\xff\x8e\x9c~\xfa\x04\x00\xb6\xae[@\xbc\xa1\x12U\xd7A\x11)\x9e\xdb\xd6\x15\x8c\xa0\x82\xa9\v,\r\xa4\xe2x\xa7\xbbt:\u04a3]\\Su\xcb+*\xb2m\x1a\xa2\t\xfa\x8f9\x8da\u01df\x8em\x19\u0636\xebc\xd3D_\xafi\x1a\xd5\xd55|\xbbpQ\u01bb\xef}\

xf0\xb6\x94r(\xc0\xb9\xe7\x9c\xd3\"\xe6\xc7\u018d\x85\xfb\xfd^\u07fe\x03\xfe\xa5\xdf\u0777o\xff\xbd\xbe\xb6e\xcb\xc6\x03\x87M\xbf\x86\xc56d\xc8`\x00\xce>\xfb\xcc\xd5\x1f~\xf8\xe1.\xa0CQQ1\x1b7n`\xf0\xe0\xc3\xfe\xa1\x17\xf4\xcf\u0271\xaa\xaa\x8f\xe2\xe2\xed\xfc\xf9\xcf\u007fa\xf7\xeerrr\xc8\xcd\xcda\xfb\xf6b\f#\xc9\xd1G\x8f\xe5\x96[nf\u0528#\xc9\xce\xce&\x91HPPP\xc0\x9f\xff\xfc\x17\xfe\xfe\xf7wH&\x13dffb\x18&\xf5\xf5\xf5\xa9\xeb\xb6,\vEQ\xf0\xf9\xf4\x94V}\xf3\xe6-<\xf1\u0113\f\x192\x84\x8c\x8c,,\xabe\xd9\x18\xe8z\x80X\xac\x81\u0673g\x13\x89D\x00'\xdal\xdf>\x8fX\xac\xe1W\xa3\x9dw\xfc~\x92\xf4\xee\u074f\x9bo\xbe\x89k\xaf\xbd\xce\xf55w|Z\xf6\xe4\u021b\xea\u01fd\xc20E\x11n\xbb\xb8\x1f{\xbf4\f#Izz\x1a\x93&Mb\xc0\x80AHi\x1e\x90\xbc\x88\xe1Jw/\xba\xf0l^}\xe9\x19\u028b\v\u0639u\x19=\x0e\x1bO\xd2H\x80\x90\b\t\u00b2As\xa3r\x01\x12\x05M\u06a9\x86\x18
\x91\xb6\xa7F\x97\u062e\\Q\x95\x02#\x99\xc0\fe0\xea7WP\xb6\xbd\x90\xf5\u02d7\x12\b\xa55\xd9\t\x9c\nQM\xd3)/\xaf\xb4\xe6\xcc\xfd\xba\xdb\x1f\xfex\xff\x93R\xcaqB\b\xe3\x8c3~sP|\u043f\xfcr\xa68\xe9\xa4S$@\xef\xde\xfd\xfe\xd1\xdc\u040b\x8b\xb7\x1d\xbbl\xd9\x0f\x9d~\xf8\xe1\a\xa3\xacl\x97\x15\b\x042\xbbw\xefv\x88\xcf\xe7\u03c8D\"\xbb7l\u0638\xda\xef\xf7%{\xf4\xc8W\x87\f\x19\xa2\x0f\x1e|\xe8\xf6v\xed:\xce\xde\xd7\xef\xf3|\xf1\x9f~\xfa/b\xe8\xd0!r\xec\xd8c\xfe\xb3\xc0|\u0528Q\x00\xf4\xea\xd5o\xfe\xf8\xf1\xe36\x86\xc3\xe9\x1d\x8a\x8b\x8b\xf9\xee\xbb\u017f(\x98{\x94\u039bo\xbe\xc5\xf7\xdf\u007f\x0f@,\x16\xa3\xb8x\a\xe3\u019d\u00ad\xb7\xdeB\xbf~}\xc9\xcdm\x0f@2\x19E\xd7u\x86\x0e\x1d\xc6s\xcf=\x83a$y\xef\xbd\x0fH$\x92)u\u011e\u05a9\u0264\x81\xdf\xefD4\x86\x91d\u039c\xb9\xbc\xf7\xde{L\x9cx-
\xba\xae\xff\xecQ\u073f:v\xef.g\u07bc\x0584Xo\x86\r\x1b\xf6\x0f#\u05d68L\xd3DUU.\xb8\xe0<\u05ad+\xe0\x89'\xfeD\xc1 \xe3A\x00\x00
\x00IDAT]]\xad[\xca\xefT\x88\xee\xebu)\x8aB8\x1c\xc64M\f\xa3\xa1\xd9s\x9a\xde_]\u05f1,gS\x988q\"\xff\xf5_\x17\xe1\xd4\x15$\x0f\xd8\x06\x050\xa0\u007f_F\x8d\x1e\u00e7\x9f}\xce\xf6\xc2\x05\xf4\x1dq:BU\xb0\xb1Q$\xf8\xa5\xc0\x92\x12Kq\x146R\xb3\xb1m\x81\xb0%\xb6\x02B*\bl\a\xe8\xdd\r\xc0\x96\xa0HP\x10D\x1b\x1a\xc8\xed\u0715\xa3\u03fd\x92]E\u06e8\xad\xac$\xe0\x16\xf7y\u01b9\xaa\xa2
\x84\xaal\u07bc\x95\x99_|yBfV\xe6\x8bR\xca+\x85\x10\xf2\xfd\x0f?P\xcf;\xe7\xdc\x03Z\x1a\xbcr\xe5\x0f\f\x1e|X\xea\xff\x1e\x90\xbb\xefS\xa0\xa4\xa48TXX8z\u0252%\xf9\xb5\xb5\xb5\x87\xf9\xfd\xfe\xa3\x8b\x8a\x8a\xa2\xf5\xf5\xf5\xf6\xc9'\x9f\xa8\x1b\x86\xd5.\x12i\bF\xa3\x11\x99L\x1a6\b_(\x14\xd4\x15E\xc54\r\xa2\xd1X\\Q\x84\xbd`\x81_|\xf4\xd1\x14%\x1cNo8\u7733\xcb{\xf5\xea\x95\x00\xb9\xb0c\u01ce\x9b\x86\x0e\x1d\xb2}\u0420C\xe6ge\xb5\xdd\rp\xf3\u0377\xc8=\xe9\xbb\xff\b0o\u05eec\xea\xf3^\xbdz}\xb3j\u056a\xa3\x8a\x8bw\xb0`\xc1\x02&N\xbc\xaa\x99\xd4\xeb\xc0\x82\xb9\x8f5kV\xf2\xd1G\x1f\x11\x8b9\x15\x8d\u0468\x13-\x8f\x1e=\x8a1c\u01ba
\x1e\u00f6mW\xfa\xe64\xf7\xcd\xc8\xc8\xe2\xb6\xdbn\xe5\xeb\xaf\xe7\xb1k\xd7.4M\xdb\xe75\u06f6M\"\x91$\x10\b\x10\b\x04\x88Fc\xbc\xfe\xfa\x9b\x9cv\xdait\xe8\xd0\t!\xcc\x16\x01\x94\xde\u01b6l\xd926mr\x12\x9fc\u018cf\xf0\xe0CS\r\x1c~m#\x91H\xe0\xf7\x87\xb8\xfb\xee\xbb0M\x83\xe7\x9e{\x9ex<N(\x14\xc40\x8cf\x16\xb6M\xefW\xd3\x13\u05be\x005\x18\f\xa4\xb4\xf8\x97]v\x19\xf7\xde{7\xe9\xe9mH$\x0e\xecI\u02f6\r4=\xc0Y\xe7\x9c\u0367\x9f}\u038e\rKH\u0517\x12\nf\x93HDQ\xb0\x91B
U\x89\xa5J\x92&\x18\x16\xd8:(\xa6DH\xe9D\xe7\xaa\xe2\x14\x18)\xce\u05fcf\u03f6\x04,I]]\x9c^G\x1c\u03c8\x93\x971\xeb\xdd\u05f1L\x13U\xd3\x1c\xcb\x00\xe1Q?R\b\x81\xbd~\xfd\x06\xe5\x9bo\xe6_\x91\x97\x97W\x06\xdcs\xde9\xe7Ze\xbb\xcaD\x87\xf6\x1d\xfe\xe5I\xbdn\xdd\x1a1o\xde|\xa5

\xaa\xaaRy\u8847e<\x9e0\x81f@.\xa5l\xf7\xc2\v\u007f\u02c8D\"c\xb6o\xdf6\xf2\x9ak\xae>\xb6\xa
a\xaa\xaa\xe7\xb6m\xdb\xd5\xda\xda\x1aa\xdb\x16\x8a\xa2\x92H$\xb0,\v\xc30\x89\xc7cn\x9eC\x
b8\xef\xabt,\x9c\xa5\xed\xe6\xc0\xf4\x80s\x10\x91\x9eT\xd7\x1f\x0e\x87sV\xacX\x010\xd8\xe7\U0
00d16dbK\xf7\xee\xdd\xe3\xb7\xdf\xfe\xbb)\x1d;\xe6\xbd\u007f\xd2I\xc7\xfd0h\xd0\xd0\"o\u04ff\xe
f\xbe{\xb5+\xae\xb8\xdc\xee\u0673\xb7\xfdo\r\xe6M\xc7\t'\x1c\xff\u0357_~u\x13\x90\xb1j\xd5\x1a\
u05ae]\u01e0A\x83\x81\x03\x1b\xb1z7r\xe6\xcc/(,,l\xa6J\x00\xa7$\x1c
\x12\xa9C\u04f4\xbd\x8c\x944\r\xfa\xf5\xebO\x97.\x9d\u0675k\u05cf\xca\xdbl\xdb\xc60\f\xfc~\xc7\
xf6\xb7\xb0p=_}5\x8bK.\xb9\xec\x17;\x85\xfc\xe3\u0701\x0eX,X\xf0m\xca5\xf0\xf0\xc3\x0f'\x18L\xf
fUQ,{\x03z\x94\xb4\xb4\f\xee\xbf\xff\x8f\xe4\xe4\xe4\xf0\u05ff>CII\t\xc1`\x10U\xf5\xa3\xaa*\xc9d2\
xb5\x89\xefi\xc9\u073c\xef\xa7s\xc2J&\x93dggs\xed\xb5\xd7\xf0\xbb\xdf\xddF\x9b6\xd9\xc4\xe3\x
91f\xf3\xea@\x9d6|>\x9d\xa1#G\x93\x99\x95\u016e\x1d\x85T\x94\xae\xa6\xef\xb0\t\xd4V\xc6\x01
\x81%,\xa4\x10(\xd2F*N\xa7\n\xa9\bPAZn\x8b9\xd7\xfaV\n\u0464\xdc_\xa2I\x81*!\x19K\x12\xb5\x
d2\x18>\xe1|6.[\u0136\x8d\x85(\x9a\xe6\xb0-
\x12\xa4\xf0\\\xd4Ql[\xca\u014b\x97\x88\x8cp\xf8\x8e\x05\xdf~\xbbl\xcc\xe8\xd1\x1fuh\xdfA\xd6G\
xea\t\xa7\x85\u007f\xf2k+,\\K\xbf~\x03\x91R\xf2\xc9'S\u0122E\x8b\x94\x01\x03\x0e\xb1\x00\uf05
42\f\xa4O\x9b\xf6y\xc6\u018d\x1b\xf2KJJ\x87M\x9cx\u0571555\xc7l\u0738l)/\u07cdiZX\x96I<\x1e
O\x15\x8a\x99\xa6)\x15E\x11N#\x10\x87B\u06f3VDU\xb5\x94$\u007f\u03e0,\x99L\xb2{w9R\u06a
e\xd8Ae\u01ce\x12\u05af_\x1f\xf0\xfb\xfd\x17v\xeb\xd6\xed\u0082\x82\xb5e\xf7\xdc\xf3\xfbWF\x8
d\x1a5g\xfc\xf8\xd3\x16\n!\x12\x8f<\xf2(\u007f\xfd\xeb_\u052b\xae\xfa\xad\x15\n\x85Y\xb7n5\x03\
x06\f\xfa\xf7\x05\xf3\t\x13N\x9f\xfb\x97\xbf\xfcu]l\u024e#\n\v\v\x99?\u007f\x01\x83\x06\r>\xa0Me\
xbd\u0098\xfa\xfa\x1a\x96.]F<\x9e@\u04f4f=\x1e\xcb\xcb+R\xc7\xee\xfdo\b\xec3\x91\xd9\x14\x04
\xbc\xd7\xe0\x1d\xfb\x03\x81\x00\xd5\xd5\xd5\u031b7\x9fK.\xb9l\x9f\r\x84\x0f\xceP\xa8\xaf\xafa\u
0672\x1fp\xb8\xc7\xde\f\x1e<\x98\u007f\x87\x11\x8fG\b\x06\u04f9\xe3\x8e\xdb\xe9\u07ff\x1f\xcf?\
xff\x02\xf3\xe7\xcf'\x1a\x8d\x10\x0e\x87IK\v\xa56\\\xa7\xefg\xa3\u4c39\xbb\xa2#M\x1d3f\f\xd7\\s\r
\xe7\x9f\u007f.\xaa\xea#\x91\x88\xfe2zk\xefd\x90\u04c1\x01\x87\x8fd\u1b19\xec,Y\xc9\u19ddOL\
xd6b&-0\xc1\xb6l\xb0\x04~U\xa0\xa2:RD\xd5y-
\xb6\x05\xb6pZ\xcba:\xe0.\x15\x1c\x8d\xba\"\x10\x8a\xc46-
jj#du\xec\xc5\xf0\t\xe7\xb3\xf3\xf9'\xb0L\xc3\xdd\xf0\x1b\xbdn\x1c;^!b\xf1\x18\x8b\xbf_\xaa\xf5\xe
b\xd7\xf7Y)\xe5:!DA8-
\u0316\xad[\u945f\xff\x0f_\xd6\x17_L\xa7_\xbf\x81M\u05cb\xf4\x00|\xfb\xf6\xad\x87|\xfe\xf9\xe7\x
c3W\xaf^\xdd\xfe\x9ak\xae>\xbc\xb2\xb2*'\x91\x88w6M\xb3\xf7\xb6m\xdb(//'\x99Lb\x9a\x16\xb6m\
xb9\xf5\x03\x8dEa
\xd04]\xec\x19\xc4\xed\xfd9?b\x94'S\xbeQ^m\x82\xd360\x81\xaa*\xd4\xd4\xd4RXX\xd8!//\xef\xbe
M\x9b6\xdf\xfb\xf5\xd7\xdf|\xf0\xe1\x87\x1f\xbcu\xce9\xe7N\xbb\xe9\xa6[\xac\x9bn\xba\x85\x17_|\
x9e\x8d\x1b7\xff{F\xe6\xf3\xe7\u007f\xc3QG\x1d\x8d\x10\u00ba\xe3\x8e\xdf}SP\xb0\ue23a\xbaz\
x16.\xfc\x96+\xaf\xbc\x1c\x9f\xcfw\xc0J\xe0\x1d\x9f\x0e\x9d\x8d\x1b7\xa6|G\x9aj\u01e5\x94\xacZ
\xb5\x8a\xda\xda*\u06b4\xc9$\x1e\x8f\xeds\x12\x94\x95\xed\xa2\xbc|\xf7~\x8f\xe3M\x9f\ub045\a\
xfe\xabW\xaf\xa6\xb4t\ayyy?\x1a\xd5\xff\x12\u00fb\xc6m\u06f6\xa5\xbc\xb8{\xf6\xecI\u07fe}\xd8_
g\x9e_\xd3\x10B\x90H8\xf9\x8e\t\x13\xce`\xe8\xd0\xc3\xf8\xec\xb3\u03d86m:\xf3\xe7/\xa0\xb2\xb
2\x12EQ\xdd\xc8L\xa4\xf2}M;`\x85BA\x86\f\x19\u00b8q\xa7r\xfe\xf9\xe7\u04eb\x97\xf3\xde\x1chje
\xcf\xe0\xc1\xc6\xc6\x17J\xa3\xff\xd0\xe1,\x9c5\x93\x92\xad\xab\x88\x8bJ\u04bbeR_[\x0f1\x05b&\\
"ia\x1b\x02M\xd8H\xdbi\xe7\xa7\t\x05K\xb1\x1d\x8d\xb9\x05\xa6%\x91\xb6C\xb1\b\u06e1X,\x17\x

bc\xccx\x82\xb8\x11\xa0\xe7\x11\xc7\xd3g\xf17\xacY8\u05c9^\x9b\x18\x05\b!R\r.**\u02993\xf7\xeb\x0e}\xfa\xf4\xb9\t\xb8\x0e`\u02d6\xad\xec\xdcUF\xc7\xf6\x1d\xf6z-
\xc5\xc5\u06d86m:\xd7^{='\x9f<n\xcf\u0377\u05d3O>9~\u04e6-
G^{\xedu\x03l\xdb\x1eTVVFYY\x19\x89D\xd2\xcdg\x18)\x16\xbf\xf9zSR\xc0\xdc|~;\xf7\u054b\u029dG\xa3\x0e\xdfk\x8a-
]1~c\x91\x95\xe7&\xe99\x9b\x89\xd4)\xad\xe9\x89\u0272L6o\xde\u0136m\xdbD\u01ce\x1d\xcf+,,\xfc\xcd\xef~w\xdb\xfb\xe7\x9cs\u059bG\x1e9\xe6\xabk\xae\xb9\x1e\x80%K\x163|\xf8\xc8\u007f/0\xef\u0631C\x13\xaa\xe5\xb8\x17\xbe\xf9f\u078d\xbbw\xef\x0e\u035f\xff-
K\x96,\u1a23\x8eA\xca\xc4\x01\x05\x92\xa2\xa2bJKK\xf7\x99\b\x9b;\xf7k\xbe\xfbn1'\x9f|*\x81@\x90d2\x912k\xf2\f\xb8>\xfex\n;w\x96\xb9\xaa\x06s\xbf\x1bGS@\xf7\x80\xbb\xb4\xb4\x94M\x9b6\x92\x97\xd7\xf9\xe0\xc7\xe4\xee\xe9c\xf5\xea5)\xd7\xc0\u07bd{\x92\x9b\xdb\x1e\u00c8\xff\xdbT\xf99\x00`\u0429S\x17\xae\xbd\xf6z&L\x18\u03d2%KY\xb1b\x05\x05\x05\x85\x14\x15\x15S]]\x8da8ADFF\x06\u077au\xa3_\xbf~\f\x192\x98\xa1C\x87\u0437o?@!\x91\x88\x1e\xd0\xd3\xe3\xfe\x02s\xdb4\xc8\f\xf9\xe9;\xe0\x10\x00vo\xddH\xd5\xce\":\x1d2\x94x2\x06\x9a@\xf8\x05VRAIZ\x88\x84\x8dL\x9aH\xc3FJ\a\xc4l/\x02\x176\xb6\x05\xd8\x12i:\x85D\xd8\x02Ew\x14.\xb1h\x946Y\xed\x18v\xea9\x14\x17\xac\xa2\xa1\xbe\x1e\xdd\x1f\xf0\x10/\x05t\xb6\x9bP.,\\\xcf\xd7\xdf\u033bDJ\xf9\x8a\x10b\xd9\t\xc7\x1f\xc7\xd6\xed\xdb\xf7z\x1d_~9\x93.]\xba\xef\xb9N\xd4Y\xb3\xbe8g\xfa\xf4\x99'\x9fq\xc6Y\xa3\xaa\xaa*\xfbVVV\xb2k\xd7n\xa2\xd1(\x8a\xa2\u062a\xaa\n!R\xddNS`\xbb\xaf\xc0\u010b\u031d\x8f\x8eE\xb0\xd7(\xd5m\x85\x8d\xf4\xac\u007f\xbd\xc6\x1e^\x03\x12ic\xe36\xf7p\xbf\xef\xd9\r\xe3Ya\xa7TOJ\x93\xa8\x1d,\u02d2EEE\xb2\xa4\xa4T\u07fau\xeb\xc5\xc5\xc5\xc5g=\xfb\xec3\xafM\x9at\u00fdB\x88\xda\xe1\xc3G\xf2\xd4S\u007f\x12w\xdcq\x97\xfc\xb7\x01\xf3;\xef\xfc}\xea\xf3\x93N:\xb5\xcd\xdbo\xff]]\xbdz\rEEE\xbc\xf9\xe6[\f\x1e<\x98p8L\"q\xe0\x00\xbd\xb2\xb2\x8a\xda\xdaZ\x14E\xa4\xfcF<P\xaf\xae\xae\xe6\xb1\u01de\xa0S\xa7<\x0e9d0{\x9a\x80}\xf6\xd9'\xbc\xfe\xfa\x1bX\x96E
\x10H)\x1f\xf6G\xb7\xec\x19\xb5\xd7\xd4\u0532cGI\x8b\xb8\x17M\xc1\xbc\xa6\xa6\x86@\xc0O\xff\xfe\xfd\xf7\xb9\xc9\xfd;\x8cX\u0331-\xee\u0739+\x9d;wc\u0084qTUUSWWO,\x16sU*\x82@
@\x9b6\x19dee\xb9\x9e\xe7\x0e\a\xef\xd5\x17\xfc\u049b\x9c\x04\xa4m\xd1\x06\x18\u0437\x17\x999\xb9\xd4W\x95\x13\xad*G\xd7AE`\t\xa7x\b\x9f\x00]E\x04\x05\x18\x02\x12\x16\xc4,DR\xa0Z6\x8a\x00[*$m\xdbQ\xb3\xe0x\xb6\xa4\u0339\xe26\t\x9fI\x9d\x9a\xa4\u00e0\x91\f\x18}<\x8b?\u007f\x1f\xfc\x01/\x94\xf5\f\x19\u075f\x17\xd4\xd7\u05f3t\u9cb4\x0f?\x9er;p!@~\x13\u04f1-
[6\x88\x1e=\xfa\u0213N:\xa5\xe9z\xe8\xf8\xfc\xf3\u03dez\xf9\xe5\x97]\xbfe\xcb\xd6\xc1\xa5\xa5\xa5Zee\xa5\x97\xb714MS}>\x9f\x02({6\xd4\xf6\xd6X\xf3\xfb
RHo#\xb1M{\xaf\xf5g[\x0e%c\xfd\x9f<\x86\x9a\xffm\xab\xc9\xdaQT\x15MU\xbdz\x13\xe1\xf0\xf3\xd2\u07bau\x9b]^^\x11,))\x99TYY1DJy\xb9\x10b\xd3\x1dw\xdc%O;\xed4\xf1\xfc\xf3\xcf\xc8=7\xb5_\x1d\x98O\x9d\xfa1g\x9ey\x16\x00\xf5\xf55C\xee\xbf\xff\xfe\x0f\u05ef_\xef\x8f\xc7\xe3\x00\xbc\xfd\xf6d\xbav\xed\u01ad\xb7\xdeL \x108`\x9cr\"\x11wmO\x95\xd4\xed\xf1\xa2-
)%\xf3\xe6\xcd\xe3\x82\v.\xe2\xea\xab'2z\xf4h\xc2\xe10\xe5\xe5\xe5\u03181\x83\xb7\xde\xfa\u007f\xec\u0739\x93`0\xf8\xa34\xc9\xfe\x12\x9c\xb1X\x8c\xda\xda\xda\x16\x03\u6595\xa4\xb0\xd0)\xb8\xc8\xcb\xcbc\xc0\x80\xfe\xfc\xbb\x0eo\xf3J$\x1c\xfa\xcc\xe7\xf3\u047e}\x1e\xed\xdb\xef\xef'l\xa7\x89\x83;\x0f\x0ff2\u061bJ\x83\xf2\xf38\xe6\xb8\xe3\xf9t\xea\x14j\xcaw\x82\x04U\x15(\x96\x03\u0336G\x1d(\x02\xe1S\x10\xba@\x06U'R\x8f\x99\u0204\x85\x86\x8de\xb9d\x85l\xf4b\x91\xa6D\x899\x96\x00I#F\xa8]\x16\x87\x9et6\xdb\xd7.\xa7\xacx\x1b\xbe@\b)\xad&\xe0\x99\xf2\xba\x91\u06

f7o\x17\xb3f\xcd>#\x1a\x8f\x8d\v\x05\x82\u04fd5\xf5\xd8c\x8f\xea=z\xf41\xdc\xff\xa7\xbf\xf7\u07bb}\u05ef/\xbc\xf8\xa2\x8b.\xbcp\xed\xda5\x1d***\xa9\xac\xac\xf6\xfa\xb5J\xc5\xe11\xf4=\xd7~s\x8a\u0109\x99m)\x9b\xb9FJ\x01\xd86\xb6e\xa5(\xa6f \xa9\xfb\xd0C!B\x81\x10z
\x88\x1e\f\xe1\x0f\xa5\xe3\v\xa5\xe1\xf3\a\x91\xaa\xcf\u9deai(B`\x18\t\x92\xb1(\u0246:\x12\xf55\xc4\x1bj\x89\xd5V\x93\x886\xa4\x92\xac\u0496\xae0Bs\xe7\x87T\x84\x10JCC\xbd\\\xb9r\xa5hhh\x18\xbdk\u05eeO?\xff\xfc\u04f3'L8\xbd\xe0\xb3\xcf>\x93\x17_|\xe1\x8f\xea\xf2\u007f\x15`\xee\x019\xc0_\xff\xfa\xccog\u0318\xd9s\xe5\xcaU(\x8a\"-
\xcb\x12\xc9d\x92\xb7\xdez\x9bc\x8e\x19\xcb\u0631cS2\xb2\x03\xc3\x15\x8b}\x82\xae\x97\x10]\xbbv\x1dw\xdf}/m\u06f6\xc5\xef\xf7\x13\x89D\x9a\xa9W\"\x91H\xb3\xe3\xf6\xfe\"q\x0f\x04\xfc~\u007fJ9\xd1r\xe8\v\x85\xf2\xf22v\xedr\xf8\xff\xdc\xdc\\\xb7\xcaU\xfe\u06fb\xe2I)I$\x12\xbf\x9e\xebu\xe3\xc2v\x99\xe9\f\xe8\u06db\xa9\x96EEi\x11\x89H\x12MU1\x15\v\u06e5\x01\xa4t\x13\x9d\xd2%\xdc\x15\x10A\x05|:2\xa1
\xa2&:N\xa4\xeaq\u0136\xed\x16|J\x89L\xd8`@\u012a\xa5}\xf7\xc1\x1c2v\x1c\x15\uff88\xb4\xf7P\xfb8\xa5\xa5(B\x88\x9a\x9aZ\x96,Y\x1a|\xe7\x9d\xf7\xae\x91R\xce\x12B$\xcf8\xf3L\xed\xd3O>1\x00\xdeyg\xf2\xb8\a\x1f\xfc\xe3\xbd?\xfc\xb0r\u052aU+ihh\xa0\xa1!B2\u9038\xa6iBJ)\xf6\x06p\x1c{\x01o\x9d\xb9fa\xa9^\xacV\xf3\x80J\xf3\xf9\b\xb6\xc9\xc1\x9f\x1e\u019f\x96\x8e?-
\x03\u007fZ\x06\xa1\xac\xb6\xb4i\u05d1\xb4\xb6y\xa4\xe7\xb6'\xb3m\x1e\xa1\xac\\|\xe1L\xfc\xc1\x90\x03\u07b8.\x94\xb8'\x16\xe9P\xa4\xb6\x91
QWE}Y1\xe5[\n)[\xb7\x94\xe25K\u067d\xb9\x00\xcb2\xb1,'I\xae\xeb\x1a\x9a\xa6\xba\x1b\x8c\x10\xb6m\u02cd\x1b7\x8a\xfa\xfa\xfa\xfe55\xb5\xd3\x16,\x98w\xe7\x981c?\xbc\xe0\x82\x8b\xe4s\xcf=\xc7\xf5\xd7_\xbfO,h\xf1`\xbej\xd5r\xc2\xe1\f\xf2\xf3{\"\xa5\f\x9dp\xc2\xf1\xa3\v\n\nH&\x93R\xd34\xe1M\x90\u035b7\xb3f\xcdZ\x8e:\xea\xa8\x03\xa6;\x0f\x06\x83\xf8\xfd\xfe\xfdz\xb1\xf8|>\f\xc3
\x1a\x8d\xb2}\x1f\u071f\u01c3\xef\xc9\xd7\xed\v\xc8\xc1\xd1rgff\x12\x8dFS\u0296\x83=\xbc\rm\xe7\xce2<\x1b\xe2\xdc\xdc\\:t\xe8\xe0\xc6w\xad\xa3\xa5\r\x87\xda\v\u0465Kg\x84\xa2PWVJ\xb4\xbe\x16\xdd\x1fBS\xc0R@X^\xcc,\x1b9\x11w:\nU
\x82*\xaa.@\x15(\xa6D&m,KbI\xe7\xae+\xe0\xfa\b\b\xac\xa8A\xb4\u06a4\xef\u19f2q\xe9\xb7I+\xf8\xc1md!R\xc9C\xcfS]QT\xb9q\xc3F1o\u07bcqg\x9c>\xe1:\xe0\xe9O?\xf9\xc4L\xc4c\xdd\xee\xbe\xe7\xee{\xdf{\uff49\u02d6-
\xa7\xba\xba\x8aD\"\xd9\xcc\x12\x03\x10{\xaa\xc0\x1a\xbf\xe7\xca(]%\x89m[\u0626\x99\nk\x03\xe9\x19du\xeaNF\x87.\x84\xdbw&\xabcW\xdat\xe8L0\xbb\x1d\xe1\x9cv\xa4g\xb5\u00d7\x9e\x81\x16LCSU\x04`z\u007f\v0l\xe7/\b\x01\x9a\x04\xd3v\xde7E\x80\x94\nR\x82?\x10\"\xd8&\x9b\x9cn\xbd\xe93\xfa84\x01\xbb7\xae\xa5\xf0\x9b\xe9\xac\xfej*\xdbV,\u00b2LG!$mt\xdd\xe76;q\xf6\xbd\x9d;w\x8aE\x8b\xbe\xcb\x0f\x85B\x1f|\xf9\xe5\u0309'\x9dt\xca+\x93&Mb\u04a4I\xbc\xf9\xe6\xeb\\v\xd9\x15\xbf.0?\xf4\u0421\xfc\xfd\xef\xffO\x00\xf2\xf5\xd7_;\xa3\xa4\xa4d\x80\xdbzM\xec\t\xd8k\u05ee\xc30\f\x97\xaf\xfe\xf9u\xe7m\xdb\u6493\x93M}}\xfd^\xfd\x1dM\xd3l:\u0270m\x99\xf2[\xd9W\xc2\xe5\xa7\xf0\xcbR\xcaT\xc5gzz:\xb9\xb9\xb9-
\u6f94\x96\x96R[\xeb\x80y\x87\x0e\x1d\b\x04\xd2~&O\xee\xd6q
\xc0\\U}\xe4\xb6k\x8f\xee\x0f\x10\xad\xa9$\x19i@\xf7\x87\x1c@\x15\x8d\x8a\x1c\xe1\xa4\xfaR`\xeeQ*\x02\x89\xa2*\u0220\x8a\x1dT]\xfd\xb8\xc42m,)Q\xa4\x13\xdd#\xc0\xb6\x05\xf1\xfa\b\xe9m\xba2p\xe48vm\xdf@<\x1eE\xf7;y$\xe1\x05\xd2B\"TE\xc4\xe2q\xeb\x87\xe5+\xb4\xe93\xbe\xb8PJ\xb9\xe0\xaf\xcf<\x93\u007f\xe9e\x97=\xb9l\xd9\x0f\xf9\u06f7ow\x03
\x81\xa2\x88f\x94US\u00f3\x94\x9d\xb4P\x90H\u01f6\xd70\xb0\f#\xb5\x0eC\x19Y\xe4\r<\x9c\x8e\

xfd\x87\u04b6\xd7@\xb2\xbb\xf6!\xbbs>\xc1\xecvh>\x05IG\xb1\xa3\xba4P\u04b21L\x03\xd3H\xa2\xe2\x18\x8dY\xb6\xd3\x1a\u06e3\xd6U\xc5q\x93\x94\x80\"$6`\xd9\xc2\xf5\xa5\x01UU\x1c\x17N\xa9`\xab*y\xfd\x06\xd2e\xe0@\x06\x9fr.\xdf}\xf0\n\x8b\xde{\x89\x86\xaarL\x04B\x98\xf8\xfd>\x84\xb0\xb1m[\xa8\xaa\u02ae]e|\xf5\xd5,TU}y\xc1\x82o\xca\u018c9\xfa\xf3;\xef\xbcC\\v\xd9\x15\xf2W\x17\x99\x1bF\f]\x0fJ\x80\x0f>\xf8\xe0\xf8\xd2\u049d~\x97\xb0V\xf7\x04\xf3/\xbf\xfc\x92\x82\x82\x02\x06\x0f>\xacY\x92\xf2\xe7\x89F\xa1s\xe7.\xe4\xe5ud\u06f6\xed.ol\xed\x15u7\xe5\xd0\xf7\u03c3\xcbf\x1e\x1e\xfb\x02x\u03ec\xa9\xba\xba\xca\xddH\xda\u04af_\xbf\x16s_\xca\xca\xcaR\x1c~^^\xdeO\u069cZ\xc7\xc1%\xceC\x19Y\xa4\x87\xdb\xd0P]I\"\xda@ZN{$\x12U\x11h\x9a\x02\x86t\xacl\xed\xa6\x80\xeeT\x89\xba\x04\x82\u04c2BU\x90>W\x9f(\x14\xa4%\xb1\\6W\xd8N\xa8.5I2\x91\xa0\u03f0\x93\u067af\x11k\xbf\xff\x12[\xf77\x82\xb1\x10N\x85\xa9\x94\xf8\x03\x01e\xe3\x86M\u031e=wP]m\xed\x9b\xef\xbe\xfb\xde\xc0\x82u\uba29\xa9F\xd34\xe9IR\x9a\xae\r'pRP]\xc7Q\x89\xc0\x966\xa6i\x91\x8c\u01f1M\x03\t\xa8\x9aN^\xdfC\xc9\x1fq,]\x0f\x1bC\xdb\u0783\xc8\ua50f/\xa0\x12\x8bK\xa4i`$\xe3\x18qGo.\xdc\xfc/\xd2=uH\xf0\xb9o\x8d)\xdd\xcaW\xf7-
U\x04\xa8R\xa2\xbaI]7\xbcCU\x15\xfc\x9a@W\\\xba\a\x89\xb4-
,\xdb\"n\n\x14\xa0]~w\xc6\xdf\xf6\b\xed{\xf6g\xe6\xd3\u007f\xa4\xa2x\v\x00\xba\xa6\xa3\xfb|\x18F\xe3)\xa4\xa2\xa2\u009c3g\x8e\x16\b\xf8\xff\xec\xea\xf2\xb7\xbc\xf1\xc6k\xea\xe5\x97_i\xfd\xaa\xc0\xfc\x99g\x9eS\x00{\u04e6\xc2\xfe\xe7\x9e{\xc1\xa8\xba\xba:\xf7\xbe\xee\r\x1c\x9b6m\xe6\xb9\x17^\xe6\u017f=\x87\xae\aI&\xa2?\x9b\x8b\x8fm\x1b\xf4\xec\u0643\xee\xdd{\xb0p\xe1w?\xda\xe7\xf1\xc7<\xb0\xf7l^\xe0%D\xf6\x04\xfc=\vP\xbaw\xef\xf63v\xa4\xf9\xd7GEE%\x91H\x14\xbf\xdfO\u06f6\xb9\xad`\xfe+\x18\xa1p\x069\xed\xda\xd3P]I2\x16I\u9f85\x04MQ\xb0UW#-
,\a\xbaE\x93~\xa1\x02W\xaa\xe8&L]\xac\xb7\x14\a\x98\xa5\xe5&B\xa5\x1b\xa1\x9a\x82D<Iz8\x8b\xc1c\u03e6x\xe3\n\xeak+\xd0\xfdi\x8dk\xa71\xb6\x11\x80\x9c5kv\xe8\xeb\xb9s\x06\x16\x15\x15\xa1(\xc2\xd24]\x95\xd2nV\xc0\xe3\x9d~UUs\n\x97\x10\x18\x96\x8d\x91L\x92\x8c\xc7\\zR#\xdc.\x8f\xae\x87\x8d\xa1\xf7\x98S\xc8\x1bx8Y\x9d{\x10\xcaH'\x123I\xc4c$\xa2N\xe5+R\xa4\xac\xd8=\xce\u06f6\xc1\x96\x8e\x15\x81-
!f:\xad\xf5lo3K)r\x1cfQW\x04\x9a\"A(h\xaa@\xf5\uc445\xe7\a\xef\xa6
D\xa3F=Z\x1fE\xf7\x058\xe2\u070b\ted\xf2\xf1#7SQ\xb4\x99\xa4\x91D\xf7\xe9\xf8|>\x12\tG\u07aci\x9a\xb6s\xe7Nk\xf1\xe2%\xbd\x1fz\u807b\x80k.\xbf\xfcJ\xeb\xbb\xef\x16*G\x1c1\xcan\xccd\xb5\xf0Q]]\xad8\x89\x90\xf7G\x00\xfd]
\u072f\x89\xf9\xbb\xef}\xc0\x1b\x1fNs9l\x95\x9f#g\xe8y\xac\xa4\xa7\xb7a\xf8\xf0\xc3\t\x87\xd3\xd9W\xd6|\xcf\xe3\xdf\xdeI\x99\u01afk\x9aJ
\xe0\xdfo\x13\x83=\xc7q\xc7\x1d\v\xa8?[G\x9a\u007f\xf5\x94\xe2\x9d\x18\x02\x81\x00\xd9\xd9\u066d`\xfe+\x18\x19\xe9id\xe7d\x13\x8f4`\xc4b\xa9\xee\x13\xb6\xab\x8f\x16x\xa0\u04f4\xd0\xc7\x13i7\u2ba7\x1a7%\u0616t8\a[b[\xce\x03\v\xa7\xb24i\x13\x8f\xc7\xe9\xd4g\x04\x03\x86\x9f\x04\x12,\xd3H!8M\xdc'\x15U\x15\xbbv\xed\x92E\xc5;,\u007f
UUU\xa5\xb4\xf7\n~\x9c\xa4\xa6\x82iY$\x92\x06\u0446:\"u\xb5$\xe31\x82\xe16t\x194\x82\xe3&\xdd\xcf\xf9\xff\xfb\x01\xe3\xee}\x96\xc1g\\F\xdb\u0783\x90\b\xea\xabj0\xa2\x11t\xe1PF\xaapZ\xe8y\ax[:\\x\xd2r\x94\x99q\x13\x92\x96$fB\xdcr^\xa6)%\x86\xed\xbeg\xd2\x01y\x84@SU\xfc\x9a\x82\xae:\xe5\xff4\xed\xbe\x97z\xfb\x9a`\x84\xa2`$\xe3$\xa3\x16\x83O\x19\xcf\xe9w\xfc\x89\xb4\xcc\x1c\x12\x89\x04\xa6i\x11\b\x04S\x9e\xfa\xae\xeaE\u0670a\x03\xeb\xd6\x15\\=m\u06a7\xe3\x01\xde}\xf7]\xf5W\x13\x99K)\xc9\xce\u0392\x00\x05\x05\x05=\u0768\xdc\xdc\xf3\xba}\x814\xba\xf4\x1dJy\xf1F\xea\xaav\xf1\xe4\x13\xffM\x87\xae\x87r\u0310nhD\u070d\xff_\xbf\x16\x80\xf1\xe3\xc7\xf1\x

e1\x87\x1f\xf1\xed\xb7\v\u007f2\xf0y\u050b\xa3+u\"n\xa7\x11\xf0O\xe3\xf5{\xf5\xea\x95j\xc3\xd6R
\xda\xc8y\x96\xb7~\xbf\x8f6m\u06b4\"eK^G\x0e$\x90\x95\x1e
'3\x83u\x898f2\xe9\x02\x92K\r\x02\xaap|Yl\x84\xc3k\x8b\xa6
\u07ac\xa0\x11\xcbv\xc0\\\x91\xa0\x9a\xeeF\xe0VW\xdaB\xba\xa9T\x81\x11K\xe2\xcbl\u00e1c\u0
3e1h\xc32\xcavlD\xa2\xa7:\x86\xa6*)m\x89\xaaiB\xd3T\xd59\xd9:rB\xa7\x01\xb5H\x81\xa1\xed6\x
c70\xe21\xa4\x94\x84\xdad\x91\x99\u05ddN\x83F\xd0k\u0329t\x188\x9c\xb4\xdc\x0e\xe8>\x05\u
06c4d<\x86i\x99h\x02\x14UC\xb8{\x8d\a\u0216\x14\xa9\x86\x1b6{tW\xf2\xf6\x1c/\xf7@\xa3$S\xc
1\x89\xb4}\x9a\x82\xae)(\n\xa8B\xa04\xba\xffz\xf9`\x8fU\x02!S\xf4\x8cpv(,\xcb\xc02T\x86\x9dy6\x
b5\xbbJ\xf8\xec\u007f\xee!\x1a\x8d\xd0&\xb3\r\xd9\xd99TVVx\xa6_\xc24-
\xb9j\xd5*\xb1d\u0272{\xa5\x94\x8b\x85\x10\x15R\u06a9\"\xa4\x16\x1d\x99\u007f\xfa\xe9\x14\xa
5\xba\xba\u0192R\x86jjj\xfa\xef\u06b5\x1b!\x84\xb2g\x04\x98\u06f97\u01dcw7\xdd\x06\x8c\x06\x
a0p\xd9<\x9e\xfe\xf3\xb3,^\x1f'\x12W\x11?\x13\xd9\xe2y^\x9fv\u0684\xbd\x1aR\xecI\x914}4MDy\x
91\xb5cw\x9blf\u04b4\xbfq\xe1\x85\xe7\u04ed[\x0f\xa4<\xf8\x8e\x89\xceuJb\xb1\x98{\xc2\u041a\x
a8lZ#\xf3\x96\x1c\x18\xa5\xf9u2BA\f\xd3\xc42\x12\x0e\x9f\x90j\v'Q\\5Fc\u0562\xdb\xfc9\u0144\b\
xb7\u0273\xc0\xb4\x1d\xf0p\u0336\x04\xc2rM\xba\xa4\x9b\x10t\x1bA[\x96E<\x16!\xb3c>\x83\u01
dc\x89\xae\xfb\x91\xaeX=\x15d\xb9\n)e\x8f\xc6\x16B\xa8(Bi\x8c\x82m\x9bd<\x8a\x11\x8f\x92\u06
ed7\x87\x8c\xbb\x80\xa3o\xf9\x13g>\xf5>\xc7\xfd\xfeY\xba\x1e}\x06zN\x1e\r\t\x93\u02aa\x06j\xea
\x1a\x88&-
\x926\xc4m\x88\x9a\x92\xa8%\xa9KJ\"\x06DM\a\xd4\rK\x12\xb7\x9c\xa8\u0734\x1b#pO\xa9#\xf6\
x00h[\x82\xa6\b\xd2|*A\u0761V\xbc\u04cc\xe2n\x8a\xba\x02~\x15|\xaa@\xf5z\xab\xcaF\xa32p\xe
8\x18\x80D,\x81\xb4a\xf4\x05\x13\x19~\xfaE\x98\xa6IMM-
\xa1p\x06\x99Y\xd9)\xdf\x1f!\x84\u0639\xb3\x8c\x82\x82\x82#\xdfyg\xf2a\xce\xfb\xa4\xfc:\"\xf3\xe
5\xcbW*\x80=o\xde\xdc\x1e\xb1Xl\x90k;\xbb\x97\x8a%\xaf\xe7at\xedw4\u0246\x04E\x05\x8b\xa9\
xad,a\xfe\x8c\xb7\xe8\xdas(\x81K/\xe0\x90Nq\xfc>\xf8W\xf3\xa1\x8e\xf9\x95\x8fK/\xbd\x94\xb9s\x
bf\xe1\x8b/\xbe\xd8o\x04\xbf'\x00\xee\xcb\xe3\xfa\xa7P\x13\xf9\xf9\xf9L\x9cx\x15\xc0A\xf72W\x0
0MuZ\x8ey\x8bN\xd5tTM\a\x1c\x1d/\x12Z{\xf6\xb6H4O\xf1\u0376ma$\x13H\x17\xccE\x13\xe5\x8
a\xf7\u007f\xa9\xb8\x06Y\xd2u\au\x13|R8`g\xb8\xe0+\\\x13-
'd\x95\xa9\xed\xdcvs\xa9H\x81\x15Obj\x1a\xf9\x83\x8f\xa1\xeb\xcayl.\xf8\u07b1\xc8udx.\xed\xe0l\
x14\x9ew\x8b\xf7\x90\xb6DZ\x166\xa0\xfb|t\x1c|$\x9d\x87\x1dC\x8fQ'\xd3a\xe0p\xec@\b\u06f4\x
89\x18&v<\x8a\xb4\xadF{^\x9a\xf2\xd4\xce?V\x93\x90Cu\xbfg7\tEll\xea5y*\x1eM\b\xcc&\xbf\u03ef
)\xa4\xfb\x14|\xeeRHu\xd6\x13\x12\v\xc7\xda\xc0k\xab\x8at*m=\x9e\xddF\xa04v\xe2\u00d3P\xc6\
x1a\x12\x842\x82\x1c}\xe9Ml_\xb5\x84\x1d\x05+\xa8\xaa\xa9\xa1mN\x0e\xd1H\x84\x86\x86:\xc0
Q\u022d\\\xb9\x8a\xbe}\xfb\x9c%\xa5\x9c#\x840\x17-
Z\xc0\x91G\x8ei\u0651yeE\x85\x020o\u0782n\x96e\xf5\xb5\x9cb\x85f\t\x11\xdd\x1f\xa0]\xb7CAjt\
xe9=\x8a!G_\x8a\xa6\xfb\x89\xd6\xedf\xda;\u007ff\u019c%\x945\x04\x11H\u051f\xe1\u055af\x82\
x8e\x1d;q\xdbm\xb7\u042d[\xd7\x1f\x8d`\xf7\x17m\xef\xeb{\xfb\x03\xf5\a\x1f\xbc\x9f.]\xbac\xdb\xc
6A\x8b\xca\x15G^\x8c)\x14j\x92\xb0`\xe9\x0fl\u073c\u074d\xccu\x84\x10\x18\x86\x81\x914S\xfe\x
ed\xad\xa3\x05R-B
\x14\x05i\xdbX\xa6\xe7\x18\xe8\x16\xd2\xe0&\u294dg'%\x84\x82PDc\xbb;\xe1D\xe5\x86\xddX\\$\
xbc\x877\xb7S2u\x99\xda\x04\xa4\r\xf1\xfa\b\xe1\u030e\f\x18y\n\x81`h\x9f\xa78\x8fJ\xf1|Ql\xdb\u
00b6LT]\xa7\xeb\xe1c9\xe6\xd6\xff\xe6\xa4\xfb_\xe5\xc8I\x8f\xd2\xf6\xb0\xa3\x89K\x95XC\x84x4
J\"\x9e aX$-

\x97\xdep\x81\u0652\x8d\x11\xb7\xe5\tt\x9c\u032esZv\x9fc\x81K\xb9x\xd7@\ua512p\u007f^U\x04!\x9fJ\u062f\xa0\xb9t\x94\x94\"\x15\xad{\xf9\x84\x14\xf7\xeeF\xf86\x8d\a\x10\xd1d\xb3\xf3\"\u007f\xe7\xff\x92h}\x9c.\x87\x1c\u00a8\xf3~\x8bP5\xca\xcb\xcaPT\x95\xac\xac,TEK\x9dNjkk)))\xbdp\xf6\xec/\xdb\x01l\u06f6]\xb4x\x9a\xe5\x99g\x9f\xb3\x00\xca\xcav\xf6/\u06f5\xcba\u063c\x12iU\x03\x04\xd9y\xbd\xc9\xc9\xebC<R\x8fe\xd9\f\x19s1\xbd\x06\x1d\x8f\x10P\xbam9\x1f\xbd\xf9\x14s\x96l\xa7\xc1\x0e\xa1i\xe2_f\u03dd\xe6\x04&'\x9dt\n\xf7\xdcs\xf7\x8f\xf2\xc5?\x06\xe8?\xc5t\u9847\x1e\xe4\x92K.\xe1@v\xa4\xf9)@\xae\x00\xe5q\u061c\b\xf0\xf1\xbc\x15\xdc}\xef\xfd\xacX\xb1\x12P\x11\x8a\xea\x1e\xad\x1d\xe9\xd5\xffg\xef\xbc\xc3\xec\xba\xcas\xff[k\xed}\xca\xf4\xa6Q\x1b\x8dz\xef\xb2\xe4^\xe4&\x9a\xc16\x98bS\x13'\xb9$@n\x02\x81\x9bK\x80`r\x93Pb_\u01d8N\x8c1\xc1\xa6\xd8\xc6\xd8`\x1b\xdce\xc9Un\xea\xbd[3\x92\xa6\x97SvY\xeb\xfe\xb1\xf6\xdeg\x9f\x11\xe4I\x00\x97\x9bG\xdb\xcfQ\x9b\xf1\u0319s\xd6\xfe\u05b7\xde\xf7\xfd\xde7\x88\f\xfd\xd3\x0e\x82'\xaf\xd7\xcf%bpCZ=\xb6\xd1\xf1\xa3\x1a(\x93Q\xf51\xc8\x04vAT\nc\xf2o&B\xbeM\n_\x8f\xc6\xe5u<Q\x1a\x82\t5\x9e\xa7\x99<\xe74\xa6\xcd]I\x18\xd9b\xa4=k\x84\x89C\xb25~\xa9\x88\x92\x92I\xf3Wp\xdeG\xff\x915\u007f\xff]\x96]\xf51\x1a\xa6\u03a30:\xc2\xe8\xd0\x10^$\xdf\xd3)\xbc;\xfd3hS\xe9\xfc\xd3D\xa4\x88Z\xc2@\u06df'\x96
\xc6\xcf;L\x15\xda\x18v\xc9(IcVQ\xe7\xcadZ\xb6\xea\xebF_#\xd4\t\x1f\\\xb5\xa9\x98\xa4&\x98\n\xff`H^w\x81\x88F\xfca\u025a\xb73\xf7\xf4\xf3)\x15F9v\xf4\x18\xb5uu\xe4\xf2\x15\xaf\xa7R\xa9\u012e]\xbb\x1a\x8e\x1e=\xfa&\v\u00fe\u05fc\xae\x8b\xf9\x97\xaf\xbdV\x00\xa11&\u007f\xe4\u0211\xb3b\xb7\xc2\xf8RN\x06!\x04\xad\x9d\xf3i\x9c8\x13\xaf8\x8a_.R\xd78\x8e3\xde\xf0?h\xef\x98\aF\xb3\xe3\xf9\xfb\xb9\xf5\xa6\xaf\xf2\xd8\xe6>|\x91'\xe3\xfe\xfe\x05\xdd\x16\u0590\xab\xaf\xfec\xae\xb9\xe6s\xbf5\xc9=\u059d\xff&<=)\x96\xd1b\x1e[\xd8?\xf3\x99O\xf3w\u007f\xf7\xbf\xd1:\xa4T*\xbc&\xa3\xfcR\x80#\xa0\xab(xn\xc0\xe5\u0387\x9f\xe1k_\xfeg\xb6n\u0708\x9b\xab\x01\xa5\b4\x14\xfc\x900\xb4\xb8k\x18j\x82P\x13\x84!\xfad\x87\xfe\xbak\u03f56\b)m\x94\xdb\x18\x9e#5\xf4\x99\xae>VK.\x04\xa1\x11\x94C\x13\r\xc8\u0204(\x14X\\8^\xa2\xb1ml\x8c\x9b\xdb.^R\x1e-P\xd78\x89\x99\x8bW\x93\xc9\xe6\xd1:@D.\x82\"ZpZ\a\x94\x8b#\xd45\xb6p\xe6{?\xca\x1b?\xf3u\x16\xbd\xfd\xc34L\x9a\xce\xc8\xd0\x10##C\x16\u0089\xbeYhD\x02\x9d\xa4\xbb^\x13\xad\xdfT\x13\x9e\xa8s\u049d\xb1I?\xe2\r
\xfa=\x88\xb2Ok3\x92\xa6\xbc$\xe7V\xfb\x99W\u03caT:}\x13\xd1\v\"\xe9\xf1I\xac\x12td\x8f\x9bl>1m\x10\xe1\u07e5B\x89\x96)\x93Xu\xe9{\xc9\xd55p\xac\xbb\x8b\xd1\u0451*\xe5[\x18\x06\x1c=z\x8cm\u06f6\xaf\x8c\xbf\xff\x81\x03{_\x9f\xc5\xdc\x18\u00e7>\xf1\t\x03\xf0w\x9f\xfe\u07e7\xbd|\xa4\xfb\xb2\x91\xe1\xca\b\xbd\x90\x12\xa4$W\xdf\xc8\u0139\u02e9ik'\b\x03\xb4\x0e\xf1J\xa3L\x99u*\xe7^\xf2\t\x9a\u06a6\x10\x06\x05\x9e\xb8\xef&\xbe\xf1\xb5\x1byd\xe7\b\xa3\xa1\x83#\x19\xb3\x8c\xff\xebW\xa9TB)\xc5\xff\xfc\x9f\x1f\xe7{\u07fb\x89\u014b\x17Uu\xe4\xe9\"\x1dc\x95\xe9\xa9Q\x1b\u079c!\x93\xc9\xe0\xbanR\xe0\xdb\xda\u06b8\xf1\u01af\xf2\x0f\xff\xf0\x8fH\u9f22.\x90\xff\x99b\xdeW\xd2<?\xe8p\xdfcO\xf0\xed\xcf\xfc%\xcf?x\x173O=\x8f\xa9KO\x87\xd0\xc7\vBzF<\u02a1N4\xf3\xf1\xe3dw\xfe:\xea\u0205\xb0\xce\u007fA\x80T\x0e\x8e\x9b\xb1\x1dfU'k\x12\xfd\xb8\x89\xd4(\xb1\xaf7\b\x0e\xd1)\xb8\x00\x00
\x00IDAT\xbc\x10\xbc0]\xa9l\x9foa\v\x91H\x1b\xd3V)\u0698\n!\x1a\x1a\xfc\x92\u03d4\u06671a\xdaB|\xaf\x94$\xf9H\xa5\xec`\x8d\xef3e\xder\xde\xfa\u026fp\xea\x87>E\xe3\xf4Ex\xe5\x12\xa3\x03}\xd6O\x1dY\xe9\x9c\xe3\xcd\xc4T\to*\x8a\x14D\x82g\xc7\x18\xb5I\xc9\x10\xc7v\xec&\xc5\x1b\x84\x91\xea\xa4!\xa7h\xcaI2R$|\x90H\x1a1\x128FF\x9b\x99\x92\x15,\x9eH\xd3c\"\xf9\xa7\x8eI\xe4D\xdeY\xed\x83\x1ek{\x02\x0f\x96\xbd\xf1\n\u67f3\x86r\xb1@oOo\xd5ty\x18\x86\x94J%\x8e\x1d;\xf

e\x16cL-
\xc0\u05293^\x9f\xc5<%\xe9k\xda\xf7r\u05e7\xb6\xed\u0611>'\"\xa4\x83\t\x02\x9a&Lg\xe2\xdceh\x13\xa0\x85u?\vB\x8fR\xa9\xc0\xac\xa5o\xe0\xd4\v\xff\x84\x9a\xbaf\xfc\xf2\x00\x8f\xde\xfe5n\xfa\xee-
<q0d0*\xe8R\xfc\xee\x05]\b\x81um\f\xb9\xf2\u02ab\xb8\xed\xb6[\xf9\xc8G>\xc2\xf4\xe9\xd3S]9d\xb3Y\x9a\x9a\x1aijj\xa2\xbe\xbe.\xc9\xf6\xcc\xe5r\x94\xcbeJ\xa5\x12\x9e\u7854\xe2\roX\xc3]w\xdd\xc9G>\xf2\xd1h\xc3(\xbcf\x8e{BX\xc2wg1\xc3\xd3[\xf7q\xff\xb7\xbe\xcc\u02db\x9f\xa6\xa1m\"\x8b/y?\x8d\x1d3-
\xfc\xe3\xfb\x1c\x1f\x1a\xa5\x14D\x04X\xea\x11\x17\xf5\x93\xd7\xeb\xe3\x9e\xf2<\x9fR\xa9\x84\xe3\xba(\u05ed(G\xd2
\x8c\xf9\xcdx\xbb\x01J\x91\x17K\xacp\x11\xd1\u007f:\xe9ze\xa5\x80\xa5\xfd\xcbM\xe4}n\x04\xe5B\x89\x9a\x86\x89L\x9fw\x16\xd9\\\xad\xf5K1\x10\xfa>Fk\xe6\x9d\xfd\x06\xde\xf6\xa9\u007fa\xd6y\x97\xe0\x87\x9a\xc2@\x0f~\xb9\x80W\xf6\b\xcbeL\u08c3\xc0\x8e\xe9\xfb\x1e\xc6\xf70\x81\x1f=\x02\xc2
\xb0|@h\x1f\xc6Da\x1b:\xd2\xc2\xc7\xda\xf0\xdf\xc2+\xc4\x1d\xb7#\x05\rYI]F\xe0\xa4D7\xf1C&\x9dw\xf5C2\xb6#\x8f\xcf?qa\xaf\x04ZXX&\x9e4\x15\xc9\xe6\xe9{\x1e5-
\xb5,\xb9\xe82\xeaZ\xda\x18\x1a\x1c\xc0\xf7\x03\\\xd7M\x92\x95FGG9x\xf0\xe0\x94M\x9b^<#~\xfe\xaf;5\xcb\xc3[\xf6\x8b\v\x16N3\x00\xff\xfa\x9d\xef]\xbde\xcb\xd67\rG\x86N\x90J\x06\x91\x92q\xd3\x17\xd1\u06b9\x800(\x81cU\"\"\x14\xf8\xbe\x87\x11\x92\xc5g\xbd\x87\xc2H\x1f\xcf=z3\u0151\xe3\xfc\xea\xe6/\xe3\xe6\xf2\x94>x\x15+&\xb8\x8cw}\xa4\xb0\xbb\xf0\xefz\x83\x94\xcbe\\\xd7e\xe1\xc2\xc5\xdcp\xc3\xf5\xbc\xfd\xed\x97r\xc7\x1dw\xb2a\xc3s\xec\u07bd\x87\xa1\xa1AFF\xc2\u0102\xd3J\x11\xad\xe6<\xf6[\x99?\u007f\x1e\xef~\xf7\xbb\xb8\xf2\xca\xf7\x90\xc9\xe41&xM;r\x00\x13\x84\f\x84.On;\xc0/\xbfs\x1d\a\x9e}\x84\x9a\xa6q\x9c\xf1\xa1O\u04be\xf4lv<\xf5\bB(|\u07e7o`\x18?4d\x18sl\u0546\x93H\xcb\xeb\u5494JE\x86\x87\x87\xc9\xe4\xf2H\xc7M\x06\x85\xa2\x066\x1a\x0eJK\x11\xa3\xc6\u0280\xa7\r^\x84K\x88\x04\xaf0\x952&Rmm\xbcE\xa4\x8c\xa9t\xd4\xf6*\t\x81\x1f0g\xe9E\x1c\xd8\xf1\x14{\xb7=\x89T\x0e\x99\\\r\v/x\x1bg\xbc\xe7\xcfh\x1c\xdf\xc1\xc0\xf0\b\x81\xd6\bc\xbd^\xb41\xd6*
\xf4#\xff\x15\"\xf5L\xb4\xa5\b\xfbg\x1d\x85B\x04\xa4\xa2\xdd\xe2\u009a\xb4\u0572B\x06G?_<\xd9i\x8c\xc1U\x82\u01ac\xa4&##\xe3\xac4\x142\x16\x86\x8a\xbfG\xdco\x8b\xb1\x1fN\xe1\xf9\x95\xff9\xdeFu\xca\xda#\xde\n\x851\xf8e\x98s\xe6E\xb4\u03d8\u01de\r\xebl\u4723\x92\xc2S\xf6\xcatww\xf3\xd4SO/\x06\x1e|\xdd\x15\xf3\rG\x06Y9\xa9\xd1\x00\xf4\x1as\xee\a\xaf\xfc\xe0\xff\xdal\x13\xae\x13xEHE\xe8{\u050d\x1f\u03d4%\xe7\xe1d\xea\t\x87\aQ\nK\xb4hkG\xe9{%2\xd9ZN\xb9\xe0j\u02a5a6=\xf9\x13\x06{\x0e\xf2\x8bo}\x01\xad4\x85\xf7\xbf\x8f\xa5\xad\x19:\xb3\x1eY\x19a]\xbf\xe3\xf3\xb6\x1e\xe7v\xe7\xbc\xe0\x82\x8bY\xbdz5\x9b6mb\xe3\xc6M\xec\u0673\x87#G\xba\x18\x1c\x1c\x8cB\x9a\xb3\xb4\xb5\xb5\xd1\xd99\x85\x8e\x8e\x0ef\u0318\u03aaU\xab\x90\xd2\xc5\xfa`\x17-
\xae\xf9\x1a\x16\xf20\b\xd02\xc3\u04fb^\xe6'_\xfb\x17\xb6\xfc\xf2\x87\u0535\x8c\xe3\xd4\x0f|\x92Yk\xdeE\xc1\vP5\r\xb8\xf9:\xfcr\x99\x91\xc1~\xeb\xe1Aug\x1e\xc7n\x9d\xbc^\x1f\xd4\xe7\xf0\xf0\b=\xbd\xbd\xe4\xeb\x9a\xc8ds\xb6\u0324\x8a\x95I\xad9\x19\xfd]G\x05\xa7\x14X\u065d\x13O2#\x11\xd1{l\x03'D<\u0493*\x9e&E\xbbF\xe3\xf0\xc6\xe0\x97\xcb\xd45Mf\xde)o\xa1\xeb\xe0V\x86\xfb\xba\x19?u>K\xdft%\xf9\xb6\xc9\f\x0f\x0e\xa0\x84
@F\x10\x8d\xd5\xc0\xa3\xed0\x91\x91\xca\x16\xed\u0529!\xf6d'5\xa1j\xf9\u0644\xf2M\xac\tb\x1e\

xc0\xaau\xe2\xc2n\xfb\ua332\x1dy\x8d##7\x9aJ\x017T|\x1f\x05\x95\x13J\xfa\xa3\x95Bn\xaa\x9eK
\xe5\xb3L\x15$\x14\xff9q\x1e\x8e\x88P\x11h\x9a&\x8cg\xea\x92S\xd9\xf7\xfc\x13\x14K%\xb2\xb9
\xbc\xb5\xda\xd5Vd\x10\x84!\u01cf\x1f?\xfdu\u05d9\xef\xe9\x1fefsm\f\xaf\x9c\xfb\u037b\x1e\xfc\xf
1\x86'\u058d\u04e1\x9f\x9cq\x84T\xc9\xca\x1b?\xfb\x14\xa6,:\x87\xb0XB\x87\x1aGY\xb3\x04-
L\xf2\x92\x95K\xa3dk\x9a8m\xcdG\b\xfc2\x9b\x9f\xbe\x83\x81c\xfb\xf9\xc5\u05ef\xc1\u01e7p\xe5\
x87\xe8k\xc91'W\xa45+\x12,\xeew\xc5\xf9-
1j!\x93\xa5KW\xb0t\xe9\n\x00FF\x06\x19\x19\x19\x89\xc6r]\x1a\x1b\x1b\xc8\xe5*CGa\u8f6a\u065
0\xff\xd1\x15\x04!\x197\u02c6]\x87\xf8\xfa\xf5\xd7\xf3\xdc=\xb7\x92\xcd\u05f0\xec=\x1fc\xe6EW\x
e0\xf9!\u0294\xc856\xe3\xe6k)\x0e\xf7S\xec?NV\x1a\x1b\xfe\xabu\x82\xb1J%\xff\xdbD\xc8\xfd\xff]
\xc6\xed\xd5;8\xcc\xf1\x9e~\x1a\xa7\xce%\x93\xaf\x89\x94 i(
E\xeaI\u06f9\n$A\b~d0\"\x84\xed\x85e\xba+\xaf\xa0\xc8X%u\x1c\x9cV\t\x84H
\x17\xec\x1a)\x97Kt\xce=\x8b\xa9s\u05f3\xfd\xf9\xfb\x18:~\x84\x83[^$\xdf1\x93\x92\xafq\xa4\xed\
xae\x8d\x01\x95\x10\x93\xd2B$&LHY\x11\xdd{!V\xab\x9eH$\x11\xa9gd\x12\xfc<)\xbcFG\xa7\tk\u
022e5()\u027b.9iU'\bQ\u054d'\xaf\x8d\x11U\xc4\xe5\xd8\x12\x9d*\xf7c\u0781\n\x8e.\x10U\x9b\x8
3\x89\xbe\xae\x10V\x11#\x02\x8d\xceH\xa6.YE\xbe\xbe\x89rq\x04'\x93\x8dN%!Zkc\xb4\x16CCC\
xf3_W\xc5\u0718\x12k\x9e\xc9\xc6Eq\xfc\xbac\xfe\r\xb7\xdf\xf6\x83\t\xdd\a\xf7V\x93\x9e\x80\x0e\
x03jZ\xc73\xef\r\xef%\xdf\u070e\xd7\xdfoM~\"\x12\"\x99B\x8e\xce\xfa\xa5\xe20\xf9\xfaq\x9c\xf1\x
e6\xbf\"\fC\xb6m\xf89C\xc7\x0eq\xff\x8d\xff@qh\x90\v\xde\xf7aF&7\xb2\u0414\x98\x90\xd5\xd6\x
dfY\xff~\xe4\xa8\x1d\xd7\x1fM2@\xf3\xf9<\xb5\xb5\xb5\xc9\xc0\x90%0*!\x15\xaf\x97\x82\xa7\xb5!
\x93\u0272i\xdb.\xbe\xf4\xa5\xebx\xf4\xee;\xa9o\x19\xc7\xe2w\xfe\x053/~\x0f\x81\xd6\x04\xbe\x8
7P\x92l}+\x99\xda:F{\xbb\x18\xe9=\x8aB'\xa4T\x95N\xf9\xb7$'\x9d\xbc^E\x80%:\x1d\x1d:\xd6\xcf\
xf0\xc80\x1d\xcd-dr\xf9\xc8\xe7;\x9ar\x8c\x8b\xad\x88\x8ap\u0535J\xa1\bBA \x05J\tT
Q\x0e`4\"\xf0\xa3\xe2])H\xf1z\xb6\xfauRU\xb0\xe2\x96\x1e\x1a\xf0}\x8fL\xae\x81\x05\xa7_\u0391\
x83\x1b\x19\xe8\xda\u03ce\xc7\xef\xa5}\xd1*r\xed\x93\x18-
\x160\xd2>w\x19\r\xed\x84F\x83\x88\x02'\xa4B\xb9\x0eJ\xaa\b*1Ud\xa7-
\u4552jL\x94\xc7\x19oB\xe9IS\xc0\bI\xc6U\xe4\\\x85\x92\xd2nV\xc6$Dp\f\xef\xc2\u0621\xb8\x13;\
xed\xb1\x9dz\xa5\xc0\x9b\x04\xd3\x1a\x13\xa7Q\xe9\xe7\xa3\xc1+\xad\xed\xfd4n\xfa<\xf2\xf5\x8
d\x14\x87\xfa\xad\x8b\xa2\x90\x84\b\x84T\xa2X*q\xe0\xc0\xc1\xe2\ub998\x1f\x19\x1a\u5fae\x8c|\
xe04\xa1\x8d1\xe2%\xc3\xf5?\xf9\xe1\x0f\x97>t\xfb\x0f\x13\xcf{!\x95\xf5i\xd0\xf6X8\xed\u07372\x
e1\xac5\x94J\xc2\xfe`\bth\xb1<\x19\xb9\x02\x89T\x01)\x95\x86\xc97\x8c\xe7\xccK\xfe\x1a'\x93e\x
cbS\xb73\xdcs\x84\a\xbf\xf3E\x06{\xba\x19\xfd\xd3O\xe0\xcd\xef`\x8e\xf6\x99\x9a\xf3\xad\xe5\xe
5\xefY\xd0\xd3E\xda\xf7\xfd\x13,q_OE\xdcz/;\xf8\x81\u6a67\x9f\u57fer\x1d\x0f?\xf0
\xcd\x13\xa7\xb0\xfc\u028f\xd1q\xe6\x9b\xd1BP.\x95PB`\xb4&\xd7\xd4F\xa6\xa6\x1e0\f\xf6\xf5R(\
x14\xc8fs\x95\xee$\xf2\xa989\xe2\xff\xdaw\xe5J)\x86\xbc\x80\x03]\xc7\b|\x9f\xda\xc6\x16T&G\x1
0\x84)\x89\\\x05K0\xc9H\xbfD\v\x89'\x04\xc2u\x11J\xa2\x82\x10\x95q\x80\x00\x11D\xbai\x01&\x1
2L\xeb\x04\xfb\x15\t\rZY\xe7\x95\x13\x80\x1fj\xf0<\xc6MY\xc8\xf4E\u7c79\xff(G\xb7?\u03c1\rk\x9
9\u007f\xc9{-\x84a\f\xa1\x16\x84R
\x94\x83\xe3f\x11\xcaE\x1bM\xe8{\x04\xa5\x12^\xe0\x11ze\x82\xe2\bAq\xd4\x12\xa0~\x19\xed\x9
51:@\x87\x9a0\b\x18\x1b\x9cb\xa2\rF)\a\x95\xc9\xe2\xd4\xd6S\xdf\u0502jmA\xd7\u0510\xc9\xe5
psy\x9cl\xce\x16\xd70\xc0\u8832Y\xa5\x02\x9f\xd3K\xbc\xba\xdb6\x15\xc8)!\x98+\xffF\x15\x1cS\x
bd!\xc6\xde0u\ud4e8il\xa6\xf7\xf0>\xb4\xd6(\xa9\x92S\x8e\xe7\a\f\f\x0e\x8a\xd7M1\u007f\xf28\xe
2\x1d3\xa56\xc1\u05dc\x8d%n\xb9\xef\xd7O\xbe\xe7\u07ef\xfb\x02\xc4QSR&]\xb91\x9a\u018eY\

xcc~\xcb\x1f!\x9b\x1b(\xf4\x0f\xa2\xea2\xa8Q{,\x11\x84
t\x12\x1b\x15\xbfa\x00\xe5\xf2(\xb5M\x139\xe3\xd2O\x92ije\xe3#\xb7P\x18\xe8a\xfd\x0f\xbf\u03b
1\xfd{\xe8\xfb\xcb\xcf1\xb0r%\xbdy\x98\x93/\u04d4\x93\x04z\xec\xfe\xfa\xfb\x17\xf6\xd7\xd5\x15u\
x01n&\xcb\xe8h\x81\xbb\xef\xfe%\xff\xf7\x86\x1by\xe1\u0157\u8637\x843\xde\xffW\xb4/_\xcdh\xa
9L\xe0\x95\x93\x13\x8f\u059a\x9a\xd6vj\x1a\x9a\x018\xde\xd3C\u05d1nf\u035a\x811\x12\xc7Qd\\
'\xf2\x948YP_\xebj.\xa4\xa2o`\x80\x03{\xf7`\xb4\xa6\xb1}\x12nM=\x85b1\x9a\xb4\xa4\xe2\xfd\x1f\
xfb\xa4
\"\xfbWM9\x10\x11\x84\x11!\xe2R!\x1d\x89p\x01\xcfD\u49b4\xddx\nX\xf9M+>\u0351\x06\x81G\xb
6\xae\x819+\xde\xcc\xcb;\x9f\xe5\xf8\xa1m\xec]w/\x13\x16\x9dJ\xe3\x94\xe9\x84a\x80v\xf3\xb6\x
93/\x15\xf0\a\xfb(\r\xf53r\xfc0\x83\x87\xf70\xdau\x90B\xcf\x11\xcaC\xfd\x04^\x89\xd0\xf7\xac\xef\x
8b\x0e\xaa\xa6[#l&u?Vj\x83\x88\x88P\xe1\xb88\xd9<\xf9\xfaF\x1a\xc6M\xa0\xa5c:\xadSg3n\xfa|\x
ea\xdb'\x90ol!_\u05c0\t\x03|\xaf\fJ\"\x1d\x17\x19\x9f\f\xc6\xf4\xdfT\x9dE*8\x8d\x10\xa2\xaa\xa6$\x
f8z\xa2\x8d\xafl\x05Z\x83\x93\xaf\xa1\xa6\xb95B%Bdd\xa7!\x84\xf5\x8b\x17B4\x19c:\x84\x10\x87
_\xd3b~\u02ce\x11\U0004e675\x06`\xb3\xf7\x17\xff\xf8\u0736\xc3W\xde\xfc\x95\xcf\xd3\u007fx\x
9fA\b)b\x9c<\xfa\x81\xa5\x94L;\xf72Zf/\xc5/\x87h)\xd0\rY\x8c\x12\xa8\x11\x1f\xe1\x03\xa1\xed\b\x
8d\xd4\x15\x1aY\xc4\u064d\x05\u071a&V\xbc\xf5\xa3\xe4Z\xdby\xe9\x17\xdfb\xa0{?;\xd6\xfe\x82\
xa1\xe3/s\xf4C\u007f\xcd\xe9o\xbe\x9c\xd1\xf6Z\xe6\x86\x05&\xe4\xad\xd7q\xf0\xdfP]g\x8cA)\x85
\xeb\xe68t\xf8\x10\xb7\xfc\xe0\x87|\xf3\xdb\xff\u0191\xaen\xe6\x9c~>g\xbc\xf7c\xb4\xcf[\xc1\xf0h\
x01\x1d\xf8\xa9\xde\xc1\xfe\x9aoh\xa2\xb6\xb5\xddb\xb1\xbd\xbd\x1c>|\x98\x85\v\u7864$\x93\xb
1\x99\x86\xe2d_\xfez\xe9\xcd\x19\xea\xef\xe3\xe0\xee]\xb8n\x96\xfa\xb6v\xa4\x9b%\x18\x19E\vI\
xd97\x14\xfd\x10)\x05\x19G\xa2\x94\x81\xc0\xe0\xa1)\x04\x10b\x13\xe5\x851\xc8x\xbc\x11cG\xc
c\x1d\xd7f|\x1a\x1dy\xa3\xc70\x86I\x8a\u007f|*\x8d\x15(\xb1\x02\xc4\x18\b\xca%\x9a\xdbg2k\xe9
\xc5\f\x1c;\xc0\xb1]\x1b9\xf2\xecZ:f\xcdg\xa80L\xcf\u07ad\f\x1c9\xc0\xe0\xa1]\xf4\xef\xdf\xce\xd0\
xe1=\x94\x86\xfa\b\xcb\x05\x82r\t\xed\x97O \x19S~\x1fv[\x12\x95M\xaa\xfa\x95\x89\xe0
\x1dF\xb8\xba\xfd\x8c.\xa1P\x99\x1cN&\x1b\u017e\xcdb\xf2\xc2S\x98\xbc\xf0\x14\xc6\xcfZ@\xe3\
xf8\xc9\xd6\u07e6P\xb0A\u03ae\x83P\u059a7\u0749\x8b\xb1P\xcc\tR\xd0\x18\u007f7i\xda8\xf9\fk
\xaf\xebP\xd3\xd8\x12y\xcfW\xc3\x05e\xcf#\x9f\u02f5\x06~i\t\xf0\xda\x14\U000d738d0\xec#\u039
e\\g\x00^\x1a6\u007f\xbd\xad\xab\xf8\xa9[\xbez\x1d;\xd6\xfd\xda\xc4o\x03\x91#\x98\x10\x02\x1d\
xf84\xcfX\u0234\x8b\xae@\xe5k\xf1\v\xc3\b#0J\xe1\xd7f\xd0\x02\x9c\x11\x10:\xb4\x93V\x91-
g\xa5\x15\x10\x18!(\xfb\x05d6\u01e2\v?Dm\xcb$\x9e\xff\xf9W9\xb6\xfbE\xba\xb6=\xcf\u03ff\xf4I\x
8el{\x91s\xdes5\xbd\v\xe617\xf4\x99]\x13P\xebZ\xd8E\xff7\xaaL\xb9\\\r\xc6h\x1eyl-
\xdf\xfc\u6df9\xff\xbe\xfb\xf0\x85\u00eaK\xdf\xcf\xf2\xb7\xff\x11\u0353gP*\x8cb\xc2\xd0\x12P\u04
6b\xa8\x845br\x1c\x87\x96\xc9S\xc9\xd5\xd4s\xb4\xbb\x9b\xbd\xfb\xf6R_WK\xb1XD)\x99`\u007f'\
xaf\xd7\xfe\xf4\x050\xd8\xd7\xc3\xc1\xdd;i\x1e\xd7NC\xebx\xcaAH)\u0414BC9\x88\x92\x854\x94\
xb5\u5374\f\xf0\x8dU\xb0\b'\x9a\x88\t+\x86[\xf1\x9b+\xa5\x8bQ!\xda\xe8$9\xc8\xf2U\"\"\x11+\u05
b8q!O\xab@\x02\xcf#WS\xcb\xf4\x85\xab9\xb0u\x1d]\xfb^b\xff\xd3\x0fP\xd7\xd0H\xcf\xf1\xc3\xec\
xdb\xf0(#\xdd\a\t\xfd2&\f\xac\x1f\xba68\xb9\x1c5m\x13\xc97\xb7\x93kh\u00adm\xc0\xc9\xd5\"39\x
a4\x93A(\x85r3\xb6\x19\x14$]x\x02\x99':p\x8d\t<\xfc\xd1aJC\xfdx\xc3}x\x83\xbd\f\x1f{\x99\xe2`?\x
a5\xe1~\x06\x8f\x1cd\xff\xb3k\xc954\u0471x\x15\xb3\xcfZ\xc3\xf4U\xe7\xd2\xda9\x9b\xc0+\x12\x9
4KH\xa9\x10\x8e\x83t\x1c\xdb\u96f8\xf5\x11\x95\x01\xac1\xc7\xfcx\x8f\xd1\x11dc\f)\x8d\x8e\xfdM
E\xae\xa1\xd9\xdaf\xe8\x94\a<\x10\x86\x1a)e\xaeX\xb6\xbc&0\u02c6\xae\x11\x06=\u0139\x1d\x
b6\x90\xef\xf3\u037b6\xf5\xf0\xa5{~\xfa#\x1e\xfd\xc1\x8dQ\xed\xb6lf\x85L1H7\u00cc\x8b\xdfc\xbb

\xf2\xe2h\nB\xb1\xdd{X\x13\rA\b\x0f\x8cF\x9a\xc8\xf6&u|\xb4\x83jV\x1a%1\xccX\xf5f\x1a'Na\xe3\xbd\xdfb\xe7\u06bb\x19\xed\xeb\xe6\xf1\x1f\xdc\xc0\xc1-
\x1b8\xe3]\u007fF\xd7Eof\xb8\xb3\x89Y\u01a7\xcd\r\xc8\bC`\u018c=\xff\u007ft))q\xdc\f
9t\xf8\x10\xff\xfe\xc3\x1fs\xcb\x0f~\u020e\x1d\xdb\x197y:g\xbf\xe9C\u033b\xe0r\u0716F\xfc\xe20FWf\xd6\u048bPk\x8d\xc20\xaes\x165u\r\xf4\x1d{\x99\x9d\xbbw\x01\xe0\xba\xcek\x1e\xa0q\xf2\xaa\xc0{\x18\xf0\x03\x8fC\a\x0e\xd0\xdf{\x8c\x19\x8bW\xd0\xd0:\x8e\xc1\xd12\u00de\xcd\xca4Q\xa1\xd3X\x13-
\xdf\x18L\x18\u0372K\xa7\"\xdf3\x06\x19\x8e\t\x95\x90\x02\xa5\\\v\x8b\x1aM\xa8u\x05h0Q\x11OKB\xc4\x18Y\x9e\x81Rq\x94\xe6q\u04d8\xb5\xf4\"z\xbbw\xd3wd7\xeb~p-
~\xb9\x801\x1a\xe1\xb8dj\x1b\u0237\xb4S7\xbe\x93\xda\tS\xa8\x9f4\x9d|\xebD\xf2\xcd\xe3\xc8\xd67\xe2\xe6\xebPY\xab\x9f\x17\xcaI\b\\De\x9c^P\x19\x15M\x80\x11a\xe1\xa1\xd0+\u23ce\xe0\x8f\xf4#\v}\x04}\xdd\xf4\x1f9H\xcf\xdem\x1c\u0779\x91\xc1\xae\x83\x14\xfa\x8e\xb3\xfd\x91{\xd8\xfb\u0323t,Y\u0172\xb7\\\xc5\xfc\xf3\u0782[S\x87_*$Cs\xcaq#hX\x9c@pV\xce\x04\xd5\x1fM
\xb1\xb1p\xba\x14\xb8\xb9\x9a\x14\xa9\\)\xf4\xae\xe3\xe0y\xde`}}\u04de\u05e4\x987d`e\xab-
\xe4]\xa5\xf0\x8am\xa3|\xef\xf1\xc7\u05b9w\xff\xeb\xe7\xed@\x80\xdd\xee+\x18\xb3\x90h\xbfD\xc7i\x173\xe3\xe2w[\x02\xa4\\Np\xf4\xcaiE\x12\xe4\x15h\a\xe9Ysb\x13\x92\x1c\xe9*>\x956\xa7P\a\x1e\xda\xd3LY\xb8\x8c\xf1\xb3\xff\x0f\xe3f\xce\xe6\u017b\xbe\xcf\xe0\xd1\xc3\xec}\xf61\x8e\xef\xdd\xc1\x8e\xf5\xbf\xe6\u0eeef\xe5\x19g2\xbb)\u03f4|H\xb3\xf4QBW\x19\xe9\xbc\xde\x0f\u064e\xa3\x90\u02aa\x85F\xcbe\xee\xbc\xeb\x1en\xb9\xf9\x16\x1e_\xfb\x18A\xa0\x99\xbb\xec|\x96_\xf8>\xa6.9\x9b\xc0\u04d4\xba\a\x10\xf5\n2\xc2\x06\x16D\xb7g\xec\xe6\xe1\n;X\xd1:u6nM\x1d\x00\xbbv\xed\xe6\xe0\xe1\x97\xe9\xec\x98L\x10\x04'%\x89\xaf\x13(MH\xc5\xf0\xf0\b\x9b6m\xa1T,2\xb5\xb3\x93\xd6q\xed\xec\x18\xf1\b\u3390\x8a_\xb9L\x95\x1d\x13%V\xc4\xf7\x9a\x88\x9d\x12\x13a]T\x18\xa5\x83t2h\x1d\"D\\pL\xc5L\xcaD\xb2A\xf1\x1f\xc5,\n\xa6\xcd=\x8b\xfd[\x1f\xe7\xf0\x9egq\x9c,H\x89R.n\xae\x96|\xdb\x04&\xad\xbc\x80i\u7f55|\xdbdT&\x17\xf17!h\x8d1\xf6w\x1d\x04\x10\xf8Ie\xb4\xcf\xddDj\x96\x8az%\r\xbb\xe80@\x87>N6\x87\x93\xef\xa0&;\x9d\xa6\xba\x1c\"\xf0\x18\xed=\xc6p\xf7A\x8e\xed\xde\u010e\xb5\xf7\xf3\xf2\x96\r\xf8\xc5\x02\xbb\x9f|\x90\xae-
/px\u04f3\x9c\xfd\xa1\xbf\xa6q\xfcd\xca\xc5QL`'beR\xd0S$\xa8H\x17\xf6\xdf\"e4&\r\xb1c\x8c@erV>\x99\xf8\x1d\xd9G.\x9bedd\xa4K\b\xf9\xe4kR\xcc7\xf6\xc7<\x9an}|X\\\xf3\xe2\xf6\xae\x9a\x9f|\xe5\xd3f\xb0\xeb\xa0H\b\xcf\x18^\x91\n\ud5e9i\x9b\xc8\xdc\xcb\xfe\x94\x86)\xb3\xf1F\x86*\x8b\v\xa2\u022b\x10\x11\x86\xa8\xb2\xc6)\x9b(%E\x82\xd4cH\x97hzMG\xde\xceyA\xc00\xb5-
M\x9c\xf1\u078f2i\xdeR6\xfc\xf4\xbb\x1cxa=\u00fd\xddl\xb8\xeb\xfb\x1c\xde\xfc\x1c/\x9e\xffVN\u007f\u06fbY\xbah\x01\xb3\x9asL\xad14\xcb2\xae\xd1\x15\u00de\xd7\u064d,\x85\u0779\x91\x19\x00\xfa\x8ae\x1eY\xf7\x14?\xbe\xf5V\x1e\xb8\xf7\x97\f\x1c;\u02b8\x8eY,>\xedm\xcc]y\t\xf5\xad\x1d\x94\a\x8b\x04\x04\xe0(\x8co\x90\xf5\n\x99\x17d\x1cA\x10V\x16\\F\x1a\x1c)h\x9c4\x95\xfa\xf1\x939\xba\u007f\a[\xb7\xed\xe0\x85M[\xe8\xec\x98\xf2\x9fr\x84<y\xbd\xf2]y\x18\x868\xd2ahh\x98\xe7\x9f\u007f\x01\xc7q\x98\xdc9\v\xc7m\xc0/\xf6#\x1c\x89N\x87\x11\xa7<E\"\xe64\x82)\xa4=\xf2\a
\xab\xc8\xc4Ja\x92\xcaA*\x17\x13\x86\x15\x1dK2:o\x92oR=dc\x1b\x04!\x04\x81W\xa0e\xdc4f.<\x8f\xa3\x87\xb6\xa0\xc30\"g\xb5\xed\xd0{\x8f\xf2\xf2\xb3\x0f\xe1\xd6\xd41\xf3\xa2w\xa1\x85\xc0/\x17\xa3|\u0354\xd4PJT&\x8f\x8c|\xdb1a\x02\xf3\x98\xc8F1\fC\xeb\x03\xe3\x95(\r\xf6\xa12Yr\x8d-
\x84h\x8c\xe71\\.bJ\xc36|\xa2\xa1\x89\xc6\xf1\x13\xe9X\xbc\x8a\x99g\xaea\xdb\xc3?\xe7\xb9;nb\xf8x\x17\xe5\xe2\bO\xfd\xf8\x1b\x18\xad\xb9\xe0#\x9f\xc5\xcd[{\x02B\u06d0\u02884\xae\xe80NT\xf5\xc4V\xf0\x90ruL\xe9\xe1\xa5\x148\x99\x8c}-

\xc2\xd0Z\x13D\x03\x01\x8e\xa3\xc8d\xdc$\x14\xf8U-
\xe6\xbbz\x87\xd5\xec\u05ba\u0418\xc1\x86u=\xe2\xd6\xed\xc7\v\v~t\xdd\xe7\u0341\xe7\x1eO\xde\b1&\xdeSe\xf3\xccz\xf3\xfb\xe9<\xe7R\xc2XQ\x01\x88\xc8s\x810\xc4)\x858\xc5\x00\xe5iD\x18\xb1\xeeJV\x8c\xf6\xab\xec\u0362\x1dR\x82\xca\b\x94+\tJE\x9cL\x8e\xd9g\xbf\x91\ts\x96\xb0\xf5\xc1\x9f\xb1\xe5\u05f7st\xcf6\xbawo\xa6\xe7\xd0^\xb6<\xf2K\u67bd\x86S\xdfx)\v\xe6\xcfc\xfe\xc4f\xe64\t\xeaM\t\x11\x1d\x0142uC\xbc\x8a\x1dX\xb4\x8d\x9b\b\xb7\xccf2\b'GIk\xf6\x1f9\xcecOm\xe0\xe7?\xbd\x9dg\xd6=F\xef\xe1\x83\xe4\x1b[Yv\xee;Y|\xe6\x15L\x98\xb6\f!\x1c\u02a5\x82\xbd\x81d4\xb5\xe0\t\x8co\xa0V\x92\xa9U\xe0(4\x90\x95$\xc9*\xd9\xc6\x16&/=\x9d\xbd\x1b\x1e\xa3\xa7\xeb0\x8f?\xb5\x81K\xdf\xf4F\x94\x14\x91v\xf9\xe4\xf5Zv\xe5ah0\x04\xec\u067b\x97\xed\u06f7\xd3:~25\xeds\xe8\xed\xf5Q\xc3!8\xc6\xdab:\x02\xa3\x04:\x8a\x8d\x8b\x83\"\xac\t\x96J\xf0\x10\x11\x18\xc4\x18\xab\u0724\xaf\x14\x12\xa9\\\xb4\f\x10Z\xdbD\xa2\x14\x962\x16m1\xa9\x1b\xc5\xda\xce\xda.{\xee\xa2\xd5\x1c\xdc\xf1\x04\xfbw=\x83\x92.\x06C\xe0{\xf8\xa5\x02#G\x0f2xp'RHf\xbe\xe9}dj\x1a\xac\xea#\f\u0441oC,\xfc\x80\xd2\xc0!\xca\xc3\x03\x8c\x1e=H\xb1\xb7\x1b\xa1\xa2\xf0\x94r\x11od\x88\xd2\xc0q\xbc\xa1~\xca\xc3\xfd\x04\x85Q\xda\x17\x9d\u0182+>L\xae\xa1\x95\x902\xa1\x86\xb2\x16d5\xc8\xc0\xa7<j=\xd5[\xa7\xcc\xe4\xec\x0f}\x9c\xb6\xa9\xb3y\xf0\xc6\xcf\xd3\u007fx\x1fR(^\xb8\xfb\at,=\x95eo}\x1f\xa5\xa1~k/\x1c\x1f\xdd\xe3\x13\u0258\xae;60K\xac\x86\x8dA\xc62\x03Qq\xbe!\xb1)\u0589Z(\xc6\xfa3\xd9,S\xa6t:\xafz1?6<\"\xda\xeb\xebB\x80g{\xf2\u007f~\xb4\u031a\a~\xfa\x03\x9e\xb9\xe3\xa6\xe8\x04oc\xa2\xd2\x15\u0284\x01\u35de\u02fcK\xff\x14\x95\xc9\xe0\x8f\x0eG\xedx\x88\tB\xa4\x17\xa0\x8a\x01\xaa\xac\x91\x9e\xae`aBF\x0f\x93`eF\b\x9bo(\rZ\x82\b\x81\x82F\xba\x122\x82\xc0/\xa3\u00c0\xba\xd6\xf1\x9c\xf6\x9e\x0f3\xeb\xb4\xd5l\xbc\xef'l{\xf4\x17\x1c\u07ff\x8bc\xbb7\xd2{p\x17\x9b\x1e\xfc\x193W\xadf\xd9\xf9o\xe2\xd4U+X2\xb3\x839-5\xd4\xe3\xe3\x86\x1e\xa1\x8e\xdd\xd1xE\xbbS\x13\x1f\xa3\x8dA\b\x83#\x1dT\xc6N\x95\xf6\x97<6m\xde\xcaC\xeb\x9f\xe6\xbe{\xeef\u02f3OS\xe8\xebFek\x99\xb9\xe4\x1c\x16\x9f\xf5N\xa6/X\x8d\x9b\xab\xc3\xf7\xca\x04\xdeH\xb5\x9d\\L^\x155\xc6\xd3\b\xdf
\x1a!\x93\x93d\x95\xb4Y\x8aR\xa0j\ua63a\xfcl\x9e\xad\xfb&\x85\xc1>\x9ez\xf6yv\xbc|\x94\xb9\x93[)\x16K'@a'\xafW\xef\xd2\xda\xde\xf8\xe5\xb2\u01c6\r\xcf\xd1\xd7\xdb\u00fc\xe5\xe7\x90k\x9b\xc9\xc0P\x11\xe9\x1b\x84\xaf\xed@\x91\x12h\u01e0\x95\xc48\x12\xed`\x8b\xbcR\x18e\x87td\xa8\x11\xa1\xa9\xaaGBT\xf5\x93H\xa5\x90\xcaA\x87~\x95\x83.\txc\xaa\xb0|\x91\x82b\x84\x10\x94\xca%\x9a[;\x98\xbf\xe4|\xba\x0fo\xa30:\x80\xebf\xad\u076eT\x88l\x1eod\x90\xcdw|\x03\xaf8BC\xe7lr\r-d\x1b[\xc85\xb7\x93o\x19\xcf\xc0\xfem\xbcp\xf3?s|\xeb3\x04\xc5\x02\xa1WD(\a\xe5f-1*\x15\xd2u\"\vm\x89p\x9ch\x94\xbf\U00084170A\u03a5\x00\x1c\x19\xf9\x97\x87\x01\xc5\xe1\x01\xb25u,Z\xf3\x0e\x8a\x83\xfd\xdc\xfb\u5ff1N\xad\xe5\x02\xc7\xf6lG\x87aEzhl\xf1\x8d\xcd\xc8\xcc\x18\xe2uL4M\x85R\x88a\xa9\xb8\xe9\u0506\xd0+\xd9F+Eh\x1b\x03\xb9l\x96\xce\xce)\xaf\xfe\xd0\xd0\u06a3H \xdc\xdd3t\xf1^\xdc\xcf>\xb9\xfe\t~\xfd\xad/j\xc2
\x19\f\xaa\"\xd9B\x9f\xda\xf6\xc9\u033f\xfc\u007f\xd04m\x16\xe5\u0442\r\x8a\rBD9@\x15m7.\xfd\x10\xa1+\x13\x9fF\b\x84\x0e\xab(\x86\xc4`G\xa4\xce5\x1a(\x1b\x18\t!\u0087\xb5\x0e(\x17\x86Q\x8eK\xdb\xf4y\\\xf0\xe1\xcf0\xf3\xb4\xd5l}\xf8n\xf6nXG\xcf\xfe\x9d\f\x1c\xde\xc3\xf3G\x0e\xb2\xf9\xc1\xbbX\xbf\xec4\x16\x9cq>KV\xac`\xd5\xc2Y,\x99>\x89\xf1\xb5\x0e9\f\x84\x1e^`1\xae?dQ\x8fG\f\x1c)q\\\x17p\xf1\x81\x97{\xca\xec=\xb0\x8b\xed{w\xf3\u0533O\xf3\xe8\xaf\xee\xe5\xc0\xf6\xcd\xe0\x15q\xb2uL[t6\xb3\x96^\xc4\xcc%khh\x99B\xe0y\x94FG\"WCSE\n\x99h\xc4Y`\x03{\u0168&+C\x8c\x92hW\xe2FR,#a\u072c\xf9\xb4u\xce\xe4\xe0\xa6>^\xda\xf04O<\xbf\x89\xb9\x93/\xaa\x1a\xda:y\xbd\xfaW\x10h\x10\x92\xe1\xe1\x11\x9e|\xf2I\xbc

\xa4u\xda|L];\xe5\xbe\x11{\u007f\xe8JR\x90\f\r\x9a\x10\u0436\xa0g#_\x11\t\xc65Hm!\x96\x98\xb4
\x94P\x05\x1f\xc4\xe8\xafT\n\xa1\x1c\x8452O\xec]\u01de\x8eI\xe3\xc7\xd1}\xa9\x81\x92\xe73w\xf
1jvo~\x94-
\x1b\x1fAJ\xc7\u00ae\x91L\xd9\xc9\xd7Q\xe8\xed\xe6\xa5\u007f\xff\x17T&\x87[[O\u0744\xa9\xb4\
xccZ\u0338\x05\xabpj\xeai\x984\xcdz\xc0\x14G\b=\x8f\\S+\xf9\x96v2\xf5\u0378\xf9:\xb2\xf5\x8dd\
xeb\x9bps\xb5H\xd7%\xd74\x0e\xb7\xa6\x81
\xf0*\u0773\xb6\xb9\xa09mpb\xbc\x1d\xf0\n#H\xc7e\xd1\x1b\xae`\xcf\xd3\x0f\xb3\xf9\xc1;q\x9c,M\
x93:\x93\x0e\\V\xba\xad\x13\t\xe04\xb7i8\x11z\x12U\xb80ZkJ\xa3\u00f6sO\rve\\'\xfe&\x8f\xbf\xaa\
xc5\xfc\xf9\xeeQ\xb1bBmh\xf6\xfc:\xf7L\xa6\xfeo\xb6o=\\{\xd7\xf5\x9fg\xf0\xc8~\x19\x9b\u0724\v\x
9e\t\x03\x84\x90\u0338\xe0\n\xa6\x9e\xfb6\x02/\xb4\x129O\xa3\n>\xaa\xe0#\xcba\xc5\xc1-
R]\xd8\x01\x01\x8d\bmjI\x1c\x95m\xa4\xc2\b\xab-
\xb4\x8bP\xe0(\x81\x02L\u0672\xf6\xa2\xc1\xb1\xba;\xac\x1dg)\xf0Q\x8e\xc3\xd4\x15g\u04f1p\x1
5\x87\xb7l`\xd7S\x0f\xb3\xf7\u0675\x1c\u06f3\x15o\xb8\x9f\xed\x8f\xff\x8a\x1dO<\xc8#\x13;\x99\x
bap9s\x17-
b\xf9\xc2\xf9\x9c\xbaI!\x8b\xe6\xcc`\\mM\xbaW\xc2\x18\x9d\x04V\xc4\xef\xcao\xaby\xa2*\xd8VDV
\xb8\x95\x93K\x00\xec9\xd6\xc7\xc6-
;xi\xe3v6m\xda\xc2\xd6M/\xb0o\xfbF\xcaC=\x00\xd46\x8dc\xf2\xf2\xf3\x99\xb6\xe8\\\xa6\xcc?\x8b\
xa6\xc6NB?\xa0T\x18\u0184a\x92rb\x17j\xd4%%\xfffC\x1c\x05\xa0B\x83S2\x04\xa3\x86
\xa3Py\x97\xacR\xe8
\xd7\xd4\u01b4SWsp\u04f3\x8c\x1c?\xc2\u06a7\x9e\xe1mo\xbc\x88&G&\xb9\x89'\xafW\xbf+\x0ft\
x88\xa3$;\xb6\xef`\xeb\x96-
\u0535L\xa0}\xfeJF\xb5\xb5]\xad8\x01\xc6)C\"I\x90\x17\x81FZ'\x14\xf0\x02BW
4\u0220\xd26\xc6E\xca6O\x95\x82%\xa4c\xbd\u024d\u0104\xe6\x841\xf6\xb4\x8a\u00cc\xd5u\x1
8C\xc9\xf7h\xa9og\xc1\xd2\v\u0637k\x03\xc5r\x81L&\x97nJQN&\xc1\xbc\x83r\x81BO7G7=\xc9\xe
e_\xddJ\xcb\uc974/\\\xc5\xcc7\\I\xf3\xccEH7\x83r\xb3\x89\"\u013a\x12Z\x88VD>\xe3a\x10\x10\xf8
^\x15!\x89\x90\x84\xdaP\f\f\x8e2\xb8\x91\xef\xad\x04\xbc\xd1!\x9c\\\r\xe7^\xfd)\x1a'vR\xd3\xd8\u0
0ac3.\"\xf4\xbcd\xe8'\x9em\x91\xe6D}y\xfc\x9a\xc4\x19\xaa\xb1\x90\u008c\x11\xbfHi\xdf\u02d1\x9
e\xee\b\x86\x16),]RSS\u00e2EK^~\u054a\xf9\x86\xee\x11VL\xb0\x83A[&\\\xfc\x8e\xfdG\xcbk\xee\
xfe\xb7\x1b\xd8\xf3\xe4\x03Q\xb1\x92U\t\xd3\xf1.>~\xc9\xe9\u033f\xe2\u00e8l\x8e``\x98l)D\x16}d
)\x88r\x99t\xb2\x83\xc5\xc7\x11\x11\u0623`\x92I\xa8e\x1c\xe4Wy%\xc3Hq%\r2\x8e!)i\x8c\f\x10\xf
5N\xb2\xad\x1ac5\xb0\xa1\uf8e4b\xda)g\u04f1\xe4T\x16\\\xf06\x0eo|\x9a}\u03ed\xe3\xe0KOS\x1a
\x1ed\xe0\xf0>\xfa\x0f\xef\xe6\xc5_\xdd\xce=\xad\x13\x98:k\x1esf\xcfd\xe1\x9c\x19,\x9e;\x93\xe9\
u04e60i\xe2\x04\u018dk\xaf2\xd8\xfa\xcf^~\xe8\xd1\xd5}\x8c\xae\xa3\xc7y\xb9\xab\x8b]{\xf6\xb1}\
xd7\x1ev\xef\xdd\xcf\xce\xed\xdb\xe8\u06b7\x1b\x82\xb2]\xe8\xcae\u0714\xb9L]v\x1e\x93\xe6\x9
e\u0284)K\xa8k\x9c\x84\xf6|\xca\xc5\x02\x84:\xc9f\xac\xe0Yc\b\x9a\u06379\x9a\xed\x16Q\xfa\xb
a\xe3\t(J\u00bc\xc4S\x90-
\x97\xa9\xa9\xabc\xfa\xa9\xaby\xfaG\u07e4<:\u0313\x8f<\xc8\xf3W^\xc5\u014b\xa6a\xbc\xc2\xefl\
\v\xf2\xfa\xdd/?\x88B\x8d\xb5\u6847\x1f\xa6\xeb\xc8\x11&/?\x97\xe6\xd9K)\x95<\b-\xf6-
\f\x89\xef82vB\x8c\xd1Z\x89\x8a\xf8\x13U\xb2\u0713\x8cn9\x13y
\xe98\x861n\xaa\"\x0f\x13\xa52\x18\x13F\xa4\xa3\x8eh\u0394@\xbd\u2ddb@1q\xeeg\xa85\x05\x
cfg\xfa\x82s\x98\xdcy\x0f;\xb7?m\vo\x94\x04\x11\xdb\xef\x1a\xa5\x10FYO\u01e8\u0287^\x89\xee
\x97\xd6\u047b\xebE\xba^Z\xcf\xf8%g\u04b1\xeaB\xc6\xcd_\x89r3\x98\xa0\f~\x19\xc2

\xf1^\x89\xa3\xe5D\xe4\xefbR)E\x00\xe5\xd0\xe0\x866\xffS\x1a\x83\x89\x14*~\xb9H\xeb\xd4Y\x9c\xff\xe1\xbfKN\r\xa1\xefY\u062a*\x9c\xe27\x9d\xabM\u22de\xda\xe7*\xc0DdC,$\x04^\x89\x81\xeeC\x89
\u013e\x01\x9a\\\xae\x86q\xe3\u0195O9e\xf9\xab\u04d9\x1f\x19\x1c\xe1\xd1cQ\xd3\uc3f6=Q\xe0\x8bk\ufedb\xb5\xb7}+*\x1a\u056ezB\b\xc2\xc0\xa7v\xdcD\x96\xbc\xfb\xafh\x9d4\x8f\xf0\xd80\xd9Q\x0f\x15\x00\x81\x8e<\xb2M%4V\u0605)=[\xc8\x11\xa6\xc2\x06\x1bK\xc4\bm\xc7d\x8d\xb4/\xa4\xd4 C\x99\xec\x84B\x02%\x03\x8e\x86\x1aY\x95\xb8\r\x10\xe8\x90`h\x10\xa9$\x93\xe6-e\xe2\x9c\xc5\xcc=\xf7\xcdt\xef\xdc\xc4\xe1\xcd\x1b8\xb8\xf1i\xfa\x0e\xed\xc3\x1b\x1d\xa1\xd8{\x8cm\xbd]l{\xfa\x11~\x8e\xa0\xb9m\x1c\xe3\u01f5\xd1\xda\xd2D[[\x1b\xadmm475\xd1\xda\xdaB}}=\xd9\\\x0e\xd7\xcdD\x836v\xc4\u0646\b\x94\xe9\xeb\x1f\xe0\xe8\xb1\xe3\x1c;~\x9c\xe3=\xbd\xf4\xf5\xf7\xd3\xd7\xdb\xc7\xf1\xa3]h\xbf\x92\t\x9a\xcd\xd5R\xd36\x81q\xd3\x17\xd11\xff,\xc6O]B\xcbf\xa3\xc1\x82\x00\x00
\x00IDAT\xa4\x99d\xf2\x8d\xe8\xd1\x12\xa5\xc1A\x9b\xcen\u0184\x0f\x881\x06\x9ecL\xf8e\xf4\x9e\x18$Z+T\xe8\x90)I\xfc>\x8d_+p\xb3\x06\x9c\x80\t\xb3\x171u\xd9\xe9\xec\\\xff\x00;\x9f{\x8aG\x9e\xda\xc0\xf2\xf9\xd3hS\x12\x1d\xe8\x93\xc3C\xafvW\x1e\x84(\xc7\xe1\xc0\xfe\x03<\xb1~=\x81PLYt\x06\x8e\xd3@a\xb8\x80\x0el\xc7h\vz\xa5\xa0\x1at\xe4\xa9#PH\xa4\x0fF\x9a\x8a\xe2BG2;i\xd0J`\x14\x84R \x8d@\x06\xd1\xf4\xa7\x14\b\xe1
\x8dk!\xbctdb<\xb6n*^\xe3\xe9\x8f\u0152A/\b\xc8\u05f6\xb1h\u0645\x1c\u073f\t\xdf\xf7q3\u0654\xe5m\xf5\xf0LlK\x81\x90(7\x8b\x0e\x02z\xb6=G\xcf\xf6\xe79\xb8\xf6n&,9\x93)g\xbc\x81\xc9\xcb\xce\"_\u05c8\x0e}\xc2R\x11\x13\x06\xf60.L\x14\xe5f\xed~U\xd4\xc8H
\xc4P\xf45\xae\x14d\x85\x8c\x8c\xe4l\xf3\xe9\xfb\x1e*\u04b2\x1bm3\v\xd2F\x06R\x8a\u02ae0F\xfd\x93D\xdcE\xbfK\xaa\x05\x8b\xb1%\xc9\xf0\xf1.\x86\x8e\x1dI5\xbe\xf6g\xad\xa9\xc9\xd3\xd1\xd1\xf1Rg\xe7\xf4\x97\x00v\xed\xda\xf1\xca\x16\xf3m\x83\x88\xabf[M\xf9N\xa7\xe6\xcf7<\xb7\xb9\xe3\xe7\xdf\xf82\xde\xc8P\xc5C8\xfd\x86\xea\x10\xe5\xb8,^\xf3G\xccZ\xb4\x06ul\x14Y('\fzl\xfc\x9b(-\x05\x88\xd0
}\x1d\r3\xa4X\xe3d\x81\x88\xca\xc8l\xec\x02d\"\xf1\x94\xa8\u021a\x84\x06S\b\x11\n\xc8\xc9\xea\xe2\x86\u0571k\xad)\x0e\r\x80\x10\u0536\xb63\xe7\xac7\u0439\xfc\f\x96\r\xbe\x97\xfe\xc3\xfb9\xb2\xfdE\xbavl\xa2g\xff.J\xc3\xfdx\x85\x11\xfa{\x8e\xd1\xdfs\xec\x84\xd7F\np3\x19\x1c\u01f58c\x14{d\xb4&\fmZJ\xf9\xb7\x848g\xf3ud\x9b\u06e8in\xa7m\xd6b\xc6\xcdXF\xfb\xe4\xf9\xd47M\"\x9fkB\xe1\xe0\x97\n\x94z{\x11Z$\x1d\x80\xd1&\x05\xab\x90\u023a\xaa\xe2\x1e\x93p]S!\x92\xa5\x02\xe1
\xb4\x83(\x83\xf2B\xf0\fA\x9d\xa4\xe4\r\xd3\xd8:\x89\xb9\u7f45\x9d\xeb\x1f\xc0xE\x1e\xfe\xe5=\x9c\u007f\xe1\xc5\\<\xbd\x11',\xe0\x9f\xac\xe6\xaf\xf8%\x84\xf5W\xf1\xfd\x900\f\xc9fs<\xf2\u0223l\u077a\x95\xd6\tS\xe9\x98}\x06z$\x80r`\x15\xbb&\x1d*n\u05de\x8c\xd15%\x90Q\x97NhR\x1dqtOi\x8b\xf7j!\b]\x81\xd1\x12\xa1\xb5\xbdO\x8dU\xb6\b\xe5
B\x1f\xa3\xd3\x16\xafT\x85>\x8b\xb4{a\xfcA#\b\x82\x10\xdfHf.8\x8b\xc9\xcf\xfe\x82}\xfb\xb6\u0635+\u0485<\xa5\xd1\x16\"j\xdc*\xae\x83\"\"n\x87\xbb\xf63\xd2}\x88\xae\x17\x1fg\u0082UL;\xeb\x8dLZ~.\xf9\xc6\x16k\xd0U*\xe2\b\xaa\xf0nA\xf5p\xa0\u0586R\xa0q\x94\xc0!\n\u0280$pF\x18\x91\xd8\xf1\xc6E\u061e$*\x83C\x82\x94\xbdA\xb2a\x9c\xe8\xfd\x14\x8d:\xa2\x94B\n\xe8\xda\xfe\x12\xc3}\u01d3\xcd\xc1\xdapHr\xb9\x1cs\xe6\xcc~\"\xfe\xfff\u03de\xfb\xca\x14\xf3\x83\x03#t6\xd5qag]T[\xcd\xe2\u06f7\xf7|\xfa\xc7_\xbf\x8e\u00db7\xfcFxE\nI\x18\x86\xcc8e\rK.\xf8cra\x0eot\xb8\xc2\xfe\xc6$\x89\xa80\xc32\xb4\x8c\xbc\b+\v\x04\x93\xda\xe5\xa9\x18\xea\x98\x04\x96\x13\bi\x1d\xc9|$\

x19a\x902\xaa\xae\x010\xaa\xed\xa0\x84\x9bJ\x1eO0\xb0\x94\x13b\xb1@
\x04N&K\xcb\xe4\x19\xb4t\xcc`\ua2b3\xf0J%\x06{\x8fs\xec\xc0^\xfa\x0e\xef\xa6\xff\xe0\x1e\x06\x8f\x1e\xa68\xd4Oyx\x00\xbf8\x82_*\xe1{%\u029eO\xb9<:\xf6\xb6\x04\xe5\"3YrMM\xb8\xb9<\x99|-\x99\xdazr\r\xcd\u050f\xef\xa4y\xea\\\x1a\xa7\u03a6~B'\u0646\x16\x1c\x95%\xeb)\x18\xf5\u0445\x02a\xb9h\x8f\xd2(\xabJ1\xa6z\xdb7\xa6\x1a\x97\xa7\xe2\x05m\xa8V\x04\t\xa9\x90\xd2E\xaa\f\x02\x05\x81]n\xca\xd8\x00g?\xab\xc9\xe7\x05S\x97\x9d\xcf\xc4\x05\xcb\xe9\xda\xfa\x02/<\xf4\v\x1eZ\xff^fO^C\xa7\x94\xc8\xc87\xfb\xe4\xf5\xca]\xf6T\x17R\xf6|\x94r8p\xe0\x00\xf7\xdd\xff+\x06FF8\xfd\x9c\xb3hn\x9eN\xe0\x85\t\xe9Y9\x8e\x99\xe4\x18&\x8c\xb4]v\xe4RJ\xc5f%)\xc2I\x90\xb1
\x95\x9c\x1c\xed\xf9\xc2rUV\x8b\xa0,,
C\xdb\u045b\xf4]\xa9\xab\x03\x89\xc6\xe8\xd6\rP\xf4<\x9a\x9a:X\xba\xf2M\x1c9\xb2\x97
\bP\u02a9\x10\x86\xa6\x02\x87\x86a\x88\x10hG9\"\fC\xa1\x03\x9f0\x88\f\xf7\xa4-
q\xc3\xdd\a\x19=v\x84\x97_x\x9c\x89KNg\xfe\x9b\xaeb\u0082Udkj\xc0+B\x18\xa2\xa4\x8d\x88\u04c0\x17\xa6\x03\x9e\x05\xa1\xb1\x01\x1d\xb5\xd2$\xd8u\x1c\u0111l\x88\xa2\x12\xa5\x97\x14u\xaa+\xb6\x10)\x188\xc5\x05\x1b\u049b\x1cH%\t|\u0631\xeeW\x84QS'\x85\x04cp3\x19:;;\x99>}\xfa\xa3\xe95\xf0\x8a\x14\xf3\u03a6:\x8e\x8c\x96\x99Tk\xa7\x0e\x9f\x1d\xe5\v\xbf\xba\xeb\x8e\xdc\x13w\xdc\x12U\xee\x8a\x13\xa2\x05\xfa\x15:\xf4i\x9d<\x9b\x15o\xf9\v\x1aZ;)\x8e\f&8\x9a\x88\xbc\x1eLD\x02\x1a\xa1\x11\x81%d\x84\xa6\x92p\x93\xb2\xbe5b\xcc\xd1&\x15\xfddM\xf25^`\xacw\xb1\x93\xac*K\x88\x8e\x82hP\x95x\uf60d7\xa6*B\xca\x18CP.\x13D\x13\xa9\xc2qqs5\xb4M\x99N\xd3\xe4\xe9hs\x01A\xa8\x19-
\x96\x19\x1d\x1a\xa0\xd0\u007f\x9c\xd2@\x0f\xe5\xe1~\xbc\xd1a\x82b\x81\xc0+\xa0\x83H\x9b+%N&\x8b\x93\xad\xc1\xa9\xa9#[\xd7H\xa6\xb6\x9elC3\xb9\xa66r\xf5M(7WI\x8f\x05t\x10\x12\x84>~P\xc0\xf1}T\xa0\x13\x85B%p0-
\x1c\x88\x19\xf6\xf8\x06\x16iuY\nw\xb27\x82\x90\x0eB\xb9v\x80+\xe9H\xc0\x04\x91\xb1\xbf\x94\x94\x06\x86\x187~.s\xcfz\v][_\xc0\x1b\xee\xe3\xa1;o\xe3\x8cs\u03a5\xa5#G\x9d(\x9c,\xe6\xafpW\x1e\x04\x9a\xb2\xe7\xe3\a\x01\x99l\x96G\x1f[\u02f3\x1b6\xd0\xdc6\x89\x99K.@\xca,Aa4\x9daV\x89o0\x15\xcc\xda\xe2\xdb\xf2\x04\xf5\x85 \x95t\x9f\x92\u0449\xd0~\xaa\xebH\x9c\x88
\xb1\x84\xaa\"t\x1ckb\x15\xc1\xa1\x15\xb5Fe\u011e(\xad\u0224<\x94$\x06?\xf0(gj\xe9\x9c}\x1a\xe3\xda\xef\xe6\u0211\xbdQ-
\x90\x91\u72c98\"I]}\xab)\x16Kr\xa0\xf78H\x11*\x89\xc4\x18\xa1CKrZ\u0264\v\x02\x8a\x83\xbd\xec[w/=;^b\xc69\x970\xef\xc2\u02d80s.\x19GQ\x1a\x1d\u0173)\x17d\x94\xed\x9eCc\vy`\x04~\xe4c\x93\u02d0H\x18IQr2vG\x15\xe2D\xd1NU\xf1Oo`c\x06\xfd\xa3\u04f1r\x04\xc7\xf7\xecd\xdfs\xeb\x00R\xf9\x9f\x06\xd7\xcd\xd0\xd9\xd9\xd9=y\xf2\xe4g\x01\xae\xbb\xeeZ>\xfe\xf1O\xbc2\xc5<\f\n(\xc7\x16\xf2\x97\x8d\xf9\xd8m\xf7={\xd9\xdd\u07f9\xde\xc6s\x83\x90c\x16\x8a\xd6!\u065a\x06\x96\xbd\xf1O\xe9\\|>^i$\xe5\xbef\a\x19b\xa3,!%\xb2,\xec\x00C*\x930\xc9\xeeHgIUy\nW\xceR\u0080\t\xad\x87\x84\x0e\xac\x86\xb5*\xe2{Tc\x14\x88\xdaH\xab\xae\x89\xa4V\xd5T\x86I\x919Bk\xf0\xca@9\xe9j\xa5T\xb8B\x91\xcfg!3\x81|\xdb\xe4J\x8cZ<(
\u04a9%\"\xbe\x13\"c\xba\xf8\xe8\xa8#\x9bQC\xa8C\x1b^\x1b\xb9\xd4\xc5x\xa1q\x05\"\xa7\x90e\x93\xe8\xea\x05\xd5\x1dy\u0711I!\x93\x9b\u0264N=\t\xeb.\x94\xbd\x01\xa4\x8bRN\xe4_a\xaa\xa6\x04\x85\x89\n\xbag\bU@\xb6\xd6a\xf6\xca7\xb1\xf9\xc1\xdb\xe9\u0677\x9d\xadk\xef\u7a67\x9faJ\xeb\xb9,\xccJ\x94\xd0'\xc9\xd0W\xa4\x90\v;%\xe9\xfbx\x9eG6\x9b\xe3\xf0\xe1\xc3\xdc{\xdf\xfd\xf4\x0f\f\xb0\xea\xfc7\xd02y>\xa1\x1fB\xa0#\xbf\"\x12\x88\xc3\xca\f#\x92\x1b\xab\x9a\x12B\xd9\xf7WX\xb23N|\u0462:?GG\xd0BNJ\xb2Y+$

4\x18\x19}L\xabJb\f\xf1\xa8\xe9\x18\x9f\x12\x93Z\xfb\xa9/n\xb4\xa1P*Q\xdf<\x85y\v\u03a4\xbbk_
J\x9e\x17Y\x0fD\x039\xabW\x9f+::\xa7\xbd\xfc\xc2\u018d\x93w\xef\u06a9\xf6\xef\xdcn\ub0b4~&:4
\x11\x89j\u05f5\x00\x86\x8f\x1eb\xe3\u03feC\xf7\xc6\xf5,\xb8\xf0r\xe6\x9f\u007f\tM\x13;\x19-
\x96(\x95\xcbv\x03J\fj+\x9a\x1b/4d\xb4\xc1Q\x95\xe7\x11\xf3\f1\xf9\x99\xa44\xa5\x92\x8a\xe2B\x
9cL\xacF7\xa5L\v\u0222_\xa5P\xb8\x19\xd8x\u07cf9\xbe\xdfz\x1eIe}\u0305\x90477\xd1\xd8\xd8p
\xeb\u99dfy\x04\xe0\xf2\xcb/}\xe5\x8a\xf9\v}B\x02\xda\x18s\xfa\xdd;\xfb\xfe\xe9\xce\xef|\x95c{\xb7
G\xb4\xb6\xb5r\x14VRb\x85\xf5\x02f\x9dv\t\v\u03bb\xca\x16Y/\x88\n\x87\xb1\xb0J$\x19\x14\xda
\x03\x81\xf4\x05\xf8Q\xdd3q\xe7]\tI\x8d\xb3\a\x8d\xa9:\xc0$\xb8[\x14\xa7\x12\xad+\x83\x0e\xc2h\
x047\xeaV}\"\x8a\xdbA\xd4\xc8\x13:r\xfb\xb5S\xbe\xb81\xfb,\xd2\u040e\x86P\xa3M\x00\x06B\xdf\
xee\xee1\xd6fDu\xa3R\x95P\x12\x1dI\rc&X\u0349\x98\xb7I\x1d_\x83\xacDf\xa5\xdd\xe8b2\xcaT\b
a#H\x05\xf6\x926g\xab\u0605FS\xb8\u05ab\u0669$\xb1\x8c\x91\xc0\xc6I\xe4\u06b7)\xbd\xe5\xa
1a\xa6\xccX\xc1\xfc3/\xe1\xf1}\xdb)\xf4v\xf3\xf0m7\xb1h\xe5\u9d0c\xcf2\xc5-
\x9e\xb4\xc4}\x85\xf0\x95
\b\xf1\xfc\xc0F\x9f)\xc9#\x8f<\xc6\xfa\xf5\xebii\x9b\xc8\xecS\u0788r\xeb(\x15KQg\x1b\xdd\x1f\xa6
\u04bf\xc4P\x8a\x10\n%\xdd$\x872>\xb9\x19a\tO-
\xa5%M\xa3\xc9F\xa3\xb0\u0120\x13\xf1|\x0e\xa9p\xe2\xc8$/\xe6\u0174\xb40\xa6\xd1T\a\x8f&r\x
96d\xb6!n@\xfc\xc0C\xe4\x1b\u9735\x92\x86g\xefchx
\xcab\x96\x897\xb8\xef{\x94\n\x05\xae|\xd7;~\xfa\xa1\x0f\xff\xd9\xfao|\xe3;\x97n\u06fa\xe5\x8a\xe
d\x9b7\xe5\x8e\x1e>@\xe0{\x91\xaa&\x88\x1cU\xa3\xcdJJ0\x9a\xae\x1d/\u0473\u007f'\x877>\u0
242\v/\xa3s\xe5j\xea\x9b\xda(\x97J\xf8\xbe\aZ#\x05\xf8\xa1\x85H\xe2\x00x7\n_Q2.\xe4\xd2n\x84\
xca\xde?FV\":D\x95\u0657\x957J\x11)i\xc6\xeem\xc6P\xd3\xe0\u0435}'\xcf\xff\xe2G\x16>q\x1c\x9
4\xb4V\xbeJI\xa6M\x9b\x1e.[\xb6\xec\x81\xf8\xff\x99>}\x96=\xa5\xfc\xa1\xd7\u05be\x82\x91\v\x9b
\\\r\xf0\x96\xff\xf5\xf9/\xfd\xf0\xdb\xdf9\xe5\xeeo|9\x04T\xea<E\xack5:\xa4c\xe1\x19\x9cs\xd5\xe7i
h\x9d\x867:R\x91\xebD\xd2'\xa1\x05\"4(_\xa3|PZ\xa4
\x841\x05\x8e\xb1\x12\xbbT\xe7\x9b\xcasE\xa4vTc\xacdQ\u01dd\xbe\x01?\xfa\xb7\xacD\xb8c\xb
2\x00\x93\xef'*P\x10\x8c\x89\xa2\xaa|\xcc\x0f\r\xe5\xd0$2\xb0\x18\xb2\x11Qwm}\x8au\xd2m\xa7
C\x91\x93\x82m*\xf7@|<\xad\xcaI\x8c\xb0K-
,\xfc\x14\x0fx\xa4\xd5`UZ\u05b4\x061\n\xb3\x95\xd2A\xban2)g\x17\xfc\tMT\xa2vI\x9e\x8b\x10(i\xc
8\xd5\xe4\xc9\xd67s`\xcbz\n\x03=\f\x1e\xef\xa2q\xea<\xda\xe7\u0367Y\x84\xd4:\xe6$\xdc\xf2\a\x
86WB\xad){\x1e\xbe\x1f\x90\xaf\xa9a\u03de\xbd\\\u007f\xfd\r\xec\u0739\x93\xa5g]\u01bcS/'\u050
2
\xb6j0\x11\xb4\x96\xea\uc171\xd3\xd7\xcaq\x91\xcaI\xab\xe8l\xef\xa3\x04ZEo\xba\x02\xa3$AFb\\
A\xad#\xc8\xc4\x1f\x13\"\x82\xf8\"\a\xc6P\xa3uh\xbb\xe3\xf8\x84\x9d$\x17\x8f9\xe9\x91\xea\u042
3nAG$|]\xae\x86\xfe\xa3\xbb\xe9:\xb2\a%]\vgD'\\)\x04\xa3\x85\x02\x8e#7\xbf\xfb\xb2K\xbf\xf4\u03
3a\xc7~v\xedw\u007f\xf0\x8c\x10H\xc7\xcd,,\x95\u02b2\\\xb6\xf6\xb9\xb6V\x1aat\x98</k-
\x1b\xd2s`7\x876=M\xef\x81]()i\x9a<\x8d\x9a\x86&\xeb4*\r*^\xf4Ql\x9bR\x8a\x8c\xe3\xa0\x1c\x85
r\x14\x8e\xe3\xa4|\u0355-
\xda\"M\u041a\xe4\xf9*Qi\xae\u0318\x8d9[\x93\xc7\x04p\xdfu\x9ff\xfb\xda\xfb\x91R\xdab\xae\x14\
x80inn\x16\x17^x\xc1\x86O~\xf2S\xd7^s\xcd5\x85\xad[7\x89\xaf}\xed\x1b\x89\"\xe6\x0fzm\xef\x1e
T\x00\xbd\xc6\\\xb5\xee\xa1\xf5W\xfd\xfc\xa6\x1b\xad2u\xec\x14d\xe4\x02\xd68\xbe\x93\xb3\xdf\x
f1\xb7\x8c\xefX\x8272\x12\x8d\xf6F\xc56\x04\xe5\x19T9\xc4-
j\x94\x17\x17)\x81\x92.*\xc2r\x11\"96&\xaf\xb9\x10\x91\u02df\xf8\r+\xa6R\x95uh\u0426\xf2A\x11R

\xc1\x15K\x1a=\x18`\x02\x92\x01\xa4\x8a\x06\xfb\u0109\xaex9V\n\xb1\x1dN\b\xb4FkK\x18&\x8c\xbb\xae\xb8\u02c9\x14\xbc]\U00069c0fP[\x88\xc5FhE\x12*c\xd9\xf6*\xd7\xc6\xc8HH\xbb\x02\xbfF\x1286n+\x8c\x8e\xbcUi(\"\xea\xab\x13\r\xa2\x1d\u00d6\x8e-
\xe4(\u01e6\x80\x13\x1d\x99SG\xc1\x84\x9fH\x9fTB\x83\xf6\xc1\x1b\x19a\xe2\xf4\xa5,^\xfd.\x84\x14\x8c\xf4\x1e\xe3\xd1[\xbf\u0266\xdd\xdd\xec\xf7s\x94C\x13\xb90\x9e\xbc\xfe0\xf0\x8a\xa1\xec\xf9x~@&\x9b\xc1\xf3<\xee\xbe\xfb\x17<\xf3\xcc3\xb4O\x9a\xc9\u0715o\xc6\xc9\xd5\xe3ye\x12\xec.nXte`G`\xb3>e\xe4\x8bdHd\x1d\x95H\xb9\xd4\xfd\xab\x1d(\xd7Jd[\x86\\s\x06\x91\x89&4\x1da\xff\xec\xdaiN#\x04B\xc9\xc4J\xc3(\xecC$\x06\xa6\x89-
E\xfc\u0426\xb2\u07a5\x90\x94\xbc2\xd9\xc6\tt\xce\\N\xc6\xcd\u0606H\xa4\bF)\x18\x18\xe8\xe7\xf8\xb1\xe3o\u0771s\xd7B\x80\xb7\x9fs\xca\x03\xdf\xfe\xca?~\xe0\xfd\u007f|\xf5\x9b\xdey\xd5\xfb\u007f\xb5\xf2\u0333i\x9d0\t\xa4\x93\x0e\xdb\u0104!\xa1_\xb6\x1e\xe9\x02F\xfbz\xd8\xf2\xe0\x1d\xfc\xfa\xfa\xbf\xe5\xa1\x1b>\xcd\xd1-
\xcf\xd0\xd0PO\u02f8\xf146\u0593\xc9f\xc1\xcdbT\x862\n\xed\xb8\b7\x13=\\\x90\n\x13E]'gkQ\x11p8R$J\xc5Tnv\xd2df\xf2y\\\a\x1e\xbb\xe9Z\x9e\xfa\xc9wm\xb7\xed8\x11\xf4\x85\tC-\x16-
Z\u013cys~$\x848>n\\\x9bZ\xb0`qr;\xfdA;\xf3\xa7\x0e\r\x88\xf3\xa75\x85\u0198\xa6\xfb\xb7t\xfd\xeb\xbf]\xfbOSv<\xfdh(b\x11uJEa\x8c\xc6q\xb3\x9c\xf5\u058f\xb3\xe8\xac+\xf1K%K\x04\xc6<\x896\xa8\xc0X\x82E\xa7\x18\xe0\x8a\x99\x83=.\xc5\x1dA\xfc\xfbX\xe8\xa3J\xd5)\xaa\x8c\xf1\x13T#6\xaf7\xd5\x04\x86\x9d\x8c0V\u0652U\xd5#\ufc5f\x82H\xdd\x14\xa9\x82\x19w\u035e\x86b\x90\xf2\xd8I\x9f\x0e\x12\x83|\xfb\xd6[\xa2h\f\xfcq\xc2\xc8/'<\u007f!R7\x9c\x10v\x98\xc3\u0122\x01\x11I\xc6R\xafQ,\x9b\x8a'\xf6\x1c\x17\x91\u0260\x1c\xbb
\xb5J\xe9cS\xafk,\xbdJ`\x19Y\xbdO\n@e2\xb4u\xcc\xe0\xe5\x9d\x1b\x188z\x88\x81#\a\xc9\xd471a\xf99\u4961\xd5\xd5'\x90k'\xaf\xffb!\x8f\u0580\xe7\x05\x94\xca\x1e\xc6@]m\x1dO=\xf54\xd7^\xf7\xaf\xf4\xf5\xf5\xb1\xf2\xfc\xab\x98\xb9\xfc\u0354\u02be\u0748\xe3\xb5l*\x837\x96l\x94\b\u9814\x9b\x14\xf3*\xf6.\uaa0d\xac,\xdc\xc0\x11\x98zEc\xa3C\xaeVAF\x8e\xa9Pv\xdd\xfb\xc6j\xd75\x1a\x1d\xf9$Y\x8bQ]M\x10\xa61\xe34Y\x18\xe1\xcc\xca\u0252s$G\x0fm\xa1\xbf\xefh\xc4\xe3Tn\n\xad\r\x19\u05eds\x1c\xe7\xc1\xfb\xef\xbbo;\xb8\xea\x9ak>\xa7\xef\xbd\xf3\xa7{\x9f]\xbf\xf6\xd6\xfaq\x13\x0f\xb8\x8eZ822\xd2R,\x96D\xe0\x95\x01aD\n\xff0QL\xa5T\x8a\xd2\xf0
\xc7vm\xe2\xd0KO\xe1\x15Gi\x9e4\x95|\xdbxL\xb6\x16\xa3\r\xa1\x94h\xe5D\u0779\x8ax\xaf\xd4t\xa6\x00W\u0187^\x81\x12\x16\xeb\xafVb\xa7g_\x05\xf9\x86\x1cJ\xc0\xba\u007f\xff&\xf7|\xe5o\t}\x1f\xc7q\xad@CJ\x82
\xd4\x13&L\x90+V\x9c\xf2\xe0\xe7>\xf7\xf9O]s\xcd5\u07bd\xf7\xfeB\xde|\xf3\xf7\x93r\xf0\a\xed\xccO\x9f\xd2d\x00^\x1c\xe1\xf2\xfb~\xfc\x833\xd6\xdf\xf5C\xbba\x981\x9dlT|\xe7\xad|+K\xcfy?B+t\u0673\x9a\xf1\xd0
\x83\xa8\x90k\x93\x14\xf2j\xb2\xa4\x82\xb5\xc9(\xe8U\xa9\fJ:Q\x81\x97\x95\x02O\xcac2)\xba\xd5\xf5D\x87\x10\x84\xb1A\x96H&\xe0@@ \xa0`\xc1-
\xa1RE0\xaadf\xccfB\x14\xf6j\xb0\u01f1b\b^\x84\xd1K\x91\x86\xbe-
\xf9\x14K\x9f\xd2A\xe6c1\xed\u050cDu!\x17\x15hE\xa7:h-
\x05~N\x11\xba\x12\xa3\x04F\u026abo\x84\x04\xa5\xec\xa8s&\x87\xccd\x11n\x06\xa3\x94u\xcfs\x05AF\x10:\x820\xfa\xbbQ\u0092a\xa9<\xd3D\xd2(m^d\xe0\x19\xc2b\x89\xd6\t38\xe3\x1d\x1f#\xdf\xd0L\x18\xf8\xac\xbd\xf5\x9b\xac\u007f\xf8!\xb6\x15\xb3\xbc<\xaa\x91\x9ctT\xfc\xbd`r\xb0\x83e\xe52A\x10\x92\xc9dx\xf9\xe5#|\xeb\xdb7\xb1s\xc7\x0ef,<\x93\x85g\xbe\x03T\x8e
\x8c\"\u03f5\x1d\x96\x8bo\x85\x8a<N

#\x99pZ\xfcDJ|\"\xb0\xf7\xa5\b\xa3\xf5\xe5J\xea\xf2\x8aZ'\xb2\xe2\xa8U\xc8V\x17\xd1\xea\"\xea\x1d\u00ac$pA\xe4\x14\"\xe3\"\x1d\aGH\x1c)P\xaab\xdf!RSj&u_\xa4\xbe=BH\x8a\xe5\x12-
\x13\xe7\xd19c\x99\xdd\xc8t\n\xeb\xc4\x12\xc0{\xf6\xec\xe5\xe0\xc1\x83W\x1bc\xf2\xe0\x87\x1b6o\x8b\x90O\xa1\xdfs\u025a\xef\xdd\xf4\xf5\xaf\xce~\xff\a\xde\xff\xe9\x8b.\xbe\xa8w\xe6\xdcy\xe4jj\x84\x91J;n\xc6\xc88cX\x87\x84\xbeg\u036d\x82\x80c{\xb7\xf1\xf07\xfe\x81\x1f}\xea}l\xb8\xfd&\x8a\xbd\xdd\xe4\xeaj\xa9\xc9\xd7
\x8dI\u049at4\xc3\x11\xd1T\xa8\xf4\xcfV\xf53\x99\xea|\xd4\xe8\xf5\xadk\xc8\x12\x16}\x1e\xfe\xc6\x17\xb9\xf3\v\x1f\xc3/\x15\xadE\xb7\x92\x91\x8d\x87\tkkk\u054a\x15+JW\\q\xf9\x17\x85\x10#\x9f\xfb\xdcg\x9c\u056b/\xa8J\x81\xf9\x83u\xe6?{\xf4)\xf1\u36ff\x8b1\xa6\xf9;\u07ff\xed[\xff~\xe3\xb5\xe3G\a\xfb\"VWU\xfc\u0123\xe3{{\xc7\x02V\xbf\xe33\xb4N\x9cK\xb90RU\u016a\xcc\u0628\x16j\x8eEJ\xa2\xe5\x98\n~\xae\xd6|\xa6\xf1\xf9\xaaH\x8f\xd4G\xe3\x89+G\t\x94\xac\xdep\xec\xc6\x00\"\xefX\xfc<Z\xfc\x89\xbc2\xb50\x85\x8c\xe7\xc6\xec\xdfK!\x14\x02S\xe5A\xa1\x11\t\xb9\xa9\xa9\xf8V\x98\u07d0\xf0md\xe5y\x8b1I\xe0\xc2\xd8\xc9;\x11O\xf2\x85\x06\x19\x82\n@z\x06\xe9[O\x15\x15\x18\xa4\xb6\xa7\x0e)\x04\"\x92i)\xc7E\xb9\x99\xc8\x01\xb12\rk\xe2\xae\\F\x18\xa7\x14\x15eL\xf4}\xd3oB\xb4\xb7%c\xd1\x12;\xd4\xd0<y\x16}\xc7\xf6\u0475\xf3E\xca#\x83\x8c\xf6\xf72y\xc5yd\x1b[h\x91ej]y\x12?\xff\x1d:r\xb0\xa9\uc972\x87\x1f\x84(G!\xa5\xe4\xfb7\xff\x80\x9b\xbf\u007f\v5\xf5-
\x9c}\xc9_0~\xe6\xa9\x14\n\xc3\xf6\x84\x1a\x91\x96\x95[\xa6\x92\x8b)\xa5\x83\x94c\x93qRS\x8a2\x82K\xe2\x10f\x05\xb2\xc1v\xe5\x99\xe8\xc0\x1d\xf352+!'m\x03 \xc0\x91\x96
U\u00a0\x84FI\x83\x02;\x89Le\x88M\xa4\xe4\x8a\xe2\x84S\xb6=\xc5g\xf3\r\x84\x85>^>\xf0\x12\xa5r!\"\x1c+\xcfW\xeb\x90q\xe3\xda\xe6ds\xd9;o\xbb\xf5\xb6\xaeo\u007f\xfdk\u0737\xf6\tf\u031e#\xd6>\xf4
\x00\x8f<\xf8\xe0\xba\xc7\x1ey\xf0\xa1\xc1\xa1a\x84\x10\x1d\x9e\xef\xd5\x0f\r\x8f\b!\xa4\x96R\x89\xc4\xdf)\xad\xd8\t\x03\x06\xba\x0er`\u00e3\xf4\xed\u07c9R\x0e\xf5\xcd-
\u0536\xb4Y\x8f\x980\xc0\x11\x06%\xad\ubd8c\xeeSM\xac\xb2K\x87R\xa4\xd1\x02\x83r]j\x1b2\xf4\x1c8\xc4\xfd\xff\xfa\xf7\xfc\xea\x1b\xff\x94\x88C\\\u05cd\xe0\x15a\xc0\xc8\x05\v\xe6s\xf9\xe5\x97\xddt\xe5\x95\xef\xbd\x01\xe0\xfb\xdf\xff\x9e\xb9\xfe\xfa\x1b\xf8\x83\x17\xf3\xdbn\xbf\x9dw\xbey\r\x00SW\xad\xfe\xe0O\xbe\xf7\xed\xab\xb7?\xff\x94]6\xd1\x00B\xa50\n\xb2\xb9Z\xcex\xe3\u01d8\xb7\xf2R|\xafhM\xe4\xc7\xe4U\xff\xb6\xa3\xb8\xf8\x0f\x96z\xec\xbf\x1c\u007f\x9fd\xca,u\xb0I\x13\xa2U\r\xae\x10(!\x927C\b\x81P\")\xa2\xd2\x11\xd6\xcc[\xc9\u0502\x8b\xb5\xa5\x95i/\x1dg\x8dj\x18\xf6Lb\xfcobC\x9d\x94\x83\xa3\x11\x15V6\xfd\x86\x8b\u0602
\xd020(/\xe2\x0f<\x83S68\xbe\xc1\xf1\f\xaa\xa4Qe\x8d\x1b\\\u03e0J\xd1\xe7\xf8\x06\x11\n\xa4\xb6Gi),.\xaeT\x06\xa5\xa2\"\x1eMz\xa4\x9a\x1c[\xcc\xd3p\u0558l+a\xa8\xba\u046a\xce\xfd\x11,kB\x9fl\xbe\x9e\x86\x89\x9d\x1c\xde\xf5\f#=\xdd\xf4\x1c\xd8\x05\xd9<\x13\x96\x9dC]\u01a1=\xa3c>\xee\xe4\xf5_(\xe7\xbe\x1fR*\x95\xf1}\x9b\xa6S[[\u01e3\x8f>\xc6W\xfe\xe5Zzz\xfb8\xf5\xc2+Y|\xf6\xbb)\x95}\xc2\xd0J[e\\4\x93\x81\x16\xa2\xf0a\xebA.\x1d\xf7\x84BN\n7O\xbb\x8f\uab24\xa6-
CM\xde\x0e\x18U\xa9\xac\x88<
\\\x89\xcc)\x9c\xac\xb4\xc5\u0704\xa0\xc3\xd8\xf1?\x81\x1bBm\xc64h\xa2\xca\u0714\x14\x1c\xa9\x81\x86\xda\x1a\xba\xf6o\xe4\xf8\xb1CH\xa9PJ\xa5,\x024\xc5B\x91\xfa\xfa\xfa\x17\x1e_\xbb\xf69\x80/\xfe\xc3\xdfs\xd9[\xde\u00b57|\x95\xd0\xf3\u0637o\x1f\xd7^{]\xd7\xfau\xeb~q\xf3\xf7\xbe\xf3p\x10\x84\xadm\xad\xad\vF\x87\x87\xc5\xf0\xf0\xb0\xad\x1d\xc8\xdf\xe8r\x1axe\x8e\xef\xd9\u0281\xa7\x1f\xa0o\xffN\xbc\xe1Ajjkhnk\xa7\xb6>\x8b+\x9d\xe8\xa0,*[a\xea\xd4n\x930\xad\xeaE9.\xf

9\xfa\fRJ\xb6>\xf2\x00w\xfd\xf3\xdf\xf0\xdc/\u007f\x9c|/\xa7\xba\x90\x8bi\u04e6\xb1f\xcd\xc5\xcf|\
xfa\u04df\xf9\xe3k\xae\xb9\xa6x\xf3\xcd\xff&V\xaf\xbe\xf0\x84[\xe7\xf7\x96&~\xff\x96\xefs\xe5\x15
W\xc4\xd8\xef\xacw~\xf0O\xfe\xfe\x99\xc7\xecN(\xe2\xc2\x1a\xb5\xdbR*\xc2\xc0c\xfa\x82\xf3\x98
\xbf\xf2R\x00\xc2
H\x82\x9b\xd3\xed\xa90'\x1a\xe0S\xa1\v\u007f#\x8a\x98,Y\xa9P\xd1\xd1\xd1\xc6JE\t\x1dq{\xa1O
TtTO\xa1\xa5gm#\u054dg\x10%\x83\xa8\xa9hD\xab\x95\"&\xd1Q\xfb\x1aF|\x83\x1f\rD\xe8\xcag\u
06db)\xf6\xb9\x88d\x83\x96\x17\xa8\x98\x19\xc5$\x95\xd4$#\xd2\xc9\xe4^<\x82\x1d\x9b9\xe8\x8
aBE\x88\xb8s\x16\t\x14$\xa5\xb4_\a\xfbzXZX$\x01\x1d:\xd6\xfd\xa6HRK2\xc9$?\u057a6ZH\xc5(\
x13\x195\xa5<\x9aSu@\x1b\x83\xe7\x01\x03CL\xecX\xc4\xd9\xef\xf98w_\xf7Q\xcaCC<q\xdb7\x9
94g1\xad\xef|7\xadY\xc1\x94\xac\x1fC\xb9'\xaf\xff\xc4\x15\x04\x01\xa5R\t\xdf\xf7\t\xb5\xa1\xae\xb
e\x8e\x1d;vr\xe3\x8d_g\u07fe\xbd\xccY\xba\x9a\xa5\xe7\\\x89\x91Y\x82p\xb4b\xb3:\xf6\x14\x9a\x
baO\xac~\xf9\xb7\x8bFM\x1a\x02\x14\xe0\xe6\x1459\x99D\bV\x11I\xba\xb2\a8\xae@d\x1ck\x8d\x
a1\x02d\x18\xcdFDr<%@iC\xe0[\x05\x97L\xddK\xe9\x1e!\xfe\xd5\xf7}\xea\x9b:\x980y\x0e\xfbv\xbf
H\x18\x068\x8e\x8a\x8a\xa6\x95g\x8e\x8e\x8e288\xf0>\xe0[\x00;w\xec\x00\xe0\x13\u007f\xf91^z\
xe9\x05\x96.]\x9e\xfc\\\xabW_\xf4\xbc1\xe6\x03?\xfc\xe1\x0f\xee\x986\xb5\xf3Ov\xec\xd8q\xe1K/
md`p0z\xbd\x94\x95-
kS%C\x1e\x1d\xe8g\xf3\xaf~\u02ae\xb5\xbfd\xca\xe2UL]v&\xd3V\x9c\u0174\xa5\xabh\x9a\xd0n\x
a5\x8ba5\xc1\x19\xf3ZRY\x88h\xa4o\x80\xfd\xcfmd\xf3Cw\xb3\xe1\x9e\xdb\x18Ly\xaf(%\xad\u07c
bMc\x13\xed\xed\xed\x9cz\xea\xaa\x1d\x1f\xfb\xd8G\xaf\x16B\xf4\x02,_\xbe\xfc7\xbeY\xbfw1\xff\x
ecg?\x9b\xfc\xf9/\xff\xe6\x93\x1fY\xff\xc8\x03\x13\x83r\xc9v\xb31;n\fR:\x84\xa1Gc\xeb\x14\x96\x9
e\xf3~\xeaZ&S*\x0e\xa7\n9't\xe7I\x896\x95\x9d\x9bTa\xac\x02\x95S9\xd8\xd1N\x82T.B(\x8c\b\xd
0\xc2&\x88W\xa6g\xac\xec\u0764\xc0\xe9\xf4\x10ME\xdc\x1f-
\xfc\x10(j\u06dd\xbb\"\x19T\"\xe9\xba\x05F\x1a\xfc\x10\x86}M94Q\x87\x1d\x15\xe8\x88\x0f\xb0\xb
0\x88\x88\xd2X\xe2\xc1'\x11'\u070d1\nK\xfdt&u\xbeHOC\xa7\xb8\x01i\xa2\xa2\x1du\xe2\x91O\x0
1B\xc5\x03I\xa6\xca@LK\x119\xe0\xc9dH\x84\xe8\x18\xac\x04H\xd7~M\x95z\xed\xb5\x16\x11\xa
9e\x12#}\x1d\xdd\xc1\xb1\t\u007fh
\xf0|\xc4h\x81\x05g^\xc6\xd1\x03[X\xfb\xbd/Q\x1c\xe8\xe5\x81o\xfc#\xed\x1dSi\xba\xf0tj\x95\xa6\
xd5\r#\xa2\xec\xe4\xf5\x1fI\x10+\x13\x9e!A\x10\x92\xaf\xa9\xa1\xaf\xb7\x8f\xaf\u007f\xfd[\xac[\xf7
8\xe3'\xcd\u4d0b\xaf\xa6\xbeu*\u00e3#v87\x9eGO\xcf\xc0GJ\xa6\xf4)V0\xb6q:\x91\xa3\xd2\x00\x
8e
[\xef\xe08\xb2bn\xf5\x9b\x98Y*>/d$\xb21\x834\x01\fht\xd1@`UR\xd2\x11v\xbe\u00c8\xe4\xebUZ\
x9e1\u007f3\x86\x92\x96L\x9e\xb1\x9c\x86\x97\x1e\xa2\xa7\xe7\b\x86\xac\u0154\xad/\x93)\x97\u
02e2\xab\xab{U\x10x3\x1d'\xb3\xe7\xc2\v/N\x9e\xda\u04a5\xcby\xe8\xa1_s\xe1\x85k0\xc6\xf0\xd
5\x1boPB\x88\"\xf0#c\xcc=\xd7_\u007f\xed%\x1d\x1d\x93\xfff\xf3\xe6\xad+\xf7\xec\xd9C\xa1X\xc
0D\xba}m*\n\xb5\xb8\xe1,\x17\v\xec~\xe61v?\xf3\x18M\x13:\x980k\x01\x93\xe7-
f\xd2\u0725\xb4M\x9bMMC3N&\x83#\x05\xbeW\xa68<\xc4\xc0\u04579\xb8\xe9Y\x0eo}\x91#;7\xd
1\u007f\xe4P\xd5=,\x95\xaa*\xe4mm\xe3X\xb6l\xf9\x81\xcb.\xbb\xec\x1d\x9d\x9d\u04f7\x00\xdcr\
xcb\xcd,]\xba\x82?x1\xff\xc2\x17>\xcf\xe7>\xf7y\x00\xee\xf8\u065do\xbb\xe6\v\xff\xe7O\xbb\x0e\x
1f\x8aX\xe1\xc8GA\x9b\x84\x95\x96B1\uf5372m\xc1j|\xafT\x95\x9c=\xd6M\xac\xaaS\x16\x86\x8a\
x9dd\x95\xa9C\xd5\xfa\xab\xea\u06a3\x91\xf5X/-
\x8cFF\xc1\xafFT\x8c\xf3c\uf4b8\xb3\x16\xc6Nr\u0274\xd66.\ua041\x92\xb6\xea\x96\xd8\x18\v\x8
3\xb1\xc1\xe4\xe8\xc0P.\x85\x84e\x8d\x1b\xe5%&\x98v\u0509\x93\x1e\x9f\xaf8\x83Uu\xb7\xc9\"\

x8e\xd4-
\x15[\x83\x94z\xc4\xc8H~)+\xd3{\bl\u53a4\x9a\xe67l|\xa2\x8a\u07cad\x8b\x95\xe1%G\br\x11w
\x85\xdd\x0f$\xc90\xaa\ud0a4$\bt4\x19K\x1c\x17i_\xcf\u8865\xc0+{\xe4\xca\x19\xcex\u04df\u04f3
{\a[\x1f\xbf\x8b\ue75b\xf8\xf9\xbf\xfc\x1d\xed\x13\xbeG\xc3\xf2Nra\x89\x1a\xa5\xf1\xf5\u0262\xfd\
xdb$\x88a\x18\xe2y>\x9e\x17\xe0\a\x01\xd9\\\x8e\xb2\xe7\xf3\x9d\xef\xdc\u011dw\xdeA\xbe\xb6\
x81\x95\x17}\x80\u0273O\xa5P\x18%\f\xc2\xc8\x02\xda$\xa6r\t\xe7\x92\xccW\xd8\xe1\x16Q\xa5\
x81H\xb9\xf3\x88\xd4L\x82\xb1\xfc\x89S\xa3\xc8\xd5\u02a8\xdb\x14c\uee0a>\xbcJuf\xacp@\xd6:
2\x18\x05\f[\x82\xc5\bc\xe7\xf7\xc4\x18W\xdc(\xfdb,\xd4\xe7\x05!\xed\x9dKhi\x9bB\xcf\xf1\xc3h\x
adQ\x11\xdf#\xa5\x12\x85b\x91}\xfb\xf6\xca{\xef\xbdw%\xb0g\xecky\xe1\x85k\xd2Ma\xf8\xf0\xc3\x
0f\xc9{\xee\xb9\xc7\x11B\x8c\x02?6\xc6<q\xe3\x8d7\xfc\xe5\x83\x0f>\xf4\xc1-
[\xb6\x8c\xeb\xee\xeeft\xb4\x80R\x8eA\"\xaa%\u0155\xaeo\xa0\xfb0\x03\u0747\u067e\xee\xd7\xe
4\xea\x1a\xa8mj!SSg\x9d\x1e#'X\xafT\xa484\xc0p\ufc6a\xf76&_\x85\xact\xe4Z\x87b\xe2\u0109\x
ac\\\xb9\xf2\xc0y\xe7\x9d{\xf9\xdb\xdf~\xc5\x16\x80\x1bn\xf8\xbf\xe2\x03\x1f\xf8\xd0o\xed{~\xafb\
xfe\xd9\xcf\xfe=\x9f\xfb\xdc\xe71\u01a8\xcb\xdf~\xf9\xe7\x0e\x1e\xd8_\x8b1HiI\x99x\xba3f\x8a;f\x
9f\xce\xf2\xf3\xaf\xc6\xcd\xd6Q*\xfe?\xf6\xde;\u0732\xab\xba\xf2\xfd\u0375\xf6>\xe7\xe6{+\xe7*\x
85RD9`\x81$\x04B
\x92\rn0m\f4B\xed6\xc16m\xbb\xbb\xdd\xee\xee\x87\xc1&\xf8\xb5\x03`c\x03\xc6
\x82\xdbX\xc6&H\xc6`L\v\t0
\x19$\x94\x91J\x15\xa5\n\xaa\\7\x9e\xb0\xf7^\xf3\xfd\xb1\xd6N\u772a\x92\x84x_\x9b\xea\xfa\xbe\
v\xa5[7\x9c\xb0\xf7Xs\x8e9\xe6\x18\xd3 \xa6>\xe7D\xfb\\\xc4J@\xef-
\xc6\xf5\xd8\xe3\xfe\x9a\xc3M\u0e6d\x05\xa2B\xff\xed\x01\xbd\xbe\xc8\xe3\x03\xa2\rX?\xc0\xc1\x
80D\xc6\xf7\x85N\x83U\xae\xcfN\x94\xc4y\x8b\u05eeC;J\xdaq\xd0u\f\x17\xaa\x01\xed\xd1[Ulu\xb
4\"&\xaa\x96FR\x1a^\xf9\r\u03bc\u0291\xca\xd6^\x0e\xe2\xb64+\xab\f\xa9\x8a\x90\t\xa9\x94\xf4\x
15\xb5\xa5\x0f\xec\xf0\xab\xda\xd2\xf0[\xa3\x00\x91\x85!\xeb\u0346\n]O\xb1\xecPv\bF\x15q\xc6o
\x1c\x86\xe9\xbd\x03\x92\xc4!\x89\ay\u007f0\x1a\xba\xd3\xf3LN\xad\xe6\x9a\u05fe\x83\xf9\xc3\xf
b\xd9q\xff\xb7\xd9|\xc7\xd7\xf9\xebw\xfd'\xec;\xfe\x18s\xf6j\xcen\xce\u04f4\x92\x1b\xf4\xfd\xdf?5j
%\xa3\xd3\xe9\x86\xc5
o\xb3\x9a$)7\xdc\xf0)>\xf2\x91\x8f\xd2M2\x9e\xfd\xfc\u007f\xc7Y?\xf5r2'dIRvn.\x18;Uc\f%l@\xda(
t\xcez\x94\xe1Hy\xdbe\x06h\n\xcd\u0248\xa8a\x8aJ\xba\x8fZ\xae\xeaj+\xffh\x8c\xc1X\x83\x0ey\xf
7M\xc4\xc1\xacCS\x83D\x95\x03!-\xfd\x8fLAe\xe6A\x18~\xc8\x19\x8d-
c\xdd)\xe7\xf1\xe8\xb6{H\x92\xa4\xb0\xa0\x15\xe3\x03\x8fgg\xe7\xe2\xef~\xf7\xf63\x9e\xc8k{\xf5\
xd5\xcfw@\xf7\xc3\x1f\xfe3y\xcb[~YE\xe41\xe0\xbft:\xad\x0f\xbe\xf3\x9d\xbf\xf3k\xb7\xdf~\xfb/n\u
0672u|\u03de\u0752e\xae\xdbh4bU\x95,\xcbJm|\xa5\x93\ah\xcf\xcd\u041e\x9b9\xee\xef6\xf9\xba\
u007f\xf8\u007fk\f\xaa\x9a\x1ac\xa2\xd3O?\x8d\xcb/\xbf\xfc\xa1\u05fd\xeeu\xaf\xba\xe2\x8a+\x1f\
x00\xf8\xe5_~\xab\xbc\xedm\xbf~\xcc\x06\xf6)\x81\xf9g?{#\xaf~\xf5\xcf=\x05\xff\xf0\x00\x00
\x00IDAT\x17\x15\xeb\u007f\xff\xef\xbf\xf5\xae\xfb\xef\u007f\xe0\xe2#\x87\x0e\xf9\x9a\xd1\xd6W
C\xb24alj\x05\xe7_\xf9:\x96\xad=\x9bvk\u059f\xe9Zsw\xaa\xd4\xdeR\xd7\xe4I\x8f\x1a\xb5\xee\xe
4\u0783\xe0R\xf3\xe9\xeeUhK\x18\x94\x92\x0fO(\xb7u\xb4R\xfbK\x84\x97QY\x81X\x8a\xe9\x9ev\
x80T\x90\x18\x9c\n\x92x\xdf\u7d2b\xb8\x14\xe2\xa0\x1c)x\xff\x8a\xefJ-
\xbe\xae\"\x0e\xad\xaa\x05j\xa6G\xf9\u07f5^\x9dKE\x8d\xa3\xbd\x0ex\xda\xeb\x8aW\xbfU\x9d\b\x
1a\xf9OD\x91!\x1a3\xd8E\rl\xecA\xdcJ]\x9aFO\ua560X\x04\u027c$\xcbW\xee\xe1\x86K\x95d.\xf5
V\xab\x99w\xa1\u031c2?=\xc3\xca\xf5\xe7\xf1\x82\u05fe\x93\x9b?\xf4\xab\xec{\xec!\xee\xfe\xa7\x

bf\xc3D\u00d8\xff\xf1\xfb\x8c\x9c\xb3\x92\x8d2\xef=\xe7\xff/~\x17\u007f\x92$\xa5\xd3\xe9\xd2MR
\xd2,%n4\xc82\xc7g>\xf37\xfc\xe9\x9f~\x88\xb9\xf9Y.\xbc\xfcg\xb9\xf0\xaa\xd7a\xa3a\x16\xe6\xe7
Kh\xd6z\x17V\\?\xb9\xa2\xc9D=\xb3\xa6\xfe2JC\xd7\xe6\"C<\x1a\xd1\x18\u036d\x9a\xf5\xb8\xb2\
x1b\xa1\xec\xa6ErE[\xe2\x1b\xc7q\xef\x85d\xd5\xf9\xeb\xcd:\\\xaad\x9a\x15\u02b2\xc2>BJ\x95[F
B\xd7\u016c>\xe5bFn\xbf\x89C\a\xf7\xe2\xe2\x18\x1b\xf9\xc8\xc94M1\xc6\xd0\xedv_\x04\xfc.\xc0
\xe6\u035b\u0638\xf1\xf4c>\u0737\xbc\xe5\x97U+\xf1\x8e\xcd\xe6\xf0\xa3\xc0o|\xf5\xab_\xf9\xc7/
|\xe1\v\xbf\xf3\xe0\x83?\xbcl\u01ce\x1d\x8d\x9d;w\xa1\xaai\x1c\xc7VUE\x8b<\xcez,\x9e\xcb\xdb\x
d8*u\x94\x0fw\x03x\x17\x15\xb9\xf7t\xd14M\xdd\xc4\xc4Dt\xca)\xa7p\xed\xb5/\xfc\xa7w\xbe\xf3\x
1d\xff%\x8a\x9a\x0f\x00\xfc\xfa\xaf\xff\x9a\xbc\xff\xfd\x1f8\xee\xad\xf1\x94t\xe6\xaf~\xf5\xcf\xf3\xc
1\x0f\xfe\xb1\x05\xb8\xed\xb6\xafo\xbc\xed\xb6o\xfe\u008e\x1d;\x00\x12k}]W<Q\xa7\x881\x9cv\xc
1\x8b9\xfd\xe2\x9f!\t\xc1\xc9%4\x94\x95\xb6\xe6\xe9\xd4\xf4xn\x86\x96O\xab\x8b\f\bN\x85L\ri\xfe
\xe1\f\xa9Z\x12\x8dH4\"U\x1b>\xf2\u007f\xb7t\x9d!q\x11\xa9\xb3\xa4\x1a\x91jLFL&M2\x19\"3C\x
a4v\x88\xcc\f\xe3\xec0\x1a\r\x03C\x18mbhb\xb5I\x94\u0158,\xc6h\x03#MT\x9b8i\x10\xd9!\xa2x\b\
x1b7\x89\xa2!\xa2\xa8\xe9?g\x87\x88l\x83\xd84\x88\x8cW\x93\u0628A\x145\x88L\x93H\xe2\xfa\a
\x11\x111\x111\x96\b+\x91W\xa4`\v};=\x1dM\xf5P\xa2\x92$$\xc5\x10\x95B\xd9`RG\x94)C\xaa4R
h\xa04\x9b\xc6W]\x91\xff\x90\xd8x)\xe6\x90A\x86\xa4\xf8\xbb\x19\xb2\xd8a\u007fs\u01e3\x16\x1
9\xb20d`\xd8\x12OD\x98\x89\x18\x9a\x96\xb0}L\x16\xfc\xa9\x17ff9\xe5\u072b\xb9\xfa\u07fe\x9d\
xc9%k\x00\xb8\xeb\xcb\u007f\xc9_\xbd\xe7\xbf\xf1\xad\a\xf7\xb1\u06cd\xfa\x83\xe2\x04/\xcd\xf3C
\xb3\x9b\xa4\xb4\xda\x1d_\x91\xa7)\x8dF\x13k-
7\xde\xf8\xb7\xfc\xd1\xfb>\xc0\xfe\x03\xfb8\xfb\xa2kx\xd6K\xde\xc2\xd0\xf8R\xda\v\xf3\xfe^s=\x90
\xac\xd5\x05\x1c\x0f\xaa\xc6D=\x86W\xd2W\xf8\x14\xdeXV0MCs\xccb\x1b&tj\xdaS0\xd5|i\x8bR\x
a3J\x8b\x16\xdar\r\x8a\x8e\u0448xQL<j\x89\"!\x8e\r6\xaa\xbb5*\xf5\xbd\v\x17\xac\x94'W\x9d\xc1\
xa2\xa5k|\xe4\x9a+}\xccm\x14177\u01d6\xcd[\xa6Tu\x1c8.\x90W)\x0f\x80O|\xe2\xe3\xc5\u7bbd\xf
6\xc5\xff\xf4\xe1\x0f\xff\xf9\xf3\xdf\xf0\x867\xfc\xf2e\x97]\xf6\xf7\xd7\\\xf3|\xce:\xeb\xccH\u0108s\x
ae\x16\xe9X\xfd\x88\xac\x8f\xce3Q\xe4\x95)q\x8c\x8d\xa2\xb0\xcdi\u02ea\xbc\xa4\x99e\xed\u06b
5\xf6\x9ak\xaeI\u007f\xf1\x17\xaf\u007f\xf7\xbb\xdf\xfd\xdek\xa3\xa8y/\xc0_\xfe\u59df\x10\x90?\xe
5\xca\xfc\x96[\xbef\x9e\xff\xfc\x17d\xaa*\xd7]\xf7\x86\xff\xbcy\xf3\x96\rI\xb7\x8b\xb16\x92@\xaf\x
14\xa7\x94KY\xb9\xe1<.\xb9\xfa?0<:\u51deG\xa5Gr\xefq)&\x91%\xaf[\xd1X\x8bbqD\xe2\x88\xc5\
xd10)V2b\x93\x11\x89\u00c8\xd6\x02\x9bJ&0\x18\u072b)\x04-
N\reR\xa1\x14\x87\x841\x06\x1b\x1b2\x1b\x87\x03#\"S\x13\xbe\x1e\xc8\xfc\xaa|\x92:\xba\xce\xeb
\xcbM\xa8\"L\xe5\xa7y\x99\xa3\x06\u04efj\xc5\x1c\x1e\x95\xabf\u06d6\xb7\xc0\xc0-
!\xa9\u01fdIM%\xd3\x1fP5pmT<\xb8G\x0eL\n\xa6\xed\x90\xf9\f\x86-
\xb5]\xfbJ[\x91\xbfv\u0554qS\x1dQW\xd8$\xdb\x14\u04a6\xf19\x92F\xc8DqN\xe8v:\xa0p\xcee\xaf
\xa4\xbb0\xc7W>\xf5_i\xcd\x1f\xe1\xce/\u007f\x12#\x86\xf8\x1d\xef\xe5\xa5\x17\xac`\x95i\xa3\xa9
\u00dd\x90@\ue2e0\x1c\u023b\xdd\x04\xe7\x1c\u0366\xcf\xc0\xfc\xf4\xa7\xff\x8a?\xf8\x83\xf7\xb
1o\xdf\xe3\x9c~\u0795\\\xfe\xd3ocb\xd9\x06\xda\xf3\xb3!\u078f~\x806\xa1\x1f\r\x1b\xd3&X`\xe4r\x
01A\x06z\x1a\xa9\x88\x1f\x90GBc\xd8\xf8\x83\xdb\x1cM\xf5R\xad\xc2{\xaf\xc6RY\x95\xdblHX\xa
e\x91\xa1\xc8\xdf\xdb\xf8`\x18\x13y\x03=\xa7\xa5*R+j\x10\x102\x97\x11\x0fO\xb2\xfa\xe4\xf3xt\x
eb\xbddY\xf09\x0f\x1d\xc0\xec\xdc\x1c\x87\x8e\x1c^\xf6\xe0\x83\xf7=\v\xf8\xa7'\xfb\x1e\xbc\xf1\x
8d\xff\x9eO|\xe2\xe3l\u07fe\x83\xdf\xf9\x9d\xdfED\x16\x80\x0f\x01\x1f\xfa\xc2\x17\xfe\xee\xd5_\xf
b\xda-
W\x9ey\xe6\x99/\u07bf\u007f\xff\xa9\x0f?\xbc\x89\xd9\xd9Y\xb2,+\ue0fc\xc2/\xabq)l\xa7\x15\xaf\n

\xf3n\x8f\x868\x8eY\xb6l\x19g\x9cq\x06\xeb\u05ef\xbb\xf95\xaf\xf9\xf9\x0f\\u\xd5\xf3n\xcd\x1f\xcb
?\xfe\xe3\x97y\u044b^\xf2\x84\x9b\xd5'\r\xe6w\xdf}\x17\xef~\xf7\xbb-
\xe0n\xb8\xe1c/\u07b1c\u01db\xf6\xef\u07c7\x18\xa3\"\xf5\xba\xcae\x19C\xa3S\\|\xf5/\xb2\xe2\xa
4\vH:\v~S\xaag\u02cb\xca\"\x8d\xf6L^\xacu4\xa3\x94\x86\xcd\x18\x96\x84\x86\xa4\fI\x97a\xe92b\
x13\x9a\x92\xd00)q\x0e\xe6&+9\xac\x82\xb7\xf6\v*V|[\xe7\x804\x04\x19f\xea\u056fF\\q\x11:\x04U
\x03\x91\xc5\x19K7\xb3$YL\xd7\u0174\\L[c:\x123\xa7\x11F\x84Fp\x8bS'\xb8\xcc\xe0\x9c\xff\xb9\x
19\xa6\x18h\xf6\xc4\x1e\xd6\x14]\xa6\xea'\xaeT\x92\xed\xeb\x19$\xd5\xd7E\n(\u05c1\xf3\x02\xa9
\xef\x9a\xd1\x176\x98\x17X\xa9C\xe6\x1dL(\f\x1b\xfa\xcd\xcd{\x84\xa1R\x0e\xba\x8cJ_a\x16\x19
C\xbbaH[Bd\xbd\x8aG\x9d7-
\xeb\xb6\xdaH\n\x17^\xf9z\xd2\xd6<_\xfb\xec\xef\u041a\x9f\xe6{\xffp\x03Y\xda!z\xe7\xef\xf1\xaa\x
cb\xd61f\xbb$\xdd\xf4\x84\x92,V\x81\xbc\x9d\x03\xb9*###\xb4Zm>\xfd\xe9\xbf\xe4\x83\x1f\xfc\x1
0\xfb\xf6\xede\xe39\xcf\xe2\xaaW\xfc'\x96\xac:\x9d\xf6\xfc\\!\xed\u04f0}\x9c\x83_!\xd5\x15?\x18\x
97P1J\xbf\\\xa0\xf6~kP99+D\rCc\xd4z7\xce\xda\xf5U\u05dd)\x83\xae\xd9z%\xa1H\xcfj\x82b\x9a\x
06\x9d\x8cQ#\x18u\x98Lp\xa9\xef\xe8\xb3\xf0\x18sS-
\x05\xd24\xa11<\u02aa\xd3/f\xe8\u06dfgnn\x16\xa7.\xb8\n\nY\x9a17;?~\xdbm\xdfX\xffT\u07cb7\x
be\xf1\xdf\xd7\xfe\xfb\x93\x9f\xbcA\xae\xbb\xeez\xfd\u065f}\xd5g\x81\xcf\xee\u0739\xfd\xec\x1bn\
xf8\xe4E\xe7\x9dw\xfe/\xec\u07bd\xeb\xca\xc7\x1e\xdb9\xb6g\xcfnZ\xad6Y\x96\x9bx\xd5\r\x05}\x
d5ni4b\xc6\xc6F9\xe5\x94SY\xbcx\xc9c\x13\x13\xe37?\xebY\xcf\xfa\xf6u\u05fd\xf1\x8bA]\xc3_\xff\
xf5g\u456f\xfcY\xfd\xe2\x17ozR\x8f\xfbI\x83\xf9[\xdf\xfa+\xf2\x9d\xef|'QU\xfb3?\xf3\xd3o\xbc\xf7\x
de\xfb\x82\xca\u0358\xdcs\xa5\xfa\x86\x9eu\xf1\xcfp\xe6\xc5?\x83s)i\x9a\x14\xff\x96\x05g5*\xa0d
Ei\u0614!\xd3e,\xea0\x16\xb7\x19\x8a\x12\x86\xe2\x84f\x94\x10\x9b\x8c\xa6&44#\x96\xd4W\xe1\
x15@s\xf9\v\x17\x92L\xf2\u007f1E\r\xa2\x95\\g\xc1\xa9\t\x16\x95a\x80n\xb4\xc2\x15\xe6\xaelJf\xf
d\xae|\x86\r\U0010d84b\xa1E\x8c\x03\xac\x962=\xd4\xcb\xf7<\xa5\x13\xd1\xca\x1a\xb4\xb3\x98D
-i\xc8\xd2L\xd4\xd0\xcd,\x1d\x17\xd3uQi\xf4U9\x80\xf2}Q#\xa5\x11W\u0756@\x8f\xcaWr\f-
~\x11\xc2\xe1r]\xa1@'C\u06e1:\xef\xfbV\xad5N\"\xa5}\xa3\x88\x86%\t-
\x00$\xb2~\x13\xb0\x13\xf2V%\x8fn\xf1V\xf6t:m\xd4)\x17=\uf348\x8d\xb9\xe5o\xdf\xc5\xdc\xf4>\x
ee\xfa\xea_\xf1\xfbs\u04d8\xdf{\x1f\xff\xf6\xca\xd3h6\"\u04a4M\xe6N\x1c
\xeftSZ\xed6\xddn\x02x\xbf\x95\xfd\xfb\xf7\xf3\x17\u007f\xf1q>\xf1\x89O23}\x98\xd3\u03bd\x9c\xe
7\xbc\xe27X\xbe\xee\x19\xb4\u06c1ZQ)\u02e3\xea\x06X^\x1d\xe6\x89Q6\xea3\x88\xeb\xab\x02B
I\xec\x8c@$D\u00c6h\xccz\x95\xabj\x0f\x0f_W\x95Uw\x1c{y\xf8\xde\rfo#\x10$\x89\rA',\x96\x98\x8
6\x80\xeb*\x9a*Y\xe6z\xe8y\xc1\x89\x92\u0186\x89u\x1bY\xbcj\x1d\xb3\x0f\xdd\xe7\x15&\x01\xcc
\xd34\xc5X\u04dc\x99\x99\xbd\x1c\xf8\u060f\xfa\xde|\xf4\xa3\x1f\xe1\xba\xeb\xae\xd7/}\xe9f\u067
au\xab}\xdb\xdb~-
[\xbb\xf6\xa4\a\x81\aU\xf5+7\xdc\xf0\xf1\xa5\xd3\xd3\xd3\xcf\u0777o\u07f3\xf6\xee\xdd{F\x9a\xa6
\x97\xee\u0673\xc7\xce\xcd\xcdfI\xe2+\xf6f\xb3!\xabV\xad2\u02d6-
\xdf\x1fE\xf6\xd6\x15+V\xdc\xdbj\xb5o\xbb\xf6\xda\x17\x1e\xb8\xea\xaa\xe7=\xfc\u044f\xfe\x05o|\
xe3\xf5\xbc\xeb]\xbf\x1b\xbd\xf5\xado\u0396,Y\xfe\x94\u0298'\x05\xe6\u007f\xf5W\u007f\xc9k_\xf
bz\x05x\xff\xfb\xdf\xf7+[\xb7n}\u0561C\x87\x101&\x0f0\xc8\u06dd,\xed\xb0b\xfdy\\\xf0\xdc724\xba
\x88n\xa7\x85SC\xa6~\xa8bP\x1a&e\xd8&LF\v\x8cGm\u01a2\x0e\xc36a\xc8vi\xda\x04k\x1d\x91u
\xc46\r^\xc9B\x949\xbf(\xa3y\xe8jYaD\x92o\\j\xe1\x87.\x95\xaaV+2-
\xff\xb5\xfe\x00)\x82\xe6*b\x13'%\xf4[\xc90Vi\x900\f\xa5\x99\xb7U\x0f\xb6A\xe1\x81+I\xbe\xdc0,\u
0448\xc4Y25t\u0552:K\xa2\x96\xc4YZY\x83\xb9\xb4\xc9|6D;\x8b\xe9d\x11\xed,\xa6\xed<\xf8g\xc

ex\x85Gx\xbd$\x1cL\xf5\x84\xf3\xa3\u0718\xb9t\xac\"\ub522\x82\xf2\xca\x112\xf5+\u0459\xc2B\x06\x13.\x80o\x0f\x87Z\x1d\xa6\xa9\x1fJ\x99\x1a\u0397z4\x01\x86\x87,:bI\xdb\xce\xfbT4(\xaas\x14\xba\x9d\x16\x99\x89\xb9\xe09\xff\x8e\xb81\u00ad\x9f{7\a\xf7n\xe3\x87\xdf\xfe\x12\xefz\xdb\f\x87\xfe\xeb\xefr\u077f\xb9\x8a\xa9\xc6\b\x92.\x90\xb9\xe3\xcf\xdc\xfe5\x03\xb9s\x8eN\xa7\uba55$\xa3\u0448i4\x1b\xfc\xf0\x87\x0f\xf1\xa1?\xfb\b_\xfc\xe2\x17I\x92.\xa7_p5W\xbe\xfc\xd7X\xbe\xf6,Z\xf3sE`H>0\xec\xe5\xbf\xf3hF\x9f\x16\x15\xd5b\x1a\xebB\x83\xf2\xfbT\x84\u0302\xb3\x105\x84x\xccb\x1af
\xaf\x9e\u007f\x97\xab\xca\x19\xe9\x8dB+7\xcb\nc6\xa9\u020d\xf3\an\x053f\xbd\xc4u.C\x13\x87I\xfd\x02\x8es\x81\x8e\x14\x1f0\xddM\x13\u2265\xac9\xf3Bv<|\x1fY\x96\x95\x1d\x87\x88v:]\u067am\xdb\xe8\xd3\xf1\xfe\xfc\xd2/\xbd\xb9x\xb9\xde\xf6\xb6_K\xb7l\xd9$\xdf\xfc\xe6\xb7\xe2\x1bo\xfc\xac\x84%\x9e\x83\xc0\u00ea\xfa\xc9\xe9\xe9\x83c\xb7\xdcr\xeb\xe4\x9dw\xde%{\xf7\xee\xd5V\xab\x8ds\x8e\xf1\xf1Q\xb9\xe0\x82\v\xe5\xf2\u02df\xdd9\xff\xfc\v\x0f\x06\xea\x86\xf7\xbe\xf7\xf7\x00x\xd9\xcb^\x1a\xfd\xc6o\xfc\xba\xbb\xfa\xeak\u04b7\xbf\xfd\xb7\x9f\xfa\xb5\xf4T\xbe\xe9;\xdf\xf9\xf6\xea\xf7\xbc\xe7=\xdf\xfd\xfa\xd7o]\xdfj\xb5\x82\xbfGXo\n\v(\xcd\xe11\x9e\xf7s\xef\xe0\x82\xe7^O\xbb\xe5\xe9\x15c\x1cC\xa6\xcbx\xd4f2n1\x15\xcf3\x1e\xb7\x18\xb3\x1d\x9a&\xc5\x18_\x89\x1aQR1\x18\xeb\u0273\xcc\x18\xc4(F\x15\x9b\xf9\xe0X_X\x8a\aS\xa9\f^\xd4ar}uu\xa8\xaau\xa6\u064a\xab\x0f\x12\xf3!k\xc9\x0ec$\x0f}\xf6\x15\xba\xcd\xd70\x8b\xc8(\xadK\xb2\xd4oFV\x01\xbd\x9e}(\xc1\xb6\xd6\xf3\xf2\x8a\x1f\xe0v3K\xe2\":.\xa2\xed\x1a,\xa413\xe9\b\x87\x92Q\x8e$\u00f4\xb2\x06Y\x18\xe4\xaa\xfaX-
\x11\x87\xad\xcc\x06t\xc0[YU\xee\xf7\xd6\xf4\xf9\xf3\x1a\x8a\rQ$\u0218\u016ci\u0090\xad\u0336\xf4(\x17J\xdd'\xa7\\\xf5\x0f\xfa
\x11\xba\xad\x8c\u03be.\xe9B\xe6_\xd1\x04\xa4\xebB\x86\xa8?\x15\xa2\xb8\xc9Ps\x84\xed\x0f\xde\u02ad\x9f\u007f\x0f\xdb~\xe8#\xb2&W\x9c\xca/\\\xff\x1b\xbc\xe9\xba\xd7q\xe6\x86\t\u0439\x90\x1a#?A
\xee/\x93$H\x0f\xfd\x8a\xbe\xf7$\a\xf8\xdf\xff\xfb\x16\xfe\xf4O?\xcc\x1dw\xdcA\xa39\u0139\x97\xbd\x8c\xcb^\xf4&\x16\xaf\xdaH{a\x864I\xcb\xf9\x8cJ\xb0G\xf6T_\xfe\x1e\xf8\u0160\xc8\xe7\xb7\u06a8\xef=T\xd5>\xabB\x17\x19\xd2\xd8+\xb8\x86'\"\x86\x975\xb09\xc52\xb0\x1b\xec\x8d%\xee'\xf7\xc2\xee>I\xbbE\x9a$a#\xdbk\xb6\x9d\xba\xc2\xd8G\x83\x8d\xb3.d\xb8\xd9\f\xd7\xf5\xd6\xd1Y\x9a\x87\x94\x85\x1fl-
c\xcb\xc7\xd9\xf3\xd0W\xb9\xe9\x0f\xff3\xdd4\xa3\xd1h\xfa\x19M\x96\xb1x\xf1\x14\xe7\x9fw\xce?~\xe9\xef\xbf\xf4b\x80Gwlc\xfd\x86\x93\u007f,\xef\xe3\xddw\xdf)\x17\\p\xf1\x93.5>\xf8\xc1?\x91\v/<_\xaf\xb8\u2aa7\xed\xb1<\xe1\xca\xfc\x86\x1b>&\xd7_\xff\x8b\n\xf0\xb9\xcf}\xee?\xdfs\u03fd\xeb[\xadV\xb12\xae\x85\x1e\xd9\xe0\\\xc2I\u7f00\x93/x9I\xeah\xd2fj\xa8\xcd\xe2\xc6,K\x9b3LF-
\x86l\x12\x86\x95.\x00k\xa9\xadQ#\x88U\xc4\xf8\xab\xd5\xe2\xbc;Cn\x8e\xdf\x13\u03e3\x02\x11\xce{P8oA\xa99\x8dS\xc4\xc7\xf5\xd8\xceV\xeb\x87\xf0W\x93[\u0245\x03B\xc2\u007f\x8bh\x1f4\xd6d\x81\x15\xfd\x9eV\xdd\xd2\\8\u07e8\xef\t\xe5\x81\u0222\x10\x89\xa3iRT\u06e1s\xf0|{\xa2\x1e\xd8\xe7\x92&3\xe90s\xe9\x10sY\x93\xb9\xb4I;k\xd0v1\x9d,&US\x1c\x80\xa6\x10\x97k\x11\x1a\xa1\x03\xd9K\x8a!T>\x84%QoY0\x94'\xa9K\xb1*\xcd1H\x1bz+\xf4\x9c2k\bQ\xd3\xe0:\x19i\nQ\xee@
\xe5\xe9\x99$\x1dP\xe5\xa4s\x9e\xcf\u02e6\x96s\xdb\xe7\xdf\xcb}\xdf\xfd\x1c\xd3{\xb7\xf0\x91?\xf8M\xb6\xfc\xf0>\xde\xfc\x1f~\x85\u02dfy6\xe3\xa3]\xa2(\xf5\xaf\u047f\xf2*]\x80,s\x85\x85m\xa7\xdbE\xc4099\u026e\u077b\xf9\xdc\xe7>\xcf'>\xf1)vl\xdb\xc6\u0112\x95\\z\xf5/p\xe1U\xbf@sd1\v\xb

3Gp\x9a\x15\xf5o\xd9\fJmP\x98\x87q\x8b\x8d\x91\x82'\x1f
!\xac\x8cR\xd4\b\xce\xfa\xfd\x83F\xc3\xd0\x18\xb3\u0626\xfch@N)f\x10c\xbc\xfcT\f\x98P\u03c7\x
a7\xa2&l\x83\n\u0208/\b\x99\xf1\xdeDe&y\u0634\xce2\\\n\xe3+Nb\u046a\xb5\xec\u067e\x05\xd5\
x06\x1a\xf6Ifff\x19\x19\x19=OU\x9f+\"\xb7\xed\u06ff\xff\xc7\x16r\xf5T\x80\x1c\xe0W\u007f\xf5mO\
xfb\xe3yB`\xfe\x8ew\xbc\x9d\x1c\u023f\xf1\x8d[/|\xf7\xbb\xdf\xf3\xeb\xfb\x0f\x1c\xf0\xb8k\xad\x94\
v\xbfB\x9a&,Zy:\x17_\xf5Z6,\x1fa4\xd9\u02b2\xb1y\x964f\x19\xb1\x1db\x93\x16tAUl/a\xffW\x83\x0
e\xdaX-
\x92uL\xa5as\x01\xd0\x05%B1\xeaC[\xadj\xa1V\x91Z\xb1\xec\xe5q`\u020aX\xaa\xd2\tP\x04\xacq
~\x81\x06\x10\xe3j\x1b\xa5\xf5\x14\x94\x92{\xd7\xdeK\xba\a\xd9D\xf0{\xf0\xa6\xde\xd2JE>\xe6zt\
x04&\x1cj\x16\xa1I\xc68m\x966\xfc\xefs\b\xa9\x8bh\xb9\x98\xf9\xb4\xc9L2\u0311d\x84\x83\xddQ\
xa6\x93\x11f\xd2a\xba\xce\x06.\u0495@-=\x83\xd7\xea^Gx-
\x9c\x06\xd7\xc5N\x99cU\x8clU\x06\xc8\u0434\xf4\xb3\xd1j\xd6G\xa9\u007f\x17\x03v\xc8`\xe6\x8
dW\xa78\x17|B\xea`\x90\xa6]\x16\xe6\xa6Y\xb2\xfa\x19\xbc\xe8\xf5\u007f\xc8\u0112\xb5|\xff\x96\
x8f\xd3i\xcd\xf1O_\xfc\b\x9b\x1f\xfa\x01\xaf\u007f\xfd\xaf\xf0\xaaW\xbc\x82\x8d\xa7\x8e\x11\xd9\
x0e\xe0\x82g\u01bf\xcej\xbc\x9b\xa6t:\t\u076e7\xc3\x1a\x19\x1e%u)\xb7\xde\xf6\r>\xf5\xa9Os\xe
b\u05ff\xce\xdc\xfc<\xabO>\x9b\x9f\xba\xf6\xdfs\xe6%/\x05c\x99_\x98\r\x87\xb4)\xfc{\xaa\U00019
6e0i\xd0u[\x1bC\u0413\xf7\x8f\xb4\xc3\x15]\xd9\x15r\xd6\a\x9a\x98Hh\x8cx\xae\\\xe4h\xbar\xa9\u
02e6D\x06\xb6\xfcZQn\xe5v\xc99\xe5Y\xec^h\xe9\xcb+\xaa\x9e\xff\x1e\x01\xe3\xac\x1f\xce\xe7a(
\xb9U\x84\x13\xba\xf3\x1d\xa2\xc6\x14K\u059d\u02ae-
\x9bp\xea0j\x8a\x8e\xe3\xf1=\x8f\xa7\xb7|\xfd\x96\x04\u0ef7\xdfqB\f\u045f\x10\x98\xaf\\\xb9\xb2\x
f8\xfb\x8d7\xfe\xcd\xff\xb8\xe7\xde\xfb\xe8\xb4\xdb\x18\x13\x891\xd6k=\xf1-
N\xdc\x1c\xe6\xaa+\xae\u19df\xb9\x8e){?\xa3#s4M\xeaU&5\x11S\xd8N\xab,Gj\x0e\x80\xd6S\ty\x
a5\xaa\x05\xf0\xf9*\xbdAF\xa4Y\xc1K\xe7\xa1\xce5hT\u03e7;\xf1\xc6R\xb9g\x9b\xa0\x18\xeb)\x1f
c\xb4\xcf\x1d\xdfU\x00\\\x06\xd0\f2\x803\xa4F\xd8\xf4\xaf\xcfk\xb5\x1f\xc8}\xb0r\xbe>\x1f\xd4\xf6\
x82l\xe0\xfb%\f\x1a\xad8\x1a\x921\x12uX\x1c\u03e1\u00de\xa2ie1G\x92\x11\x0et\xc69\xd4\x1d\x
e5p2\xcaL2\u0302k\x92\xfa\x8d$\xff3\xf2\xdf]\t\xf3\xcc\xfd\xd4s\xcb\x01M\xfc\xf2\x86D\xd5\v\xa3
W-
S\xbe\x83Z\x1b\x8c\xf6v,\x826\xbdf]\xbb>\xcb\u0569\xab\xf8h\x97\xff\x9fe)\vsG\x18\x1a[\xc6U\xaf
\xfc\u007fX\xb1\xfe\\\xbe\xfb\xe5?a\xcf\xf6{\xd9\xfa\xd0\x1d\xfc\xbf\xefy\x98\x1f\xdc\xf9\x1d\xae{\u
00db\xb8\xe2\xd9\xe735\tQ\xdc\xc5e\u067f\x1aP/\xd4*\xdd\xc4\aF8%\x8ab\xe2\x86a\u04e6M|\xfe
\xf3_\u099bnb\xeb\x96\xcd4\x1aM\u03be\xe8\x1a\x9e\xf9\xa2_d\xf5)\x97\xd0I\xbat\xdb\v!\u4efa8\
x96'?yE\x94\xe6\xaa\x151\x9e\xfa\xb4qE\xef]\x16'\x94S\xa2R\x8d\x12xr\f4\x86\f\xf1D\xe4\xe7'z\x
ac\x81{\x8f4\xaa\xe7_\xb4RIh\uf234\x92\xf9\xe9\xfa\x0e\b\xdf6\x9aQ\xeb\xbfo&-
\xae\xad\x00\xd7d\x9d.\xcd\xc6\x18\xe3\x8b\xd7x\u05be\x90\x05B\x96e\xb4;\xed\xc9\xfb\xee\xb9
g)\xc0\xc2\xfc\xfc\t\x11?\xfb\x84\xc0\xfc-
o\xf9e\x00>\xf5\xa9\x1b\x9e\xfd\x81\x0f\xfc\u0275\xd3G\x8e\x14Uy\u0357A3\xce\u06b8\x91\u05ff
\xe8RN\x9f\xd8K\xb7\xbd\x00\xa2\x85\xf6\x1b\xb4\x00\xe5\xfc\xe22\x81\u05b0\xe2<?nK\xf5\xb4C
H\x89\xfc\xe0\x93\x94X\x1d\x11\x19\u05b9P\xadKm\u04ed\xf8\xbe\xe2\x1a3E5\xefr\xb3\a\xa3\xb
e\xfa69p+\xbdK8ZQn\xf4zO\u0531\u007f\xf0\x05^\r>.\x979Cwa*\xdby\xf9\x9c2\x1f\xd4\x16^'Z?-
\x02m\xa4\xd4Z\x00\xac8\u01a3\x0e\x13Q\x9b\xb5\u00c7\u9e88\u0664\xc9t2\xc2\xc1d\x9c\xc3\x
c9(\xd3\xc90\u04e9\xe7\xddS5\x98bh\xac^d\x92{\xe78\xc5%\x0e\xd7\xcdhD\xb6\xbc\xf1e\xf0-
\x9c\x1bj\u55a6\xd5\n>\xb7\xa0qF\xc8b?0\xd5pP\x9az\xc0^e\x96\xa1\xb4\x17f\xb0\x8d\x06\xe7\\\

xf1\x1a\x96o8\x9b\xef}\xf5\xc3<\xf0\xdd/\xd0^8\xc2\u035f\xff0\x0f>p\a?\xfd\u04ef\xe5U\xaf\xfc\xb7
<\xe3\xec5L\x8cCd\x13\xd24)d\x8c\xff'\x02{1\xe4\xec\xa6d\x99\xa3\xd1h\x80\b\xbbw\xed\xe6K\xff
\xf0e\xfe\xee\xef\xfe\x8e\xbb\u007f\xf0\x03\x92\xa4\u02d2\xe5k9\u7c97s\ue56fat\xf1:\xe6\x17\xe6
H\xb3^gQ-
=u\xb4\xea\xc1\x13\xb6\vm\xe4\xf5\xe4\x95\xd9F\xff\x9et\xa5\xbf\x14p\xd6x)b$4\xc6\"\xec\xb0=\x
06\xbdBeVt\xb4\xc9\xdb\xd1.\x9c:\xfc\xe7y\xb9\xb9\xec\xd5I\xe9\x0e\x8a\x11\u0730A[\x02]\x17\x
16\x87$\u0200S\";\xc5\xf8\xc4j\xaf3\x0f[\x98\xde\xee
\u0262\xc8N\xce\xce\xcd]\x04\xdct\xe8\u0421\x13b\x15\xed\xb8`\xfe\xe8\xa3[e\xfd\xfaSTU\u01ef\
xbf\xfe\xba\xdf\u06f1\xe3\u0471N\xa7\x83\xb1\x91\xe4\u0546\x15%IS\xc6\xc6\xc6\xf87/\xb8\x8as
NZL\xb7;\x1f\xb8\xdc|PG\x9f\xc9S\xfe\x9e[\x93\xf9ac\x14\xacV\xd5W\xe5\xa2\x10k\x86\xa8\xd2$\
xf5\x03H\xadT\xf6\xf4.\xd6\xe4K:e\xb5\xab\x95\u0163\xd4\x18\x1f\xcc\n\xc1Y\xd9W\xfa&\x1c2B\x
a9\x98\xf5\xa0W\x17ZQY\u0411ZUr\x94J]\x06\xb2\x1b>\xb1\ajUj\xde=\xa3G\xd1\r\u0200\x9bD\xa
9t-
B\u04e44\x9b\tK\x9a\xf3\xacs\x87ie1sY\x93\xc3\xc9\x18\xfb:\x13\xec\xedLp\xb8;B'\x8b\x83\n\xcd\
x0f\x9fr\x1c\xeev\x1c\xdar4\x86\xad\x9fW\x1c\xe5\xa8R\xf2<R)X\xfa\x1e\xcfb\x9cS\xba\n\xed\x8a
\x0f\xbaV;\xea*\xf0V\x82\xb0\xb3\xa4K+KY\xb2\xee<\xaey\xfd\xffd\xfdYWp\xe7\xd7>\u03ae-
w\xb1\xf9\xe1\xbb\xf8\x93-\xf7\xf3\x8d\u06fe\xc4+^\xfe:^\xfc\xe2\x97p\xc6\xe9+\x19\x1f\x8f\x81
}\r-
\xf9\xff)\x1a\xf5\x1c\xc8U\x85F\xa3I7\xe9\xb2o\xdf\x01n\xbd\xf56>\xfb7\x9f\u5bbb\xee\xe4\xf0\xa1
\x03\x8cM,\xe6\x94s.\xe7\xfc+_\xc3\xda3\x9f\rv\x88\xd9\xd9#%h\x86*E\\X\xe8\x91\xea\xec#\f\xf9\
xc5`L\x8c\xb1q\t\xfeG\xad\xa9+\n\x94\xbc*\x0f\x16\xb7v\xd4\x06\xa7\u0363T\xe5\xc7RR\x95\u00
e9Z\xdbVv\xa6%}Z\xfdr\xcde\xc5Z\xce{2\xf5\xb4+\r\x03\xad\xb0\xf2\x9f/\xe08\x05\xb5\x8cN\xae`d
l\u048b,\xac\xb7\xad\xb5\xc6\xea\xbe}\xfb\u067d{\xd7Ya\x89'{\xfc\xf1\x9d\xb2r\xe5\u069f\xe8\xea\
xfc\xb8\xe1\x14\x87\x0e\x1d\x92{\uee57ukW\xbf\xf5+_\xf9\xc77m\u0672\x05D4\x8a\xe2B\xfb\xe1\
u0375\x84k.\xbf\x8c\xb7\xbc\u654c\fY\u04b4\x8b1\x1a\\XC\n\x8f)\xe5\u007f\x0e\xc1\x18%6\x19\x
d6(\x1aU<J|\x19\x87qJ\xac\x19\rR\xa2\x8a*]\xfaj\xe9\n\x81\x93\x0fS\f!2\xcd\xff\xfe\xd4X\xba\x12\
x05\u0661\xb7\xb5,@\xbf\xc7\xd5@\x82\xda%\a\xf7\x9a[a\x8fk\xa3\xf4\fc\a\u04075\x80\x97\x92
\xaf{\xb5\x14\x9f\xa3\x12\xa6[\xba\xddIe\u05f3\xe8^\xab\x19\x9cU\xffu|74d\x13&\xa26K\x1b\xb3\x
ac\x18\x9afys\x86\u0278\xe5\xb5\xf3\xea\xe5\x92Y\x98%d\xa2\xa4\x06\x1aC\x96\xe11\xbf\f5\xa8
\x12\u02c1\xdci\x19D[\xb8_\x86\x8f\xd4\xc1\\\u2f27{\xaa\xd8\xc4\xdb\xff\xdaL\xa9\xe4\x83\x15\xa
f\u007fm\x19)\xbc\x17Y\x92\x107\xc7X\xb3\xf1\"6\x9e\u007f\r#\u32d9=\xb4\x9b\xb9\xe9}\xec\u06
7d\x9d\u007f\xfe\xf6\u05f9\xe3_\xfe\x85\x83\x87Z4\x9aKh4\u0188\x1bC\xbej\u0574P8\xfd\xffm\u0
765!(!_\xf7\x8e\xa2\x18E8tx\x9a\xcd[\xb6\xf0\xa5/}\x99\xf7\xbd\xef\x8f\xf9\xccg\xfe\x9a\a\ueed7\x
cc\xc1\xba\xd3.\u44ab\xdf\xc0\xa5/|\x13K\u059dG\xe2\xa0\xd3i\xd7\xfd\xf4\xb52\xac4e\x88\x89\t\
x96\u01de#o`\xa2\x86\xf7A\xe9\xd3\x1d\u02601\xb8_\x0e\x8a\fY\uc8c1\xecT\x84\x1d\xb3D\x12t\
xe0\x03G\xdf\u04bf\"\xd4\x13\x01V\xa3(\xf3\xeb#\xcb|\u05a6Vh\x91\xdc\u05e4\xd2\xc1z\xe7^\xc5
9\x9f^\xe5:J\x9a(i\x16\xb6\x8dss\xaaF\x03\xb26{6\xdf\xc9t\x9e\r\n\x18+,,\xcc\xcb\xe2\u014b\xcds\
x9f\xfb\x9co\xfd\xd1\x1f\xbd\u007f\xef\x99g\x9ein\xbe\xf9\xef\u007f\xa2\xc1\xfc\x98\x95\xf9=\xf7\x
dee\xce?\xef\"\xa7\xaa+^\xfb\xda\u05fcv\xd3#\x8f\x00\xa4\x91\x8d\"#\xa5F4u\x8e\x95K\x97\xf2\xb
3/|\x1eK\x17\x8d3\xbb\xd0\xc2\x19A\xa2\xfcb\xcc#\u01f4\xc8\xef\xb3\x01p\x8c\xd12q'\x84X\x9a\\
?\x1bj\xbfc\xd7\x18T\x94$\xa59P\x0ejF\xb5\xa0\f\x9a\x95\x9b\xbc!i_\xcb_\xe4\x1d\xf6r\xe5\xa2\x0
3\xf4\xb3\x14\x1e1\u054d8\xe4\xe8\xecb\x1f\xb4\x84\xc7K\xdfQ\x12n^W\xf1\xcf\xea\xe9>\xd4T~\

u007f@\x11\xa9\x01\nEt\x9e\x11e\xd4v\x18\xb1\x1dV4f8i\xe4\x00\xfb:\x13\xeciO\xf1x2\xc9\x117
BK#T-MBu4
\x06(\u007f\\\xf9AX\x030\xf17g\xea`\xbe\xeb\x98K\x1c\x99z\xb32\x97o\xf0U\xed\x84+s\x93j\xadX
0\xbb\xaat\x17\xe6\u0212\x06\xa3S\xeb\xb8\xfc\xe5\xbf\u0269\xe7]\xcd}\xdf\xfe,\x9b\xee\xfa*\xfb
wm\xe2\aw\xde\xc2=w\u007f\x9b\x9bnz&\u03fd\xea%\\\xfa\xcc\xcb9\xed\xb4SY\xbf~\x19\x13c\x0
6c\xbb\x88\xa4\x15J@~L\x00^\x1a.\x19c\x18\x1a\x1a\x06,Y\x96\xb0\xe3\xb1\u01f8\xf7\x9e\xfb\xf
9\xd6?\u007f\x87[o\xfb\x06[6oaf\xe60\xd6X\xd6m\xbc\x80\x8d\xe7^\xc9i\x97\xbc\x94\u0255\xa7\x
93e\x8eVk\xde\a\xaaH\x91A\x86:\xa9\f7)\xe8\x95\"\xcd&O\f\n\x15y\xfd\xaa\xae+\xb1*\x11.\xbe\x8
32!\xe3\xd5\bi\x03h\n\xc3F\x8e\xbe\x96\xa6u\xb6\xbd\xa0\xe3BA\x93\xef\u007f(up\xef\xe3\r{7P\x
b5\xfc.\r\xee\xa5\xdd\f:\xed\f\xd7J\x91\xd4\xfb\xf6D\x95\x94\xaf\xa4\xdbab\xd1*\x16\xafX\u03ceG\
xeeF\x1b\xc38\x1cVD:\x9d\x0es\xf3\xf3g\xddx\xe3gO\x03\xee\u0773\xe7q\v?\xd9.\x11G\x05\xf3\x
bb\xbe\xf7\x1d\xce?\u03db\xa0\xbf\xff}\u007f\xf4s\x0f=\xf4\xf0%\xd3G\xa6\xb1\x91\xb5\xd6V\xc7\
x16\x8a5\xc2\xf3\x9eu1\x97\x9c{&I\x96\x90\xe2\u021a\x96\xc8xm\xb6\x84\x05\x1d\u07d3\xfbIvL\
x1aIUC\x85\x1b<\xbf\xab\xc0j\x8e\x9aI_\xb7\xe9\xf4\xd5x/\x05Q\xfaN\xf8'\x9aa\xaa\xd5v\x0f`\x0f\
xfa|\xfd'j\xcd8_+\u0740\x1c\x85)\xd4c3\x88\xc7\xe9\xcfC\x84\x9b\x96R\xc1\xa2\a\x19\x14\x84\x9
a\x83\xb7\xeb}\xd4\xf5\xd7+2)K\xe3Y\x16\xc7sl\x189\xc8\xe1t\x84=\xdd)v&\x8b8d\xa7\x10\x1d#u\
x06\x1bUm\x05J\xb8uZ\x06S\xf7V\xa4\xdd\f\u6e8e\x85\u0115C\u047cC\xa2\xb6@\xd838\xa5\xb
7\xe7.Z\x98,Ih\xa5G\x88\xa2\x06\xcb\xd7_\xc4\v6\x9c\u03f9\x97\xbf\x92\a\xef\xf8{\xb6\xdc\xfbuv
m\xbd\x87{~\xf0M\xee\xf9\xc1wX\xb5\xfa$\u03bf\xf0r\xce;\xf7\x12\xce:\xf3,\xce:\xfbTN9e5##\x16#
)\u05b8\xe0_\xdf;\xfdx\xe2CT\xe9y\u074d\x91\x90\x16\x1f\x85\u7532c\xc7c<\xf4\xf0#\xdcs\u03fd\
xdc~\xc7\x1d|\xf7\xbbwp\xe0\xc0\x01\u04b4K\xdc\x18b\u0769\x17p\u0499?\xc5\x19\x17\xbe\x80\
xa5\xeb\xcf!q\x86v\xbbE\x16\xf4\u05e5o\x8f\x94iV\xb9@\xc0\x14\xe1o\x14f\xc8&p\xe4b*Wf=.\xb1
\x8f\xeeSBU.>\f\xdcB:Ii\xc6&\x80\x82\u053f\xa3G~:\x88:\xaf\u0280k\x15y\xf86c,N\xd2\xda04\xf7
W\xd7\xca\xf6j\u6105\xaec\xae\xe3H\xda\x19\xcd\xc4\xd1\u0410tU\x19\u00a7i\u00a2\xa9\xa5,]\x
b1\xbev\xb5\x87\x82\xdf\x1d>|\xc8l\u07fe\xfd,\x80\xb7\xbf\xfd\xb7\u007f\xe2w\x89\x8f\n\xe6\xf7\x
dds\x8fg\xe9TW\xbe\xeeu\xbf\xf0\x86M\x0fo\xf2\a\xb9\x11\xbf\xb6\x1f\x02bS\xa7\x9c\xba~-
/\xbf\xf6y\x8c\x8e\x8f0\xdf\xed\xfaTxc0\xa4Xu\x1e\xc8\xc9\x17\x19\x94H\x9d\xaf\xbe\xc3\xe2\x8f8
\u05e3
\x19D\xcaI\xed\xfa*+\xf0\xc1\x15pu\xa2.(1Ye\xff\x91\x01\xb6\x9f\xfd\x9f\x91~8,\xe0\xfb\xa9\x9a\x
b5>\xe9\xe6_\xeaC&=\xc6\xd7iX\xcb\x14W\xef\x12\xea\xc9Ee\x18\u0148m3\x1auX\u079c\xe1T\x
b7\x97\x83,\xe2\x80]\xc6\xfet\x193\xf1\x04\xa9D\xde/#\xf0\x94N\x198Ps\xaa\xb4S\xc7\\W\xe9dZ?
g\x8c\xa0qE\x02\xa7\xfd\xd4XM\u03a95c\x91B\xf5\x9a&]\xb2,\xc1\u0688\x95'\xff\x14\xabO\xbe\x
84\xf3\x9e\xfdsl}\xe0\x9bl\xbe\xfbk<\xb6\xf9\xfb\xec\u067d\x99=\xbb7\xf3\xb5\xaf\xfe\rk\u05de\xca
\xc6\xd3\xce\u1b33\xce\xe3\xb4\xd3O\xe3\xb4\xd3N\xe5\xe4\x93\u05b1b\xf9$\xe3c\x11qD!\xe9)\x
aaU\xc9\x13e\xfay\xef\xf29\x19\xeaf\xa3\x8e\x03\a\x0e\xb0e\xcbV\x1e~x\x13\x0f\xfe\xf0!\x1ex\xf0
\x87\xdc{\xef\xfd\xec\u07bd\x8b,\u02c8\xacedl\x82U'\x9d\xcb\xfa\xd3/c\xdd\xc6KY\xbc\xeat\xc4\xc
6,\xccuH\xb3\xcc\x1fzZ\xd3y\x96\x9d\x9b\x04\xaf\xe1<K\x17A4\xb8\xf4\x05\xf3\xaczoS\xbf\x92u\x
c0b\x99\xb7\xb6\xf5FZ*\x905\rCc\x11c\xb1\xc1\x86\u074a>\xab\xba|o\xa1\xd7\xee!7d\xab\xf8{\x9
b\x9e\xd3Z\xfcp\xccW\x02N\xea\x93\xd0\u028f\xcaTi%\x8e\xd9VF\xe2\xbcL7\xf76\xd7\xfc\x1a\xcf
\xd4o\x8a\x8a\"&f\xf1\u04b5\x8c\x8c\x8c\x91d)Q\x14\xf99\x9e\xb5\xb2g\xcf\xe3\xcc\xcf\xcf]\xa4\x
aa\x93\"2}B\x82\xf9\xe6\a\xefd\xe3\xd9^\f\xffG\u007f\xf0{/\u007f\xf0\xfe\a.\x99\x9b\x9f\xc7\x18cLe
m\x1f\x81\xd8Z^\xf2\xfc+9\xfb\xec\xd3h\xa7]2\vj\fq\x90\x0fVR\x12|\xc6e\x05\xc8sS)\xe9\xc9\xc3\x

e9\xa3\"jU\xb8\x1c\x15\t\xb5oPW\xd5>k\x1f\x84\xf4\xd6\u294b\x9c\x1es\u03a3\x95Fq\x90\x19m/\xe4\x1d\xebsOk\xe3_\xf0\xa9Z\x13\xc4\x14\x80\xdes\xf3\xe4\xaf|$\x8e\xa9x\x81)\xd3f\x9d\x1cb:y\x9c\xddf9\xbb\xa3\xe5\x1c1\x93$\xc4\xc1\xa3\xd9U\xba%\x0fz\x9dLYH\xfd\r\x98\xb8\xf2F\xae\xfe*'\x12\u031b\U0008391e\xd4R\x02\x98\xf6(\x93\xa4^\x14:%u\t\xe9\xcc462,]s6K\u05dc\xc5i\xe7_\xc3\xe3\x8f\xde\xc7\xceG\xfe\x85\xc7\x1e\xbe\x83\x03\x8foe\xe7\xceG\u0631\xfd\x01n\xfb\xfa\x17X\xb2l\x15\xabVm`\xf5\xeau\xacY\xb3\x9aSN\xde\xc0\xa9\x1b\u05f3j\xd5r\x96-]\xc6\xc4\xe4\x14C\xcd!\xe2\xc8\xd2lZ\x1a\x8d\xa8\x06\u058a\xd2\xedt\x99\x9d\x9d\xe5\xe0\xa1C\xec\xdf\u007f\x80\u077bw\xb3}\xdb\x0ev\xec\xd8\u03ae\xdd{\u063a4\xa7\n\x00\x00\x00IDAT\xbe}\a;v<\u0291\xc3\a\x01\x88\x1bC\f\r\x8f0\xb9d5kO\xbf\x84u\x1b/e\xf9\u06b3\x18_\xb4\x864\x83v\xbb\x85s\xed\xf2\xc2\xcez{\xa0\xb0OQ\x84/\u7b85\x06\xa3>\xecA\x8c\xc5`\a\\Ez\x9c^\xd1'e\x15Uy\xc3\u0418\x88\x19i\x1a\x1a\x86b~Q\xab\xf0\x85\xbay\u0500\x8a\x9c\xaa\xc1\x9dT\xa6\xdc9\xa0\x1b\x1f%\xa9d\x15)n\u0391\xfb\xfflg0\x9fx\xa5U\x1e\xfc,\x15\xea\xd0\x05\xff\xfc\\\xff\x9e\xa4)K\x96\xaebjj)\xbb\xf7\xee&\x8a\xe2B\xf2\xdciw\xd8\xf9\u062eE\x0f<p\xdf(pb\x82\xf9\x1f\xbd\xff\xcf\xfc\ub9fa\xec\xf5\xaf\xf9\xb9_zx\u04e6\xd0R\x9aR\vm\x84$I\xb8\xe0\x19g\xf2\xfc\xe7=\x8bh\xb8\xc9\xec\xfc\x02\xc6X\xef`\x18\x8c8\\x\xb7\x8d\v\x1f\xd5w[{\xe5}\x955\x9d\x9a\x87\xb9<%\xd4\xd3\xda\u0630\xce\x16\u66d6\x1a8e\x82l\xce\xf4if\xfb\x97\x84\xfao\x1b)\x8e\x05\xcf\xf2\xe7\xbf\xd3T\xf4\x1f!7T\xa4/\x00\x8f\x1f\x85\x8e9\n=\xa3\x15\x11x\x11y\xaa\xbd\xa7\x99T\xdad\xdf^\x8fH\x87\x11\xf6\xb3\xa4{\x84\r\xe9n\xf6D\xcby\u052cd\x0f\x8bHL\x8c\xa8\u00d2\xe1\x1c\xb4\x12e.qt\xb3\xa0\x00\xaa{y\x15\xb0\xa4\x86\xb0\x02*a1\xacR\x98\xd5\\\xf4\x82\xb9Y\xcf\xeb\xa3Tl\x11\xf2\xf4\x9b\xd41\xd7=\x82\x901>\xb5\x8a\u0265\xeb9\xe9\xcc\u02d9\xbdb7\x87\x1e\xdf\u009em\xf7\xb0{\xfb=\x1c\u07bb\x8d\xb9\xf9i\xee\xbf\xffv\xee\xf9\xc17\x111\x8c\x8eN2\xb5h1\x93\x93\x13L\x8cO2>1\xc5\xc8\xc8\x18CCC\f\x0f7\x19\x1e\x8ei\u013e%hw\xba\xb4[m\xe6\x17\x16\x98\x99\x99\xe6\xf0\xe1ifff8r\xf80\a\x0f\x1c$I;\xa1\x03\x89\x19\x19\x1bg\xe9\xea\rL,Y\u0272\x93\xceb\xd5)\xe7\xb3x\u0169LL\xacfhh\x8a4I\x99_\x98\xf7^#b\xe8[\xe6\xd1r\x10\x9e\xf3\xc2B\x1e}\xe8kT\xa3\x12\x92\x82\x82\xb7\xbd\x1bTvT\x87\x90\xd2s\x9d\x86n\xcaz\x19\xb0\xb3\x06;\x161<j\x89\r\x03\xc9\x19\xed)D\xa4\xd2(\xeb\x00&T\xfb.1-\xa6QbL\xbd\xab\u00ebVP?8og\x1ab\xe0\fY\xa6\xc5,\xa8\xba\xbfP\xca\u05c5,I\x19\x1d\x9dbdd\x1cuY9\x93Q\u03db/Y\xba\xe49;w\xee<\x03\xd8}B\x82\xf9\x87\xff\xe2\x06\x05\xf8\xf8\x87\xde\xff\xb3\xf7\xdd\xff\xc0E\v\xadv%\x1d#\xb4[\xaa\x8c\f\r\xf1\xe2k\xae\xe0\xd4\xd3Nf\xb6\xdd%\x8d\x9aEE.\xb9\xfdN\x00q\xa9\x82\xcb\xc0\xab\xa0\xa2Q\t\x91f\x88<A\u042e\x03B\xee\u0218\x03\xa7\xe9\x19\xd6(\x90`\xc9B\x8c\x9b%\xf3v\x005S^z\xa6\xf2U\xc0\xaev\x98\xf9W\x94\xc0\x9d\x8fl{)\x19\x11*|{\xbd\t\ueb45\x06\xc5\x06\u0213\x04\xf5\xe2\x97:-\x8d\xb0\xb4\xda\u0754RN1AQ\x122\x15\x1b$,\xcb\x0e\xb3\xc8\u0370F\xf6\xf0\x98]\xc1\x0e\xbb\x9a=Lq$\x8bh'>X\xd89\xad\x9d\xb5\x03;\x19)\xa3O\xeb<uYa\x15\xaa\x8d\xda\u0560\x95\xe3\xbd\\\xac\xd2B\xf5\xe0_\xf3vk\x014\xc3F1KV\x9c\xc6\xe2\x95\x1b\xd9p\xd6\xe5t\u06f3\xcc\x1c\xdc\xc5\xc1\xc77sh\xcff\x0e\xef\xdd\xc6\xec\x91},\xcc\x1dazv\x96\xfd\a\xf6\x93t;\xb8,\xe1\xe8A\r\xbdG\xae\xc16\x87h4\x87\x18]\xba\x9c\xa1\xd1qF'\x97\xb0x\u0769,Zw*\xcbN>\x83\xc5\xcb\xd6\u04b0c\x183\x02\xa9\x90\u0336\x98\x999\x12\xb6\x1eM\xbd\tQ_\xfbK\xd1\u0494\x85\x8c\xe4\xe1\xaa\xc1,\xcb\xe2o\f\ttK\r1u\x90\xba\x8b~\xff\x15\xbc\x141\vC\xcfh\xc4\u041c\x88\x88\xad\x10\x99\x90\xe5J\x9f\xfbN%\x01\xb1'\xc36\xbf7\xf2\x9c\xdf\xda\x1c\xaaz\xb5W\x12w\x8cx\x0f\x19-\xfdvT\x95N\xeaH3\xafV\xc9[\x15\x13\xd4m\u04ab\xce\t\x85\xa1K\x1c\xcd\xc6\x18\xf1\xd0\b\xe41\x95\xe1\tdiF\x96ef\u07fe}\x8b9\x01\xfe\xf4\x81\xf9G>\xfc\xa7\xf2\xe6\xb7\xfc\x8a\xaaj\xf3-

\xff\xe1\x8d\xffq\xeb\xb6G\x01\xb0\u0592+X\xac1t\xbb\t\x17^\xf4\f\xaey\u03a5\x18\x11:\xce\x12\x19\xa1\x11\xf4\xe0y\x80\xb1uZ;\xb2{\x17_T\xca1e1\xc8|\x92
\x9e\xff=\xc3\xd0\xc1\x1b\xdeG\xb8\x02\xa0\xb5OR\xe5\a\xb3\xb6\x10\xc3y_\x13\xd3\xf3\x95\xda\xc3<\xf6\x91%\xa1\xa2w\b]\xe2`1\x90\xd5\x06\xb7\xdeU\xc6\x06=\xbb\x1b
\xab\xa4\u03d6T)Ez\x83\x8e<\x8e\xf3\xb9~0\x95\xb0*-
>\x86\xab\xfao\x96\xc2\xc9N\x8d\x9fi\x94\u06f4J\xa4\tK\xdda&\xb3Y\xd6\xcbn\xb6\xb2\x8c\xfb\xb2\x95<\xea\xa6\xe8h\x03+\xfe9\x17G\x9f\xd4+\xcd\x1c\xc4]8\xefr\x83\xaf\x1c\x99\xb52\xf8\x10\xad\xce?\xfdB\x99\u04de\xbc\x8c\x9e'\x9dyS\x11\xefS\x9fet\x16\xe6\xc0\x18\xa2x\x98\xf1\xc9q\u01a6V\xb3b\xc3\x05\xa0\t.Kh\xcf\x1fa\xe6\xe0.f\x0f\xef\xa55\u007f\x88\x85\xb9\xc3,\xcc\x1e\xa6\u04de'\xe9\xb6H\xba\x9d`\xd5\xec\x13\xb2\xc4\x1aL#\"j6i\f\x8d0<1\xc5\xc8\xe2e\x8c,Z\xc2\xd8\xe2%L.[\xc5\xf8\x92\x15\xc4\u00e3\xa8X$\x8a0\b\xe9L\x87\xee\x91\x16\xddVB\x9af%\u0329+\x02\u0335\xb7\v5R\xef\xf5\x82<\u0408\u0146\x94)\x0ft\xe1\xea\xecuZ\u8ece\xa4\xcfD_E\xc8bCf\r\u0450\xa1\xb9(&j\n6\xa75\xa8[\xd9\xf6\xa1\xb9\xd6\xff.}\x95\xbfV\x92\x8d\x06\\\x9ba\x05Z\xabn\xbd\b]\a\xad\xb44\xb6\u00c8\xb7\x98\u02028\"\xffy\xc1\x8c&\x97j\xaa\u02c8\x1ac\u010d\xd1zL\x1b\xbe\xf8\u073b\xf7q\xee\xbf\xff\xbe\xe4\x84\x04\xf37\xbf\xe5W\x14\xe0\xbd\xef~\xe7\xd9\u07fd\xe3{g/\xb4Z\x1e\xccM\t-\xce)\xe3c#<\xff\x8aK\u0670~-
\xfb\xe7\x12\x8c14Ih\x90\x16~)\xb5\xe4\x13\xd5\xfa\u025a\v\xb2\xf3\x05\b\xe9+\x04\x8e\tR\xdaC\xa5\xa4\x18\xba\x1a\xd1\x15KLV,\x03U\u056b\xd6\xd7@\b\x10\x91\xfa\xea\xbdpI\xac\xfb\x0f\xea\x00~{\xf0c\xf2\xdfms\x97\xa0\xa2\xa1\x84|\xe3\xd5\xd4l\xbd$\xac\xda\x1c\x9bb\xe9\r\x0e\xd0\x01\xdc\xfc\x13\x16\xdb\xe5\xb3\x06\x13\xd2\xd2+\u06e7\xb9'Mi;\xa0\xb5\xda\u0285\x1e\xcbi\xc6X6\xcb9\u0332\x81\xddl\x91\xe5\xfcPV\xf3\x98.f\x8e!\x9f\xfc\xa4Y9\xbb\xec\rL\x92j\x958\xe0Y\xea`2\xab\x8c\x9c\xd4\x1e\x85F\xf8y\xe1\u07camT\u007f\x81\x92t\xdat\xdb\xfe\xf7Yk\xbdEj\x1c3\xbax\x94\xd1EkX\x1dN\a\t\x8f\xcb9\xc5e)Y\xda%s\xa9/\x88\u0160V\x90\xe1\x88hx\x88\xa8a!\xb6d\xc6'*EQ\xa9\xc5vY\x86f\x19Y\xda&I3\x9cq\u0210b3AS!u\x8afZ\fX=X\a\xf0\n\x87h\xe9l,\x15P\xb3X\x13\x87\xcc\xce*\xb0J\x9f\xda\xe8h\u05d4V\xba\x1cg\x84\xcc\nQShN\xc5D#\x16#\x14`\x9eW\u0245+\xa8\xf6\xf0\xe3U\xfd~\xcf$\\z\x93+zd\x04B\x99PU}/3\xf5@\x9e\vy\\\x00s#\x82\xc9*\u0770\n5\a\x8e\xd0\xf1;bT\"\u007f\x8dT\x8bG\x81\xf9\xf9y\xf6\xed\xdb\xcf\t\a\xe6\x9f\xfa\xd4'y\xc3\x1b\xae\x03`\xdf\xfeCo\u07f9k\x0f\x99sXk\xc3p#7\xd3J9\xf7\u0333\xb8\xe6\x8aK\xe8\xa6\xfev\x1f\r@nU}U^\xbb\x93\xfbU)\x1a\x8c\xf0\x9d\x19\xdc\xdc\x1e\xab\x12\xad\x02\x9c\v@\x9e\xe1}\x1c\"\xf5qryU<\b`\a\xc1q\x17S9\x00\\52\xf9\xa8\xdd@\xfe\x15&\xa8eJ\xbe\xbc\x94n\xda\xf0Y\xa5\xee\x8dq\xac\x01\u0560\xdf*\xa1\xf3\xc8B\x0f!\xc1\xa3f\xb0\xf9\xed\x80?\x86P\x05k\x1f\x87Z\x85J-
8o_5\xbb\xa0\xf9\xd5J\x8b=)m.\xe6QNc/[X\xce&V\xb2C\x970-
\xc3\x18\xbcZ\xa96\u0516\xea{\x96k\u0635\xae\x8d\xcf\x1f\x912\x80n\xa1\xd8\x1bp\x85\x9e\u065fD\xe2\xfc\x01\xed\r\xd5\xf2MBW\xbe\x8eNq\x9a\xe0\\\x82K\xc2s\x0e\x89\xee\xd6HI\xbd\t~\xf1&\x1e\xf6[\u00a2\xb8,|}*\xe8\x02tZ\t4\x12\x88\xc0\x841o\xea\x1c\xear%\x87\x16^1j\x05F-
\x91\xf5\x1d\x86\x11p]G\x9a\xfa\x9f\xab\xf9i*ZZ4Ky\xb5J\x11\xb5fJ\x1f\xf2Z\xf4\x9f\x1eC\u07a4\x03\x8a\x1e\xff\xebR\v\xb6!4\xc6-
f\xcc\xebD\x82\x1d\x12=\xfbp\xa5\xc3\xf31\xab+\xad_\xd7rl\u05af\xa6p\x14A\x9d\x1f\xa2w\xb3\u04b7\u0204=\nu`\\\t\xfe\xf9\x82b!\xcc\x14\x83:o\xbcU\xdd\xf8-
\xe7EB\x969\x92$=\xf1\xc0<\a\xf2n\xb7\xb5\xf2\x85/|\xd1U\v\v\v\x9e\xc26\"\xa5\xb63crb\x8c\xe7_y)\xeb\u05eeb\xba\xd5a\x98\x94\u0607\xa4\x15Ugq\xba\x17\xc5q\xb8\tM~sW\xe4jO`\x188\xc8\xcaU+cF!\u00c8\x90\x15\xd3\xfdrE\xc2\xf4\xd4\a\xf9\x1a\xba\xf6\x1aF\r\xf0_9\xba\u0250\xf4\x18\x8

7U\xb74\xb5\xa7&\xa9\xd7\xd7R\xfb}G;&\x06\xcf\x04<\xa5\xe3\x87a\x16\r\x1d\xc1\x13\x1c\x9e\xf6\xe53\xd6o\x00\x97?\xa2\xf0\xc68u=L\xb2G\xda|\xc0;N\x87\v\xe41Nc\x1f\x8f\xcab6\xebr\xb6\xb1\x8c\x83\x8c\u1430\xb5\xab\xc5T\xa0\\t:\u01aax\xcd\u0295\x92N\b\ufa69\xe4\xa9J\xde\xd9i\xd9z\xbbJ&i^i\xbaj>\xa67\xf6\xf0\xfbiR/2\x8b\xdf\x19\xac\x94]\xde\xdew\x02\xfdg\x05\xed(f\u0220\x8d\x88,\xb6%\x90\xe7\x86i\xb9\xd41\u007fZC\x06\x91\x88H|\xe1\"\x89\xa0]G\x96iPrI)I\fT\x8b\x02V\xbc$\u03c6\xf7z\xe0\xcaE\x0f\xf8\xd6\x05\x05\xfd.Bi\bgn\x8cZ\xcch\x14\xf6;\x14c\xfd\xe1\xe1(3l\xa5\x17\x82\aJ\xcc\xfa\x03(\x8ev\xe7\x94~\xa3\xf5qY\xa6\xd0I=\xfdc\xaa\xc3U\xc1W\xe5\x15*\xa7W\xb5\x9c?\xc6j\u982a\xb5\xd7
\xcb2\xba\xdd\u0389\x05\xe6_\xfb\xdaWy\xc1\v\xae\x05\xe0\x03\x1f\xf8\x93\x9f\u06fcy\xcbD\xab\xdd\xc6Z#\xa5\x1cQ\u025c\xe3\u0513\xd6p\xcd\x15\x97x\xbe\xca%\x05\xdf\\\xb1\xaa\xa6w\x9b\xba\xba\xf4p\xd4\xd3Z\xfb\xa9\x96\xa3\xa1\x92\x14f\\\x8a\xadU\aYm\x01Hj\xff&\x94A\xf5R\x8c-%\xf0\xe7\x1c\x0fd\x8eR\xf5T\x0f\n\x83\x1bx\xf0\xc8\xc0q\xef\x13\u05ea\xfb\x0e\xc0\xd1\b\x9d@\xb9\xb2e\xca\x03\xf4GT\xc3\x14\xac\xc51\xd46\xd5\xe5\xa9\xfc&\x1a\xa1\xcbY\xec\xe6$9\xc0>&\xd8\xc626\xb1\x82\xdd:EBDD\x86\x13Wn/\xf6\r\x97{\xdfs\xad\xad\xc5he\xde\"aP\x9aW\xebN\r\xc6e\x85iZ\xd1\f\xaa\x92U\xf0E2-
\xbeW\xaby\xabT6X\x1d\xe5\x82VX\x1d\xaf\xea\xe3\x9d\xf3\x9f\x88\x1a\x82f\x12\x12\x90\xca\x1fT|\xbd\xd6\xdfmi\b:i\xb1\r\xc1\xb4\x14\x8d\x1cY'C\xb3\xde!q(\xe1\xadE\xd5\xfa\u047d5\x85\x84\xb3\xda9\xf5]\x1c\x85LE\xfa.YU\xbc\xb5\xed\x90!\x1e\x8f0\xe3\x11&\xf2\a\\^\xfa\xb8^)kM\xf6$\xf4\xa6Ni/\xf1%\xbdc\xd3\xd2\ufa38\xb6D+\xc0\xeb?:\x99zI\xab\x8a\x1f\xc2W\xc0>\xe7\xcb{'W*\x82q\x14IK\xfe\xec\x1d@A\x1a\xc19\xf5A\x1e'\x12\x98\xe7@\x0ep\xf7\xddw\xbf<I\xba\x11\xa1\x95)NQ\xe7h6c\xae\xb8\xe4\\\u05af^J\x96t=?.=\x9b\x93=\xc3\x1c\xe5\tJ\v\x9f0\x12i%\xb2\xd0\x15\xaa\x06\x1b\xfe\xe6z\xb7\xd7\xfa\x86\x8d\xbd\x8a\x95\xfa\xb0P\xf3)z\x1fD\xeaQ\xf9\xfb\xaa;I\xef\xe8g\x10\x90?\x19\xe0\x95>@w=u\xfe\x8f\xcf\xe1S\x8e\t\xea\x9e\xffw\xc1\xad\xa9!)'q\x805r\x98\xb3\xd8\xcdfV\xf0CV\xb1\x87I\xbab\xc1@S\xd2\u00b7\x9d\x819IZToZM\xf2\x93\xbaJ\xdf
\x181\x1e\x8d4D\t\x86.E\xb5b\xbd\xaa\xe5
Zz\xd0H\xa4N\x03i1\xa9\xf5\xc3\xe2\xda\x01\x13\x02ql\xe4\xaf!\x97\xa9\xa7\x00l.\xbf\u0492\xcf\x12WZ\xb4\u57cf\f\x8c\xfa\xb9E,\x8aSC')\xb70\xfc\xf4\u0440\xf5\xb4J\xbe0=h\x06Y\xbamV\x1e\x9f\xe4\x8b5\x15\\\x95\x12\xdf\u0350\x10OX\xccT\x04\xc1#\xdf\x1a_\r\xbbL\xc3\xfc@\n\xb37=j\x9aEO)S\xbc\x9cR(\u0574\x06\xe8\x15\xe1\x83+\xfd\xcd\xf3\x85\xc3v\xea\x818\u03fa\xcd\v4\u3d28\xcc\xcb\x01\xb9T\xde3\u03df#~\x86\xe7\xdc\x13\x13J\x9cP\x03\u043d{w\xaf\xbb\xe6\x9a\x17\x9e|\xf0\xe0!O\xb1\x88H^\x95\xbb\u0331~\xfdj^\xf2\xbcg\u04b0\x86\xb9N\xb7\x06\xe4\xbdn\x88G\x83\xab'\x10K|||\xaf-
\t\xd6o\xd5^H5}U\u07e0){\xce\xc3\x1e\x8dq<\xca\x05\"\xe5\xf0\xae\x94n\x1d\xfd\xfb\xe4\x18\xc7\u00d3;\xef\xb4\xe6\xb5\xf1\xe3\x02\xf2\xe3\x01:\xa1S(\x80\x03\xbf0\xb6\x86#,\x939\xced7\xdb\xec2\x1e\x8eW\xb2?\x9d
IlH+\xcfz\x16e\xfa_#Wl\x83j\x9d&\ntJ\xae\xc216\xf6\xab\xe2\xaeLF\xf7\x9c\xbf\xab\r\u07b5\x8aR=\xd9\u01c5\t\\@\xedb\xd2R\xb1\x18\xf0\u0671\xa6\xb4X\x90\xd2\x18>7\x8e\xd2\xde\"\"\xb7f%$h\x8d\b\u0608\x86q\xd0vtSj\x9d\xa3*H\xe6\x02p\x95\xa7\x99\f\xaa\x9b\xfb\xf60*:\xfe\xd0\x01;\x01\xd304\xa6\"\u0322\x18m\x94\x12!U\x0f\xa8\x05\u0154SX\"\xd8 U\xad\xf2\xe5\xf9
\xba\x06\xd0E\xaeD\xa9\xae\xa9\x96Py\xc5o\xf2\xb9G\xe8\x96R\xa0\x9d:\x12\xa7\xc5<$\xa7]\x14\xc1\xa6\x0e\xeb\xf2\x83\xd8o5\x9b\xe0GcB\xf9\xae\x94[\x1cZ\x9f\x0e\xf4w\xfc'\n\x98\xdfy\xe7\xf7\xb8\xf8\xe2K\x01\xf8\xf4\xa7\xff\xd7\xcb\xe6\xe7\xe7\u05e5i\x8a\x91\"\xaf\x06uJ\x14E\\\xfc\x8c

\xd3\u0638a\rY\x96\xf9\xb0\x01\xaaaaz\\\x84>\x9a\xdb\u02a0RU\x9e
\xd8\xf4%\x81\x0fPz\xc8q`\xb3\x1aKE/\x1f\x9fK\xf7\xd0J\xa8CUc\xab\xc1\x97\x9dB\xae\xd5ov4\x
b8B\u007f\xd2M\xc9q\xe6\n\xfa4\xfc\xcc'W\xa1SL\xcb\xf2\xae\xc7\xf9\t\x06\x06\xc7R\xe6Xd\xe69
ed\x1f[\xcdr\xb6\xe8\n\xf6\xbbq\xda.\xc6\xc7g\xb8\xa3\xf4?A\xf7\xa3e\x17&Z\xe5\xfd\xb5R\xd6z\x
f9\x9eD\x06\xa31\xaa.|(\xea\xb2|\xc70\xd0$\x01\xb0\xabUs1q\xd5\u06ba\x8d\xa1\n\xfcBd\xfc\xd0
TB\xe0I\xf1\xb5\xd2S\xa9R\x82{\xfd\x00\x0f\xb2\x91\x91\b\x89\x95\u01bc\u00f4\x1c\u074e\x06P\
xcd/.\t\x83\xe0
&\xb0G\xa9\x8e+\x06s\x85\xc2\xc5J\xe1u\xae\x026\x16\x1a\xe3\x11vq\x03\x86\xbc\x99]\xeec\x8f
\xfa\xeb62\x82-
\xb67\u00dd\xac\xa1\xeb\t|\xa9Sj\x96\x0eE'\x13T8\x92\x1fD\x94A%U;\u072a\xa7\x8a\"\xdeD+\v\x
11\x8f\x15jJ\xc5\xcf+I\xe2\x15q*\xf9\xe6x\xa0\\\x8c\x0f\x06\xafJ\t\x9cs\x14\x9cU\x8f\xd8\xc6\x1aK
\u0708O\x1c0\u03c1\x1c`\u01ce\x1d\xcf\x1111\xd0\x11#\xcd\xfc\xc6r\xea\x18\x1d\x1e\xe6\x99\x1
7\x9c\xc5\xc8P\x83v7\xa9\xe7\a\xa2\xc7<\x05\xf5I\xa0\u0493\x01!\ud66bs\x94\x15\xfbz\x04\x9c\x
d6\xf3@\xc5`m\x8c\x8db\xd4Z\u007f\xc3\xe6\n\t\x97\x91e.\xe8\xeb\xfd\r\x1eE\x9682a\x131#I\x13
\xb2\xa4\xeb\xdd\xee\xe85\xa8\xd2\xe3\xd6\xd0Oy1\xe88\xac\xbe\xf6\t\x1c\x9f\x9eJ]\x8f\xfa\xde\x
f5\x1fay\u03f4\xc4\xcc35\xbc\x83S\xe2\xfdIm,g\xd3\xec*\x0e\u034f\x92\xba\x88X\xb2\x01A\bU\xa
9\xa2T\xdc\xfa\xaa\U000b6cbc.L#\xf2\xb4\x1d|0\xb5\x1a\x87\xe2\xc1]\x8c\v\xc3M\x17\xec\x85u\x
c0u+}v\xc2\x1a\xacQ\xac\t\x8bs\"X\xe3\xd7\xe9\u0542f\x0e\u056c\xa0A\\\x95\xc8\x13jFi\x84\x8a\
xdb4\r\x1a\x19\x88\x1d:\x9f\xe1\xda>\xd72\x1f\x14kF\xcd`\xabj\u00ec\x15]zAg\x86\xf0\x12\r.\x95*
\u07b1\xb21\x12\xa8\x95\xe1\xf0\xaa\x99\xca\xd05,\xa2J\x1e~R\xe4\xf8\xfa\u1dc4\xa1\x83J\x91
RW\xd0R\xd2+=\xad\xddo\x95\xc0\x89\n\xaek\xb8\x17\x12\f\v\xa9w\xd8\xcc\xd7\xea\xf2
\x18\xf0rD*4\xb78\xf1\xea\x9d<0]\xa4\xa6\xfc\u0272\x94,K\xfb:e\x14\x8c54N$0\u07fau\xb3\x9cr\x
caFU\xd5E/z\u044b\xd6\xed\u06b5\u02ff\xa7\x95\x8aC\x15\x16OMp\xc6\xc6\xf5\x18\x1b\x93d\u0
770\xf03\b6\x8e\x0f\xe0G\x03\x86\x1f\x1d\u0334\xe6\xb7Ru\x13\xf1'\xb5?\xad\x8d5DQ\x8c\xb1\x
11\x9dn\xc2\xdeC3\xec;x\x84#\xd33\xcc\xcc\xce177\xcf\xfcB\x9b\x85v\x9bN'\xf1\x12M\x11\x8c\x1
1\x86\x87\x1aX#\u0111e\u0572\u015c\xb4n\r\xa7lX\xcd\xe2\xc9q\xb2$!M|\xa4Y\xear\xdd\xeb\xb1
\x0f\xa2\xbal\xf1\xe9\xeb\t\x95r'O\x8a[\xe9\xe9\x01u}\x92\a\x80\vJ\x94%\xf1\x1c\x8b\xc6\xe7Y\x1
b\x1fbs\xb4\x82ms\xcb8\xd2\x1d\xf1\x8e\x8e\x15W\xcbz\x180T\xa7\xa0J\xff\xa0</\x12\xfb\x1e\x9
b\bF\x82\xf9\x12\n6\x97\x0f*\xaaY\x00\xfa\\1S\xfa\xad\xd7|Hr:'\xaf\u028dE\xa2\x18i4\x10\xabh\x9
a\"\x89\x16\xdd@_R\xf2\x806P\x03\xdfnF\rqC0-
Cw!#\xed\xf8\u007fs\x86RSm(*\xd4\xe2\x8f\xf1&Y\u0396\ubd5a[\x17\x1b\x0f\xe4\xcd!C4\x19!\xa3
\xb6b\xb5\xe0\x03`Ll\x8a\b:\xa5\xaa\u007f\x0f3T\xa7%HW<\x91\u051f\x85d\xc1\xee\xd8U\xc1<T\
u84bf\x8f\xe2A$2\xa5\n)Ue!\x81$\x1cj\x1a\xba\x03\t\xd6\u02a0H\xea\xe7\x11\xfe\xd0\xf1\x15z>\x
9d\x17\xcdu\xf9\xf9\x90\u04d0t[\xa4\xddN\xe9\xe3RI%n\xc41\xa3\xa3\xa3'\x0e\x98\u007f\xe7;\xdf
\x05`\xf3\xe6M\xe7\x1c:th}\xbb\xdd\xc6\x18cD\xeaV}g\x9e\xba\x9e\x95\u02d7\x929\x17\xde\u013
2*\x10\xd1\xe3\x80\xf8`A\u078f\x02\u40f2~\xf2P4W\xf9\x00\xc1F\x96(\x8a\x88\xacg\xd1\xf7\x1d<\
xcc#[\x1e\u246d\x8f\xf2\u062e\xc7\xd9\xfd\xf8~\x0e\x1c>\xc2\xfc\xdc\x02s\xf3-
f\xe6\xe6\x98_h\xe1\xb2\uc60fa|l\x94\x95\u02d7\xb2v\xf5rN;e\x03\x97\x9e\u007f\x16\x97]|.\x8b\x
a6&1Y\niB\xb7\xdb\r9\xa9\f\xf0y\x91\x01\xa0\xaeO\x19p\xab\xf3\x02W1\x1b+f\a\x05%\xf2\xa3u\x
02\xf2\x14\xe8\xc8\xfc\xbd0(\xab\xe2#,Y4\u01c6\xe6\x016\u03ef`{k)s\xc9P\xe0\xa5]9H\x13\x1d\x9
cC\xa9\xd4\U000a628d]\xa9S%9x\x16\xda\x18\x01c\xf3OG~\a@\u0577\xf7\xf9\x81\xe1\xb4F\x91
\xe4\xbc-

\xc6gk\x9a8\xc2\xc41\xa6\u0440\x18\xe8&\x1e\xc8]\xeaW\xca+A)\xc5;*u7I\xadX\xdcJl\xb0\x9104
dH\xdb\x0e\xed(\x9a\b\xa6k\x82[b}8_X\v\x9bJi,e\xc5j\x8d0\xdc\xf49\x9ef<\xf2\x86]\xf9\xb0\x90R6\
\\x04\x82x\xb5fy\xd0T\xa8DU%\r\xbe)i\x00\xfa,\xfc\xb7+R\x81\xca*\xdd\u4389\x15\xf9\xa0\x15\xc
1\x86\xa7\x91*\xdes\xa5\xe2K\xae\x95\xca\x1d\x85(\xf5\x03\u043c\v\xab\x9a\xf2\xf5\xe8\x9d1b\x
e8\xb6\xe7\xe9v\x17JW\u01fcCSel|\x9cU\xabV\x9b\x13\x883\xff\xbe\x00z\xf3\xcd7\xaf\x9c\x99\x9
9^\x1c^\x8cZ\xf3\x1cG\x96\x8b\xce=\x83\xa9\xc9\t\xda\u0764\xa2\x1a\xad@\xb4\x0e\x8eR\xeb\x
d7w<\xb5\xe1\xdf\xf1i\x96R.\x97/\xd84bK\xb3\xd9\xf4i/G\xa6\xb9\xff\x81M\xdc\xfe\xfd{xx\xf3v\xf6\
xec\xdd\u03c1\x83\x87i\xb5;\x81:\x89\xe9&yjzV\x02p~\xe1U\u03efp\xa1\xcf\xce\xcd3;7\xcf#[wp\x
eb?\u007f\x8f\xcfNNp\xea\xc9\xeb\xb9\xf8\x82gp\xcds~\x8a\xf3\xce<\x99\xb1\xd1Q\xc82\xda\xed
6Y\x96URu\xf2g`zh\xa1~\xc5\u0353\xedN\\\x10\x8c\xd6\x01^C;[7Q2uB\xe3G\xe2\xec\x8f\xfd\xb5\
x95\x90k\x11\xaf~\x199\xc8\xf2\xe6,\x1b\xda\ayxv\x15\x8f\xb6\x96\xd0u\x91W\xe8\x19W\xb1U8\x
96\x8c\xb1\xfeI\x95\x9a\u0322\xb6\xd2^\xd2\xcd\xc1#Dm\x19,\x9e\x9f
\x01$\xf3\xed\xd0B\x06n\r&\xb2\xd8\xc8bl\xe4\x95'\x91\xc1:\u023a]\u0124\xe1,\xa8\x9a\xbbU\xca
E\xca\u02bd\xd8\x00\xd5\xd2\xfa\xd64\f\x8d\xa6\xf5\xdcp\"\x98\xc4\x12\xa5\x11\x92\x19\xd2\xc4y
\u0146\x84\xf0e*\x19\x00A\xf6\x87\b6\x12\x1aV\xfcf\xe7d\x84\xc4R\x94\xcfN)\xf6+\xea\x81\xe3!\x
99\xcb)\x99\x83,\xf5\xa9>\x9d\xf0\xff\x89\u02fd#\x01+Hd\xfa\u0788\xbe,#-
\xd55\xce\xf9Ak\xc1<I%*\xaeJs\x8b@\u2434j\xa5+\x05\x90\x17\x18\xe3*\xf4\x15B\xbb5K\xd2Y\xc
0\x88\xad\xf9\xd5(\xca\xd4\xd4$'\x9f\xbcAO\x180\xefv\xbb\x02\xb0s\xe7\xce\U000ccc63@*\"Q)&p
,^\xb4\x88s\xcf8\x85\x91\xa1\x06\xb3\v\xed\x1e\x05\xaa\xf6\xf0\xd6eK\x97\xa7\x87\xd7(\x9b\xa7\
xa9\"\xaf\xc3D\x9e\xf5i\xbc\x1b\xa2\x8d\x18\x1d\x1e\xc5\xe0x|\xef~\xbe\xf9\u077b\xf8\xc7[\xbe\xc
5#[vp\xe0\xe0a\x924%\xb2\x968\x8e\x18j6\x18\x1f\x1b\xf5\x96\xae\x87;\xa4iZi\xb3K\xe06\xc6\x1
0\xc7\xcd\x10,\x9bAeu8O\x05?2=\u00ddw\xdf\u03ddw\xdf\xcf\xdf\xde\xf4U.:\xefL~\xfeg\xae\xe1Y\
x17\x9d\u0352\xa9q\xba\xdd.\xadv\xa72@\xd3\x1a\xad\xa0=\xc0\xf7T\xabg\xad\x1cn\xb9\xd9X\xf
e;\\\xefZ|\xb8\xc5\u034fH\xf1\x1c\u007fHZr\xae\xf9\xbaA\x860d\x13N\x1d\xdd\u02f2\xc6\f\xdb\x16
\x96\xb1ua\x05\xfb:\x93t]\xe4\xad\x12\xc41\xd8f\x98\xbeW\xabp]\xad\xccF\xa4\xa7\xfb)V\xc5T\x8
aE,\xe9\xf1j/\x0fo\xa9\x03\x94\x94<r\xfe5\xc6Z\xef\xf2Wx\xa3keyE\xeb!\u0185[\xa4\xabk\x0fs\xbe
\xda\x18L\xc3`\x87-Q\xd4 6C0\xaf\xa4\xb3)\u074e#\xcd\xc1?-
\xb7@\v\x8aI\x15\x9b\x81\xcd\u007fm'\x14#\xd6[f8z\x17v\xbc\xa4/sy\xf5\xedc\xfe\u04ae\u00e5\xa
e\x88\a,\x14=\xb9|P\x14\xb1\xd2\xfb\xd0\xcb\xf7\xa6\u2a18'u\x19)m\xb7L\xe82\xb2BLZ\xbe\xf2\x
92*$\x1a\xf2N\xb5\xa4XDzb#\xc3\xebe#\xe6f\xf6\xd1^8\x82\x89\xa2\xb0|\xe4_\x80F\xdc`~~~\xcb\
xc4\xc4\xf8\x96\x13\x06\xcc7mzD\x00\xbe\xf9\xcdo\xd9\xfd\xfb\xf7\a\xfc\xd2\xe2buN9c\xe3z\u05a
f]E\x92\xf9\xb5\xe9B\xc1\xa2=&Z=\b\xad\x9800\x19\xec\xcb!O\x114\xaa?\xa1\xa4T\f\xa9\x13L\u0
720\u0648\u0679{\x0f\xb7}\xeb{\xdc\xfc\x95[xh\xd3\x16\u069dnQ\x01\x8c\f\x0f\x95\xa9\u9744n2
M\x969:\x9d\xa3o\x8by\xd3{C\x96e\xa8s\xc4Q\xeceVY\xea\x97I\"\x83s\xae\xf88|\xf8\b\xb7|\xe3vn
\xf9\xc6\xed\\\xf1\xcc\vx\u00eb_\xca\u0557_\xc8\xe8\xe8(\xf3\xf3\v\xf5j\xb8\xd0E\x97+OR\xd3u\x
e8\x93\x00r)4\xe9Z\xacs\xb9\xa3P:\x14F`\x90=-
C\u0601\xfd\u0660\xac\x11\xa9\x87s/j\xcc3\x16w\xd80r\x90G\x17\x96\xb2ua9\xfb;\xe3\xb4]\x84\x
c1a\xc5\r\xbcbt\xa0\x1d\x1a\x83\xe5\xb1\x15]\xb4\f\x98\xdf\xf6=\u007f-
\xd7\xeds\xa9\x9d\xe4<o%\xc3\x14\xa9\x1f\xfc9\xa0\xd7\x1f\x8eT\x853\xa4\x81{\xb6&\xf0\xca&\xf
0\xd9\x02\xd6\x1a\xe2\xe1\x06\xf1P\x8c\x1b\x03\x9d\x880\xb3)\xeeH\x02\xad<\x0e\xb1\x10\xb1P
1%\xf5?\xbf\x9d\xa1\x89\x83\u0620C\x86\xacipVH(\xd5,I\xe6+\xef\xd4\xe5\xef|\xf8c%\xc4\xd2Ul\x
03\xf2S8\v\xc3Q-

+\xf4>O\xa1\x9e\x04\x8b\\)f\xa4tU\u0334\xbe\xdf\x14\x80\x06\xd3uu\xab\x82\x9a\xbfW-
x\xd6\xdf\xf1b\x98\x9f\xdeO{a\x1a\x1b\r\x05\xae\xc7\u007fw\x14\u01d2en\xcb\xe5\u03feb\xdf\t\x0
1\xe6\xdf\xff\xfe\x1dr\xc9%?\x95\xaa\xaa\xb9\xf0\xc2\v7\x1c<x\xb0\x14\fT\u07a8SOZ\xcf\xf8\xa2\
xc5>\x12\xac\xb2uXT/\xaa\x83\x82C\xca\x16_\x9e\xccx\xf4\x89\x83V\x16@\xdc!8\xb1\x8c\x8c\f\x
d3\xee$|\xed\xd6\xdb\xf9\xcc\xdf\xfd=\xdf\xff\xc1\xfd$\x897M\xb3\x91er|\x9c\xd1\xd1Q\x8c1\xcc\x
cc\xce07\xb7P\xab\xbc\xa3(\n`\xac\xfd\x8fQ\x95\xd6\xc2B\x81\b)e\xfaz\x1e\xde\x1b\xc51.P4Y\x96
\x91\x05\xce\xfd\x9f\xff\xe5nn\xbf\xeb~\xae\xff\xf9\x97q\xfd\u03ff\x94\rkV2;7_.;h\x95\n\xd0\x01\xa
0\xf5\xc4\x01]B\xc5\xdbk\x1cVMrr\xde\u06f2\xe0\xe7\xf5)\xd2:\xc7;l\a\xbe\xdbR.\xbdHOXF$\x19\x
8b\x1bsLD-
\u058e\x1c\xe2\u0445%l\x9d_\xc6\xfe\xee8\x9d,\xc6J\x16\xfc\xe7{\x0f\xb9:9\u0557\x88#u\xf8\x96
\nG\x9bs\xad\xfdFg\xdaw=;J\xa7\xda\xe2\x00Q\xe7\x15-\xc1\x06
'\x9d\xfb_S)*L\x05Z\x99#s0\u0590bXh\x82\u0574\xb5\x96\xc8z\aFg\x14\x1d1hf\u04594x\xc7K\x9
0\xeeU\x0e(\xe3W\xf6Q\xa5\x91\x80\xa6\x8at\x95l!#1B\x1aA7\x12\x92HH\fE6kI\x93\xf8\u06575`\
x9a\x86,Q\x9cG\xfa\x82K\x0f\xedT\xc0L\x87D\xa67\xcf|\xa0\xc0\xc9\xef\x06H\xed\x8c\x15\xa4\u0
432;QL\xaa\x98\xc4\xd5_lWJ&\xa9\x1c^\x85r\u0265\xb4\x17\xa6I\x93\x84(\x1e\xae\xa5\x17\x18#\
x9c~\xc6\xe9\x13\xcbW\xac\x1a>!\xc0\xfc\x96[\xben\x80l\u05ee\xc7N5F\xce\f\xa0&\u057d\xc2f#\x
e6\xe4\rk\x19\x1b\x1fgvn\xc1'w\xd7\xe0z0<W\xdb\xfc\xa7c\xe0YW\x80\x18\xb2\x00\xe6\x19\u07b
c\u007fll\x84\xc7\xf7\x1f\u14df\xf9\x02\x9f\xbb\xf9\xab\x1c:t\xb8,6\xace\xd1\xd4\x14\x93\x93\x93\x
8c\x8e\x8e1<<D\xa7\xdb\xe5\xf0\xe12\x80$\xe7\xb3\xe38&\x8e\xbd^ya\xa1U\xff\xddZ\xa7U\xf2C
@U\u0272\xccWL\x95\x96Y\x8a\r5G\x9a\xa6|\xf4\u007f}\x91{\x1e|\x84\xdf|\xcb\xeb\xb8\uc8b3i\xb
5=\xad\xd3\xeb$\xa7\x03\xba\x97'Z5W\xfdbL\x8dX)\xdf\a\x13\u876c\xf0a\xa7F\x9a\xfd\xd83\xeds\
x9a\xc2\xf4\xaf:*\x10\x19\u01d2\xc6,\x13Q\x8b\xd5C\x87x\xb4\xbd\x84-s+\xd8\u05d9
U\x0f\xfa\xd2\x03\xe8}\xae:\x15\"WzC\xa8k\x03\x90J}\x1f\x14\x16n\xc0\xe3-
\\&M\x9dKpi\x86\xcb\xd2`o\xab\xf5\xfb\xa2j\xe1[8#zc)T\x19\x8a\f\xa6Fa{@\x8f\xac\xc5Z\x13V\xd5
\x05\xd7\xce\u0226S4\xf5\x8a\x95\\\xa1\x97\x03\x9d\x06.]\x05\x92D=\x90K^LK\xa0\x80\xbc\n&\x
b2\x824\xc4W\xeb\r\x13l\x90\xebY\x9fV\x80\xd8\x17H\x0e\xe7M\u01ea\x9d\x86S4\r\x87\xa4\x95\
u06b6n\xcd\x17I*\aE\xe8\xfdL~\x9c\x06
W\x11\xc4\t&Q$\v\x94\x91\xcb\a\x9f\xc5\xf4\x13\u0514T\x95*\xc6Z\xba\xdd\x05Zs\x87k\x94\x98\x
88\u0c8c\x91\x91a\\\x96\xdd\a\xec<!\xc0\xfc\u0421\xc3\x02p\xd3M7\x9ffmtN\xadA5\u07a8f\xe5\xf
2\xa5lX\xb7\x8a\xc8\x1a2\x858\xac\xd1\xcb\xd1\x17\x1d\v3+\xa9\x00\xc7\xd33\xec\xf4K\x11\xa9\
xda\xe0
\u8946\xa3\xe3cl\u06b6\x93?\xfd\xf3\xff\xc5?|\xf56@\xb1QL\x96\xfa\xaa|\xf5\xeaU<\xf7\xb9\xcf\xe
5\xc2\v/`\xfd\xfa\xf5,Z\xb4\x88\x1d;\xb6s\xdbm\xdfd\xe7\xce]\xec\u0673\x9bm\u06f6\xd3n\xb7\xe9
v\xbb~N\xb0x\t\x13\x13\x93\xec\u077b\xb7\xafe\xcei\xa827RI\x92\xa4\xf8\\><\xf5U\x96\xc1\x1a\x
e3U@\xceq\xc7]\x0f\xf0\x1f\u007f\xfb\xfd\xfc\xb7_\xfdw\xbc\xec\x9a\xcb\x11\x91\xe0\xec&\xb5\x9
c\xd2\xfc\u0412\xd2\x11\xfc\t\xe2d\x90\xe1\xf5\x01\\p\x1e\xacU\xe4n@\x16\u04cf\x05\xbb\xfb\xc1
\xd1\xe4\x8b
\xbdu\x81\a'\x14\"IY\u059ce\xaa\xb1\xc0\xaa\xe64\xdbZ\xcb\u063e\xb0\x94C\u0771`\n\x15\f\u07a
4\xbf<\xac\x99\xf9\xf6\xc8l\xfb\x00=\xff_\xa75\x9fm*Fa9\xb5\x12\xca\xe8pP;\\\x9a\xa0YVKw\xea_
?\u0329\x1a_\x8d\xa30\x12\x19\x0f\xacR\x89H\x93|\xaeaH\x9c\x90\x89\xd2\xed8\x92\x83\t\xc9l\x
8a&Z0\x1e\x1aI}\xdb9\xf3+\xf0>K\xb3r\x00\xd9p\xad\xe6\xcfO\x15\x9b\n\xdaq\xb8\u06106=\xb0k
$\xc5p\xba\x1cx\x12R8]\x91\xf8T\xbc\xbeN!\xad\x04k\xf4X\xe0\x1a\xa9\x843\x8a\v\u0665\xe1,\xc

8\xc3,\xf2\f\x83T1\x9dJ\x9c\xa4\xcb3\x82\u02d7T(\xa9_\u007foY:\xadYZ\xf3Gz\xcc\u0494,K\x98\x9a\x9ad\u7b9d\xdf\x13\x91\xec\x97\xde\xf4K\x8d\x8f\xfe\xf9G\xbb?\xd1`>==m\x00\xb6m\xdbv\x92\xaa6\x8aN2\xac\xdef\x99c\xdd\xea\x15\xac\\\xb1\x8cV\xe2\xfc\xc6\x18Y\xc9\xd9\x1dc\xd8%\xc5e\xf9\xf4\x01y\xe6c\xa2\xfd\x96[\u042a\x8e\x8e\x8d\xf1\xe0#\xdb\xf9\xfd?\xb9\x81o}\xfb{\u0638A\x1c\x19\xda\v\xf3\x00\xbc\xe4%/\xe2\xcdo~3\x97\\r1\u02d6-
#\x8a\x9a\xc5\xcf}\xc5+^\xc1\xcc\xcc,\xd3\xd3G\u0633\xe7q\xbe\xff\xfd\xefs\xe3\x8d\u007f\xc3\xfd\xf7?\xc0\x81\x03\a\x98\x9a\x9a\n\x11V\xd9\xf1\x1f\xa3j\x0fo\x1a\xb6\xd3\xc2.\xa1\x84e\x13\xa7\xca\xce=\xfb\xf8\xad\xf7~\x84\xf9V\xc2\xcf\xfd\xf4\xf3h\x1aC\xa7\xdd\xed\xc9\xc0|r\x15\xf9\xa0?f@\u022f\x14\n\x16\t\x16\xbdZR0OsY~\xac\x1f\xa7\x86\"N/_\xa1\xd7\xca\xd6
\x15*&2\x8e\x95C\xd3,j,\xb0f\xf80;\x16\x96\xb2}~)\xd3\xdd!\xef\xd0(\xae\x9c\xcb\u0220\xf7\xa6w\x96\u04eb\x89\xa9\x1eh=)>\xc5PS\xfa\xaa~M3\x0f\xe6\xce\xd5N$?
5\x18\xebm\xe0\xc8\x14\xe7\xb20T\xf4\x1cyd\xbceK\x8e\xe2\x12~h\x06$\x99\xd2\xe9d$\x89\"\xd3)2\x9b\xf9
\xe3\xf0d\xa4\xfaV\x85\x83\xcf8*1r\x94\xe1\x11\x198[\x1e\x90\xb9\x85\xae(\x98,\xc3v\x85\xac\xad\xb8\x86\xe0\x1a\x824\fD\x94\x9e\x97\xd6?\x17\xe9:4\xed9\xf0\x83O\xbb\x982tBzl0\xf2m\u06b2\xf8\x97\x9aK\xa5(\xd0\r*\x16\xf5\U000c0a85\xb6jM\xf2\x12\xdec\x1f\x1e\xd2\xed\xcc\xd1i\xcf\xd6<\xa4P%MR\x9d\x9a\x9c\x94s\xcf9w\xfc\xe6\x9b\xfe\x1eu\xee'^\xd1\x12}\xe4#\u007f\xde\x05\x98\x9b\x9b=\xe9\xb1\xc7\x1e\x030Z\x88\xf4\xfd\xab\xbdr\xc52\x16OM\x92%\t\xb1\xb8\xa2\xea\xabUY\x03\xd6\x02\r\xfa#i\xa6\xfb\x95\x19\xf8\x19L\xd3\x00\x00
\x00IDAT\x86.Q\xe1i\x8d*c\xa3#l\u06f5\x8f\xf7}\xe8/\xf9\u05b7\xee`dr\x92\xd8Z\xe6g<}r\xfd\xf5\xd7\xf1\xeew\xbf\x8bU\xab\xd6\xe2\x95;-\x16\x16fQ\xf5C\xa6\xc9\xc9I\x16-
Z\x02\b\xe7\x9d\a\xd7^{-
\xaf~\xf5\xab\xf9\xfd\xdf\xff\x03>\xf6\xb1\x8fs\xf0\xe0\xc1\x10\x97\xd7\xf3\xec\x02\xb7\u9723:,\x1e\x04\xf0\xf9A
\x81\xb7\x8f\xc4\xe02\xc7\xec\xdc\x1c\xbf\xfb\xfe\x8f142\u032b^\xfa<27G\x92$E\xcdl\xa4\xf4\x16\u0441U\xfb\x13H\x19\xa2\x8eo}1^\xbd\a\xad<}`=H\xe1\"\xfd\xa7M\xb8\u0524\xcc*\xed1m\xf3]\xa2_\xa6id\tk\x86\x0f\xb1\xb41\xc3\xda\xe1\x83l\x9e]\xc1\xf6\x85e\xb4\xb3\x18+>]\xaag&YqN-
\x97k*\xa3\x9e\u0497\xa7w6\xa1u\x85Q\x9e\x1eo\x83\t\xb8:\x87K\x124\xcd\x02\xc0jA\xc5\xd9(\x06c\xc8B\x97G\x14\xa3b\x8bMb[XDT\x06\xf9\xea]\xfe:\x0e\xda*d\x89bgS\xe2\x85\f\x93\xe5+\xed\u07a7\xa4\xf0\x8fq\x83\tN5R\x02b\xbenoKUHMq\xe2
\xea8\xe8\xf8\xe1\xa16\x8c\xff\x88\r.\n\xbe5\xc6K'\x1d\x9e\xc2\xd1\xfc\xec2~1\xce\xd4y\x16\xaa\xf3J\x18\x14\x82\x9eK8\xfd\u01a7\xe9:$+\xccX*\xd66=I\x00A\u032e\xea\xedE:\xadiZs\x870\xc6R\xb5hTU\x19\x1d\x1de\u035a\xd5\x1d\x80\x95+W\x9c\x18j\x16U]\xf5\u05b7\xbe\xf9\x92\xb9\xb9\u067e[\xd1Z\xc3\xdaU\xcbX41J\xbb\xd3%\x92\xca\xd0(\xf8G\f\uea5f\x1e
\xa7\x02\xe4\xa5\u046e\xe7\xa1\x1b\x8d\x06\xfb\x8e\xcc\xf1\xc1\x8f}\x86[o\xfdg\u0196.crb\x82\xf9\xc3\aH\u04c4\x97\xbe\xf4%\xfc\xe1\x1f\xfe\x01\x8b\x16-\xa5\xdbm\x15\x83JkK\xb3\v?\x1c-
S\xa5\x9a\xcd&\xa7\x9dv\x06\x1f\xfe\xf0\x9f111\xce\xfb\xde\xf7\x81\x822\xa9\xfei4\x1aDQD\xab\xd5\xea\x931V\x01_{\xac9\x9dSo\x05`=\xed277\xcf\xff\xfc\xe0'X\xb3z%\x97\x9ew\x06N\xe7\u0086\xa2T\xb8B\x18\xb0\n\xd3'0<^\xf73H\xc9.O\xcb
\xfa\x89\x02\xfaQ\u010bR\x16\x04\xd2[\x1d\x88\x14\x99\xb0b\xbc6\xc7:\x88m\xc6\xc8\xf0\x01V4fX3t\x98\x87\xe7V\x05>\xdd\u0516\x8e\xaa\xd4Sm^_qZ\xef\x9bs\xd7\u01a4\u055f\xe0\x15
\x9a{\x91Wl\x1c\xb4\xc2\a\x18kI\x93\x0e{\x1fy\x80}\x9b\u007f\x88s\x19\x93kOa\xc5\xc6s\x18\x19

\x9f
\xa4^\xf8\x85\x9c\u04359\x85\xaeS\xda\x0e\x92\xc8\u03c0\x1a\v\x19\xf1\xbc\arq\x81O\xce\xc1\x
bc7\xb0\x9b\xba\u04e1\x14\xe9\x12\xfe\xf5\x13\xa5\xf4twR\xd0\x1cyu\ud36c\u00b5\xd1q\xa88L\f\
xb6ap\x91\x906<\xbfN\xb8n\xb5\xdaI9\x87\xba\x10\xfa1\xd0)I\v\x17H[\x8dc\f\xef\xbb\xc9\x14\x9b\
x849E\x90$\xaa\xcb\xfd\u07a9\xa5~ke\xbadl\xc4\xcc\xe1\xbd\xcc\x1c\u068d1\xd6\xd3>^J\ud187\
x87\xcc\xec\xec\u0716\xd1\xd1\xd1;\x00Z\xad\xb6\xfbI\as\x03\xf0\xf8\xe3;\xd3M\x9b\x1eiW;\x11\r\
x14\xc1\xe8\xc80\xabW,\xa3\xd9h\xe0\x82\v]\x8d\xa9\xcb#L\xfa\x86w<M@^\xbf\xd7L\x80uk\x04L
\xccM_\xf9&7\u007f\xf9\x16\xec\xf0(K\x97\xad\xc0%\x1d\x8e\x1c>\xcc\u06b5k\xf8\xad\xdf\xfaM\x
16-
ZJ\xbb=\u007f\xcc\xea\xb9\xfa\xa7\xd3\xe9\xd0\xed\xb6\x88\xa2&\xbf\xfd\xdbo\xe79\u03f9r`U\xde
l6\xb1\xd6\x1e\xf5\xe7\xd6\x16\x8dz*\xf54s
\x86F\xec=#v\xed\xde\xcbG>\xf9\xb7\x1c<2K\xa39\x84\x18\x9f\xb7\xeaz\"\xb7\x8e?\x16>\xfek\xd
8o>\xf6\xe4\x17\xfc\x95\xbe\x9c\x80'\xf0(\xe8s\xd7\u0523\xf0\xe8j\x14\xb5\x12\xbcG@\xabv\u06a
6\x944\x1aq\x8cEm\xce\x18\xdf\u00d5K\x1e\xe6\u00a9G\x99j,\x90\xa9\x1f\x8eW\x96\"\xfbsV\vu\x
86\x14\u04b9\xa3\x0f\xf4\xa5\xc2g{\x83\xa8\x99N\xca|\xab\xeb;)\x97\x15\u065e\xbe\xdd7l\xff\x97
o\xf2/7~\x84\a\xff\xf7\x17x\xf8\x1b\xff\xc0\x03_\xb9\x91G\xbf\xff\r\xd2n\v\t\u05cd\xaa\x97\b\xb6Re
\xbe\xebXH\x95D\f\xd8\xff\x8f\xbd\xf7\x8e\x92\xe38\xefE\u007f_Uw\xcf\xcc\xeeb\x01,r
\x88@\x80A\xccA\x04\xb3$\x9a\x92\xa8@E\u0496D\u0274l_Y\xbe>\x96l_\xf9\xfa\x9e{\xed'\xd9\x
ef\x1d\x9fg\xeb:\xe9]\xcb\u05b3h_\xcbV\"MR$%J\fb\x00s\x06!0\x00
r\\\x84\xcd\x13\xba\xab\xea{\u007fTuw\xf5L\xcf\xee\"\xf0<\x1f\xc1s\u0392\x8b\xdd\xd9\xd9\xd9\x0
e_}\xf5\xfb~A\"l2\u00ba\xca0p2\x16\x13\x87F\u0198!?(\xdd/\xe2\x9c\xd1GrV\x88+\xfa\xa4Sow\xc
e$\xfa\x94a\xe19\xffR(\x83\xa0i\x10\x8d\x1bD\xe3\x1a\xd5\x11\x85\xeap\x82\xa8i\xdd\f)\xcb\xe8K
\xb38\xb9\x80e\x15\xdc=3\x8b\\\u4ed2\x141ag\xaa\xe5\x06\x9e\"\xc3\u033d\xdd\x15\xf9\xc3\xcd\
xd4)S\xc0h\x85\x91\u00fb\xd1l\x8c\x822Y\xafe\x17\x85a\x809s\x06\x0e}\xe63\xb7\x1c\x06\x80\x8
b.\xba\x90O\x8ab\xfe\xdcs\xcf\u03cb\xa2\xca\x1a\u02ea\xc8\x1f\xda\x18\xcc\ua7c1\xb9s\x06,\x8f\
x1f\xc2\xcf\xf1*/\u0669\u007f\xc2\t\xf3\x18\xf1\xec\xb1R~\xb0\u327f\xf6\xc6\x16|\xef\xce\x1f\x81c\
x85@\x12\xf6m\u07c2}\xbbl\x00\xf5\xfb\xdf\xff~\\y\xe55\xd0:\x9eV\x11\xef\xbc\xd1\x15f\xce\x1c\x
c0M7\xddX\xfa\x9c$l\xd0l6\v\xf8x{1\xef\xf6\xef\x94\xd9\x02BV\u041f|\xe6E\xdc\xf7\xe0:DQ\x05\u0
489\x1f\f\x93\a_\xf11\x1f\xd3\xc9\x13\x9b\xe8\x04\xcc2\xa6ge\u0325\xef\x87:\xb9\xe1i\x05\x16%\
xbe\xda\u0085P\vd\x1e\xd8\x01\x19\u0329\x8c\xe3\xfc\x99\xdbq\xd5\xc0\x1bX\u077b\x1f\xa1\u04
08Y\xc2\x18j\xdb\xc1P\xa9\xc8%\xfb\xbc\x8d\u0398\xef2s\x06\x86\x01c\"N0<\xd1\xc4\xc8D\x82z
K\xbb\u0754\x86\x8c*8\xbc\xebMl\xf8\xc9\xed\u063f\xf9g\x98\x18>\x84\xfa\xd0A\x1c\u0679\x05\u
06de}\x18\xc3{v@\x86\u0591>a\xa0\xa9\x18-\x05(&\x18!@2@\xd4\x12\x88\\G\x0e\xb8n<\xfd<-
\xca\u0736*\xa6\x05\x10\x19\xa4\xecX!\xa978 \x94}\x9dla
\xce\x16\u0334\xcdN\x03\x93\xd3\xd7\x13\xca@$\f\x910d\u04e02\xa6Q\x1b\u05c8\x9a\x06B\xe7
6\xb4E_k\xdbY\u06cf|x)|\xc8%\xe5\xdbk\x06Z\xc6\xce\x03\x98<\x16\v2\x87\xd2\xf6Z@
H\x19\xa211\x84\xc3\xfb\xb7\x80\u074e;}\x8aVZ\x0f\f\f`\xf6\xec\xd9\u06c8h\x1b\x00\\v\xd9e'G1\x
1f\x1b\x1b[5s\xe6\x8cU)\x1f\xdb/j\xfd\xfd}\x98=\xab\xbf\xe0Q\xc2\xecd\xf3\xae{4De\xb5\xf7\x84\r\
xd0l!\u03e1\x96\xb0RE\xbd\x11\xe3\xbe\a\x1f\u014em\xdbA\x81@k|\x14\xadF\xbd\xd0a7\x1a\xe3
\x19m\xf0h\x1f)\u03bdl\xd9)\b\x02Y\xda\xc1+\xa5&\xc5\xca\xcb~o\xfau\xcbC\xb7\x17a\x14\x06\x9
8\xa87p\u03cf\x1f\xc1\xe0\xfe\x03\xe8\x89B\x80\x04\x98\x84c\xb3\xe4v\xab'\x92i\u00a5{\x9f\xe9/\
x0e\x1eWa\xda?\u05feK\x98\xca\u317a\rN]{g\xa5\xf8\xb6\xfb\x8bHci\xed\b.\x1b\u0602\xb7\xcf\u0

78a\x05\xd5\x11(\x12HXf\x96\x00\u0183\xac\x8a\xd3\x03W\xe8\x8bI%\x056E\xaa\xa2\xd4l\xa0T\x82VKa\xbc\xa50\xd4\xd0\x18j\x1a4b\r\xc8\x10{__\x8f\xfdo\xac\xb7\xf1t\x89\x85b\x92f\x1dC{\xb6a\xe2\xc8
\x88$\x12m\u0408\x8d\xa5(\x12A\x93\xe5\xe5\u0216@Xgk\xdce\\A\u0576\xbb\xee(\xe2\xd3\xd9\xca\"g\x87\b\x9d\x17u\x19[h\x03\x9c{\xbe\xf8\x8b\x85\xff\xb3\xc20\xa4\v\xe3
\xc3\bbFe\u00a0:n\x10\xb5\xac\xc8'\u0760\xa7\xb9\xab\xac\xed\u0417\xb5\xb1\x9fgktq&!\x13\x03R\xce\xee\xd6\x13\\\x11\xfc\u0646g\x94\xee\x16W\x19D\x98\x18=\x8cC{7e\xc3\xcf\xf4\xacj\xa3\xe5\xc0\xc0l\u031f?o=\x00\xfc\xf1\x1f\u007fY._\xbe\xf2\xe4(\xe6\x1b7\xbe\xaa\x06\a\a\xb3A\x9f\xaf`\xeb\xed\xe9A__/\x8cq\xd1j\x1e\xf9\xdf\x0e\xcf\xe9-
\xf3~7Y^g\xfe;\x04Y\\{\xeb\xae}\xf8\xe9c\xcf\x00\xac\xc1\xaa\x93q\xf4\xd0C?\xc5#\x8f<\n\xa2\x00\xc7P\xcb\x11\x046\xb7c\u02d67a\f\xa3R\xa9\xa0\xaf\xaf\xaf\xd0]\x9b.\xf1&e]y\xe9\x82BdS\xe3I\f\x02\xbc\xbey+\xd6=\xf5<z\xab\xa1g\x1bzb|\x14\xa9t\xad=\xf6\x84\xa2b\x8eT{\xb6ig\xc7\xde\x1e\xd0L\x1d{\x83\xee\u007fkY\u007f\xc0\xe4D/d\x1d\xf9\x02h\baS\xafz\x83\x16\xce\xea\u06c3+\xe7l\xc2Y\xfd{\x10\x89\x04M-
\xa1\x8c\xf0\xc6\xc0m\xbf\x81\x8a\xdb\xf9\u00bb\xa7\x9c7h\xc06\x1cZ[:\"\xd8F)N\xc4\n\xe3J`xx\x04\x83\xdb\xde@\xab>nAA\xa3\xecN,\x8eaT\x02\b\x89\xa6f4b\x03\xc5\x04&\x81\x04\x84\x04\x04\xd1\x12\b\x1b\xce\x02\u0597\x9cr\x8emw\xc2VS\\%\u073ek\xb6\xf0\x8c\x15\xe80d\xcbv\xde\xc4m\x8b\x19\x173VH\x1b\xdb\xd9\x1b\xc7\xccd \x88\r\xaa\xe3\x1a\xb5\t\x8d\xb0e
t\t#\x93m\xf7M\x86\xddB\x98\x8f\xf2Y3\xa8e\xa1#\x06\xdbP\x0e\xb7\x80\xb1\xe1\u04acS\"\u01dc\x010zd\x0fF\x8f\xec\xc9\xee7v\xbb\x820\bDT\xa9\x1e\xbel\xed\xda\r\x00\xd0\xdb\xdb\xfbs\x8f\x97g\xc5|\u03de\xbd\x18\x1b\x1b/\x14\xa1\xf4D\xf6\u052a\xe8\xed\xadA\x9b\xd4ZV\x14\xd8%\x9aDi\xfft\"\x18,9\xb4\xe2\xba$X\xdb\xd9$Qx\xe9\x95\u05f0w\u07c1\u049f\r\xc3\x10\xbbv\xed\xc2]w\xfd\x00Z\u01e8\xd5z;\x86\x93\xdd!\x16\xb8N\\\xe2\xe0\xc1}\xb8\xed\xb6\xdba\x8c\xc1\x9c9\x03\xe8\xef\xef/\fO\xfdb]\xc6x\x99\xb2\u023b\xf7$\xa4D%\n14:\x81'_\u0608\x91\xb1\t\bY\xec\x96\u028b\u0431\xc3-
8Nw\xc6\xe9\xa0\xf7<\t\xd43\x95?\u03d4E=\xc3\u05d1:2\x81\x04C\n'\x1b\x17\x8c\xf9\x95\x11\\2k+.\x1d\u0602\xf9\x95Qh\x16HXt.\xeel\xfd\xc8\xd9\xd8\x0f\x93\x89\u007f\x00_\xe3\xcf00:\x86\xd61\xb4\u0459\xe5-
\x01\x10l\x80\xb0\x8a\xc1]\xdb1\xb8e\xa3\xf5\xc6\xf7\xa6\n\xba\xd5@\xcf\xdc\xc5\b\u7702\xb1F\xcb\xc2*
(\"(\x10\x82\x16\xa1\xda$\x04\x9ar\u1303X2\x8eu\xdb\x11\xe2\xc9\x0e&u\xc20\xed\xc5\xd1\x0e\x1e\rdl
R\U000bbdc0\x907t\x04\xac\xa7\xb8L\\QO\xb1k\x06\u0096AuB\xa32\xae![^\xb4\x90scd\xc30:\xb7\x83Na\u007f\xa9\fD\xa2s\xe8\xc8\x15\xfd\xfc=\xd87[8\x1fD\x10A\x00\x954ph\xef\x1b\x88\x9b\x13\x10Bf]\xbf1\x86{{{10{`\xd7G>\xfa\xf1\x97\x01\xe0\xf2\xcb/\xc3IS\u0337n\xdd\xdaj4\x1a\xa5\u017c\xb7\xa7\x8a\x9ej\xc5%\xebp\xe6H\u01c0\x1b4\x15\x15\x1f\x14\x06\xf0H\r\xa9<=\f\x024\x1b\r\xbc\xfc\xcaF$q\xdc\x15\"\x91R\xe2\xb6\xdbn\xc3_\xfe\xe5_\xc1\x98\x04\x95J\x0f\xaa\u055eI\xf1s!\x04\xaa\xd5\x1eHY\xc1\xd0\xd0!\xfc\xe1\x1f\xfe\x1fx\xfa\xe9\xa7\xd1\xd3\xd3\x03\"\x01{\x8cDa\xfb\x97\x0e@}X\xa5\xfd\xff\xdd\x06\xa2\xa9\xe4\xdf\xf2\xcf\xed\"\xb1y\xfbn\xbc\xb1c/(\b\xf3F\x8c\x8e\xbf\xc0v\xde\xcb\xdcm\xea1\xed\xd7,-
\xc0\xdc\xd9G\xfbW\aO\xf1zG\x15\x9dG\x80\x91\xf6#\xa5\u03f1d\x904\x10\xc26cU\x19cM\xdf~\\=g\x13\xce\xe8\u06c7\x904\x12\x0e\xb2\xdcX6\nF\u06ee\xd9h\x05\xa3\x95\x15\x00\x19c\x8b\x88

\xfb}\xc6\x18(\x1dC\xe9&4+K746\xf4\x02N\xcc\"%\xe1\xc8\xf6\xd71\xb2{+d\x18Z\xbb\xc0\xecZ0\x98\xb9\xe2LD\x8b\x96#Nb\xb0\x14PD\xd0L\b\x9a@\xa5\x01H\x96
\a\xed\xb0+n0%\xf88\nU\u0583\xe1:q+\xa2I\x86\x1c).\xaes\b\x86<\xdf\xf0\x8c^$rA\x15\xa50\x8d\xe79N\xb0\x01\xcbA]\xa3:\xaa\x10\x8d\x1b\x88\x98\x8bR\xfe\x14z\xa1|6!2n9\x1cf\xee`Hmrr\xb9\x9f[\xe7\x86\x01RFhN\x8c`\u07f6\x97aX\x81\x84\u022c\x01\x94R4o\xfe<\xacX\xb1|\x1b\x11\xed\x01\x80\xb5k\xaf8y\\\x13W\xacX1\xe7\xe0\xc1\x83mrg\x06\x91@__\x1f\xaa\xd5j\xb6\xaa\x8a<\xab+
\x0fa\xed0\x14:1\x83O\xe6\xdc\u0396\x1d\x8fIJ\x81\xb1\xf1:\xb6l\xdd1i\a\xcc\xcc\x18\x1a\x1a\xc6
\xff\xf8\x1f\u007f\x84\xad[\xb7\xe2s\x9f\xfb\x1c\xce>\xfblDQ\u037b\x9a\xd3\xddW\xdem\x8f\x8d\r\xe3\xb9\xe7\x9e\xc3_\xfc\xc5_\xe3\xbe\xfb~\x840\f\xd1\xd7\u05cbF\xa3\x81\x89\x89\t\x18\xa3\n\xf8w\xd9\xe7\xdd\xdeS;,\xc3\xccPJ\xd9E\xa4RA\x14\x86\u0633\xff\x10\xde\xdcy\x00\xabN[\x05\xc3\rOP\xc7G\r\x8a\xe4\xae+\xe4\x05w\x94\xe1\xde\xc7\xd7\xed\x17zF\xe2i\xf5\xed]\xbd<\xa6\xf9\xbe\n\u040c\u021d\x03\xb3\x81.3\xa4+J\x02\x1a\v\xab\u00d8\x1140\xa72\x86\xd7\u01d6\xe0H\\\x03\x99\x040IfA\x9b\xdf\x00\x1e\x01\x9d\xac\xb3\x9f6\t\x14+(\xd2Pnn\xc4Y\aK\x10Q\x15c\a\xf7a\xf7\xf3\x8f i\xd4!+\x95\xac\x90\xeb$F\xcf\xc0B\xcc[y>$\"$\xba\t\x04V{\x1f4\x19QL\b -\xbd.-
\xc0&\x87U\xc8cw\xb4w[\xbe\x10\xa7c\x89%\x14B6:\x96N\xff\xe9\xda\u044c\xdb\x17\x03\xedL\xca\xd9[\xfc\x8d\xa5\x87\x8alwdM\xceSk\x01\x91\x18\x04\x82\x90T\b*\xb2\xb35\x18\v\xb7\xb0K\u0720XC45\xa0\r\xc8\x18\x90vN\x93\x86Q\xd0\xf702\x85)9!\x163chp+\x0e\xee}\x1di\f\x94\x15\x9fYk\x8d\xfe\x19\xfd\xb8\u448b_\xec\xb6#\xfe\xb9-
\xe6\xcc\\\xf9\xda\xd7\xfe\xfa\xba\xc7\x1f\u007f\xc2Q\xab\xd2H)\x06I\x81J\xb5\n\x11F.\u02638\x1cI\xf3\x1b\xa9M\xa4q\x82Jyf\u07aa!m\a\xe3\x9c\xea\xc7\xc6\xc70661)\x1e-\xa5t\x8e\x88-
|\xfd\xeb\u007f\x8f\a\x1ex\x107\xde\xf8q\\~\xf9\x15X\xbat\tf\xce\xecG\xa5R\x05\xb3\xc1\xc4D\x1d\x87\x0e\x1d\u009bo\xbe\x89\a\x1f|\b?\xfe\xf1O088\x88(\x8aP\xadV\xd1j\xb5\xd0t\x1e*\xedB\xb2\xf6\xae<\xc3\xef\xa6\t\u0527\x05\xdd0#\f\x03\x8c\x8cO\xe0\xf0\u0430\x1d\xdcN\xa3\x1b\x9e\xea\x18v\xf1\x13<a\x90J'\xc2\xec'^r\x9bUr'R\u007fBo\xb3BZ\xbd\xfdL8\xf1\x1b3\xa1/h\xe1\x9c\xfe=\x98\x1d\xd6\xf1\xfc\xa1\x85\xd83\xd1\x03\x06#\xa0,t\xceI\xdd\x18\f\r2\x02\x06\n\xac\x15\f)h\x97\xf8c\xfc\u25aa7E\x80\xe1\x9d[px\xf3+\xa0\xc0\xcdi\x8c\xf3\xf3V1\x06N=\x13\v\x96\x9f\x0f\f\xd7!\xc9\xca\xd9E\x02DM@\x92\x04\x85A\xa1\x90\x921\x85\x98\xc3\x0e,\xd0;\xe8L%K
\xe73\x00\xe2\u026c\x89\x19\x1d\xb2X\u07fa?\xf5L\xf7\x95\xc9^\x88\xb3I\x8f\x81\x82\x15\x10\x05\x96\xc6\x19\x18\x03\xa1\x00\x11\v$5\x01\x1d\u0602n\x17\a\x03\xd9L
\x12\x97\u03ea\x8d+\xea\uc0ba\xd9\xdbY\x14m\x1a\x88$t\x12c\xdf\xf6\xf5h\xd6G\xb2\x9dA\x9a\xe9\xda\xd7\u05cbY\xb3f\xed\xfb\xe0\a\xde\u007f/N\xb2G\x00
\xaa\xd7\xeb\xb3\xe28.x\x8d\xa4\x85\xc6xqN\xec\x05\xf5\xb6[A\xe5\xcf8\x91\x18\x10g^\"v[l\xb1\xb3V+\xcf\xdb,+\xe8\u030c(\n\x11\x86\x01\x9a\xcd\x16\x8c1\u0632\xe5M\xfc\xe9\x9f\xfe\u07e8\xd5jX\xbe\xfcT,]\xba\x14\xfd\xfd\xfd\xd0Z\xe1\u0211!\xec\u0631\x03{\xf7\xeeE\x92(H)2\x1e\xb9R\t\x94\u0496O\u0705\x9d2\u067f\xa7\u04f9k\xadQ\xaf\xd7\x11\x86\x12:\xd1h6[^D\xd6\xe4\u066aS\xb6\xad\xa5\xfd/\x9f\x90\xba\xe9\xef\xfa\xd3\x10\xe7t\xaa\"\xbc\xebc:\xc1\xd5\xc7\xea\"@S\rh\t\x10\x82All\xb2\x0f1\x16\x84\x838\xafo\x04\x15^\x80\x1d\xf5\xb9HL\x88@\x98\xec/I\x17\"m\x94\xc5\xd1YC\a\x8e\xcd\"l\xfa\x0fs\u0389\x11A\x05\xad\x891\x1cX\xff$\x92\x89\x11\xeb\xab\xed\\7Y)\x84\xb5>\xcc_}1j}s\xa1\x1a\x13\b \x00e Y@\xb2\r\xba
\x9b\x9e\xec\xe2x\x8c\xc5\xcd\x1d\xae\xdd\xee[\xe3S\x83\xbb\xda\fS\x11\xf2/\x1ea*\xc1\\\xa8}e\xc8\xe1\x96IN\x90\xe0\xb4\xe3v\x87N[\xa6\x11\t\x824\fj\x1aH\u0148k\x02\xa6F\xd0\x06\x10-
\x05\xd9T@\xcat\xf1\x84P\xc4%\xa3t\x8f\xf9(\x84D\xdc\x18\xc1\xbe\x1d\xaf@%-

\x9b\u0104\xdc\\k\u05acYX\xb6\uc517\x06\xe6\xcc\u007f\x19\x00n\xbd\xf5\x1f\xf0\xd9\xcf\xfe\xda
IS\xcc\xcd\xf8\xf8\x84\xb2\xea\xc8\xf6\xb9\x90\x8d\x87#O\xfak\x9d\xce\u0716\x06o5\x14e\u007f
G\x00\xbb\xb5\xd5,\xc0\x9cg1NVP\x93D\u0658\xb8\xcc\xd2\xd6~4\x1a\r\xbc\xf6\xda\xebx\xed\xb
5\xd7\xdb:y\x01!l*Q\xfa\xbbS\v\xdb\xd4\x1dq\xb2\x9d\xc0d8<\x11ux\xbb\xb4o\xfd\xb4\xd6v\xf0Jde
\xdf\xe9\xce\xe7X\xecR\u0607?\xca\xfb\xe7\x13a\xc1R\x9c\xb1\xe5\x11}HM\x99\xb2!+OVk\xba\x
0e@\x8f\xb7\xb8\x17\x1aV\x01\x10\x1b\u8111$\x06\xb3\xc2\x18\xe7\xf6\xc7\xe8\x93-
l\xae/\u0138\xa9A\xa6>Bl\xb1p\xc3\x16;gg\xfc\xc5\x04ha\xa09\x8b\x9b\xb2\xc7X\x06\x18\u06ff\x0
3\a6>\xe3\x1c\xfd,<c\xbb\xf2\x04\xb3\x96\xac\xc1\x92\xb3\xaf\x02\u01c9\u015c\x99!c\x03\n\xa5\x
fd
\x99moEb\x99&\xe4\x0f\x01y\x92\xa3B\xc7rd\xb8m\f\xde6E\xed\xf6\xfb&3cb\x82\x1f\xf0$`G\x06`
@*F\xb5\xae\xa1\x8c@R%PC\x01-
\x9dY\x06\x93i_9\x8ao\xbb\xfd\x8a\x1d;\xbc\a#\x87w\x81\xd9@P\x98\xef@\xa4@\xadV\xc3\xc5\x
17_\xfc\xec7\xbfy+\x00`\u0252%'Ug.*\x95(\xb0l\x8c\xce!]\xa2\x8dU\u04f9[W;\x0f\xb5\xc0\xf3\xc76\
u07ad}\"}\x9aRC\xa8\u020d?\x15B\x18\b\xd4j=\b\u00e0\xf4\nK\xa3\xbe\xe28F\x92(\x10Y\u0225\x
b7\xb7\x17J)\xe7\x88\xc8\x1dC\u0254\x8db\xa1\x14\u04f1K)[8\u02be\xef\u007f\xcdz\x9c\aY\xb1n
\x1f\x86\xfa\x18\xba\x94\x12\x86\x81J\xb5\x82\x9e\xbe^\xebvF\xf9\xed6\xd5}[\x0e]\xb0\xe7\xfbG
m\xde\x18|\xc2\nz\xcem*f\x98\x9a4U\xa6\v^\xde\xfe\xbe\u0465_\xc4\x14\x85\u007f\x1a[<\xc0\x00
\x891\x88\x95UkJ\x10ze\x13k\xfa\xf6\xa1\x16\xc4\xd88\xb1\x14\u00ea\u05f9\xbc\x1b\xb7\vt\x1f\x
c2\u01b2\x19\x10\x8ct\x16e\xce}\x90\x82\x00\xac\x15\x0e\xbd\xfa\x1c\x1a\x87\x0f8\x9c\xdaR\x16
M\x92@\xd6z\xb0\xe4\x82w\xa0\u007f\xee2p+\x86\u0400d\x82 \xe1\x02
\x02\x90\xb4!\x9dB\xdbB\x9e\x8es\njN\x9ad\xeaL\xd3\\\xe1\v\u01efK\x11\xf7\xfcMJ\xa7\xa7\x05\x
03\xb2\xe2l\x92)wo$r\x89G\xc6v\xed\xa4\x81\xb0i
Z\x06\xa6\xe9\x8c\xc9REQA9\x9a\xbaVr\xdbN\xc45\x92\xcc8\xb8\xefu\v\xb1Pj`g\xdb\u02be\xde^\
u031f?\u007f\xec\x82\v.\xfc\xe7\xf4\x1d\xbf\xe7=\u05df4\xc5\\\x000}}}:-
\x8e\xed\xc5|\xbc\xde@3V\xae[\xcco=M\x02\x86D1_\xf2-0O%\x17\xa3\x10\xc2
d\x9b\x93\xd2\xdb\u05cb\x81Y3;\xe0\xc3\x0e8\x91M\xc6\x18\x01r\xee\xb8\xefh\xa8\x94\xca>\xfc\
xdc\u03f4\x93\x9fL\xe19\x9d\x0e=}\x1d\xff\xb8\x96\rd\x02)\xa1\x12\x85\x19\xfd\xfd\x98;g\x00P\x8
9\x8b\xdc\xe0\xa3,\xac\xdd\x06\xa1\xd3h\xb0\x8e\x037\xb7;(ktea\x96\xdc-
3\u035d\xe4B<F9\x8d\xb1\xb3\x80S)\xc5q\xda\xfc{\xd7\xd9%\x86\xd1T\x06\xb1\xdbm\xba8\x13T
D\x8c\xe5=\aq\xc1\xcc\xedXP\x19\x82fF\x02@\x933\x81%\x06\xa4\xf3\x10\x97\x04\xd3\x16\xa2,\
xc3\b\xcd\xe1\xc38\xb0\xe1)\xe8\xa4i\xbblv\xbbZ\x15\xa3o\xd1r,\xba\xf8\x9dV\xc3n\x18\xc2\x10\x
88\x05\b\x02\x02\x815\x88r\xa6\xdfT\x12T\x92\xc1\xf2\x93\x14g
\xa7\xf0r74\x8d;\xa1\x19*\xdb\xc1\xa4\x04s\xf2\u03d4?\x11\xednKIi\u051bv\xac\x17{\xe8\x9c\xf2\
u050a\x8f\x82F\x02\xd9H\x80\x96\x8d]$\xce\x191y\x00F\xa7\xae\x98\x9d\xe3\xa8J\x1a\u063f}=\x
e2\x86sJL\xf9\xff\x86\xd1\xd3\u04c3s\xce9gd\xed\u06b5\xdb\x00\xe0\xa7\x0f\xff\xf4\xa4\xc2\xcc\x
05\x11M\x1c92\xf4Hoo_\xd61\n\x91\x9f\xacCC\xa38<\xdap\x13c\x91\x8d&\x99\x84\x93\xe4\x8a\x
c2BM'\xb8X\xa4\x97\x9f-
\xe8\x1aP\tzk5\x9cq\xfai\x90R\xba\x01\xc9\u4152\x99Q\xaf\xd7Q\xaf\xd7]\x81FV\xa8\xfd\xe7\xb4\
x17\u0763-
\xe4e_O\x17\x8a\xc9\x06\xa4B\b\x90\x10PJa\u1885X\xbel1L\x12\x17\xb2\x0e\xdbi}\u0736\x1b)D
V\xb6a\xaa\xa5\x93\f>\x91\xe7\xa73\x9b\x91\xdb
\x16\xf6\xfe\xc5m\x03\xb72\x04\x17m\xcf/K\x02\xa2)\x8azZ\x9bbePO\x14Z\xda@\xbb\xf0cCl\x8d\

xbb\x84A$\x13\x9cR=\x88\v\xfa\xb7\u151eA\x18\xa1m\xd6&\x19\x1b\xf8\xe0B\x1f\x8c\x042\u0751\vxf0\xf6\xbd\xf8(\x86\xb7\xbf\xe6-
\x1e\f\xa3\x15\x82J\x0f\x16_p\rj\xa7,G\x12\xa5\xbclw\x8f\x91\x84\x10\x81\x1d\xb8:\xb7E\x169\x13\xa7-\xdf\u00a5u\xf9\xd6\xc0T>\xbc\xf0W\xba\xec
\xe5]6u$\xf7\x96X\xb7\xb9\x00\xeaN\x8eZ\xf9\xcc%\xb5\x0f\u0228\x8b\xe9\xa03\xb5\xd9Mi\x96-
\rn%
\xa5A\x89\x06%\x16b\u024c\xd4R\xa5g\xa7\x8e\xdf\xd5&\x89\xa1\xfd[ph\xdf&h\xadlg\xee.~!\t\xb5Z\rk\u05acy*\xfd\x99w\xbd\xf3]'W1\xb7\x85E\xef\\\xb2d\x91\x17\xb8\x90\xe3\xd1\xfb\xf6\x0fb\xf7\x81C\xa0 \x82f@\xb1\x80rFF\tK/\x1c\xf8\xd8\x13\xe5\xa7[\xd0\x05\x18P-
\xf4\xd6B\x9c\u007f\ue648\xa2\bR\n\x04%B\x9e\xb2!c\x8e}s)~\xcd\x05Iq\x97\xf72I!?\u0582/R:\x90\x90Xu\xda*\x9c\xbax\x81\r\u0560\xce>\x94\xa7\xdeEg\xf1_\xed\x16\x00\x85e\xe1-
ak\xe5%\xbbH\x85\xccc\u0734o\x8cL\xe5FbT\xc0\xf6\x8b\x85\xbf\x9dr\u074d\x9b\x9e^\x8dqb\u040a5bm\xa0a`\x88\xed \xd3a\xe0\xf6s\x03\"\x8d\xb9\xd10\u039e\xb1\x03\xa7\xf6\f\x02\xd2
\x11\xd6\x02V\t\x82\x16\f\x0e\x8a]\xb2\b+\xa8\x1f\u0687=\xcf=\x00U\x1f\x87\x90\x01RoXf\x83\x9e9\v\xb1\xe0\xbc+Aa\bE\x06p\xf3\x13!$\u0205@\x93\x8b\xe31.~\xad|\x97EE8\xa2D$D\xe8\x92\xcc\xc8\u0716\xe1\xc6m;\x1f\x9e|\x95\u6a7e\x99/<\xa9\x17LZ\xc8ar\x0e<\x19\x9d\xd9\x05\xb31\x99\x0f\xba=^\u02024\xc8[\x10\xb3\xe1gz\x92\x05ap\xf7\xabh\x8c\x1d\xca-
o]\xf7\"\x84\xc0\xa9\xa7\x9e\x8a\x8b.\xbeh\f'\xe9C\x00\xc09\xe7\x9cs\xa8V\xeb\x19I;]\xbf0\xed\u067d\x1bol\xd9\x0e\x11F\x0e\xf2\xb0)6\x99y\x0e\u02b9\xe6o\xd5@\x94\xb5BU\x18\x9c\u007f\xe6J\xac\\\xbe\x14Z\x1b\xd4*\x95\fC\x9f\x0e\xe3\xa4\x1d\xf2(\x13\xfc\xb4\x1f\a\xff\xdf\u077a\xff\xa9\x86\xa4\xed\x8bF\xf6\xbaB\xa0\u054a1{\xde|\xac\xbd\xe4\x02T\xa5U$\u0494\v\\\xb7\xed4g[SP;^z\xfcA!\u04c3y\xb8\xa3\"p\xb7\xad{\x17\u0225\xd0\xd9Sw\xbdj\xc7\xf2\xec<TZJ\xa3\x99h\xc4\xca\xc1e\xf0\x9c\x18\x05`\xa4\xfd\xdc\x00\xd0d;\xf6\xd9\xc18\xce\xed\u0749\xd3\xfb\xf6AV\f\x1a2\x80\t\x00\x13\x10L\x90\xd3\xe0HH\x18\xad\xb1\xeb\x89\x1fbd\xfb\xeb\xa0
\xcc0v0C\x86\x11\xe6\x9cq!f,Y\x05h\xab\xb2\x04\b\x90\xd6\xdeV\xc8\xc021&\x9b\x00\xfb\u01e0l\u0562\x1cb\xf1\xff\xdf\xf9ze\xe9\xbcT\x84E\xb9K\xe5\xa6.\xa0V\x99\xa4\x97;M\xbe\xd2,Tc48\x8d\xd6\xf3\xa0\x99\u0531\u04a43\n\xff\x85\n\r\x02A\xab\x18\u00c7\xb6\xa3\xd5\x18\xb7B!\xb2\xac\x190#\x10\x01.\xba\xe8\",Y\xb2tN\u0588\x1e\xd8\u007f\xf2\x15\xf3\x9bn\xfa\u012e\x993g>_\xadV\v\x85\x8b\b0q\v\xeb_y\x15\aG\x1a\x90A\x94y@\v6\x1e\xa6k\xcf\xde[\xc9\xcd'\xef\x06O\x9a\r\xacX4\a\xef\xbf\xf6\n\x1b\xfe`\fz{z\nR\xfb\xfco\xa0)\vy\xd9\xd7\xcb\x16\x81\xe9`\xe4Sa\xe9e~.\x96k\xaeq\xf6\xd9g\xe2\xf2\v\xdf\x06\xd5j8/\xe7\xee\x1b\u074ecCE\xf7C?\u00ac\fu~\xabe\x14\xbe\xea\xb3\xe8\xc7RfH\x8bB\xf9.\xff\x9by\xca\xe3\x90b\xf2\xc6\x10Z\x89F+\xd6HT\n\xa7\u5fd7Hd\xfe\x1e~\x97\x9f\x9a\xc6\xf5\xc9&\xde\u05b3\x1bg\xf6\xecA5J\x90\x04\x12:\x12v!p\xe7KTj8\xf4\xfa\v\xd8\xfd\xf4O\xa0\x93\x96\x1d\xcei\x9d\xe5\x99\xd6f\xcf\u01d2K~\x01\u044c~p\x1cC*\x9b\xedIA\x88
\xeaAX\x9b\x81
\xea\x01U\"\x10\xdb\xfc\xcbT\x00\xd5^c\xb9\xc4\xd1\u0187\xc1}\u031b\b%\xfe\xc4\xed\u017bDB6i\xf1\xc7\xf4a\xba6a\x13ke\xad\x12\xc8\xd9%\xb8\xc5.\xb3Vg\xbbS\xe2\xc22OY~if\xeb/\x04\x9a\x8d\x11\xd4G\x0f\xc3h\x95\xeaNA\xb0\x011Q%\xc2\xd9g\xbf\r\xbd==k\x98\xb9\x17\x00\x16-
Xx\xf2\x15s\"\u06bfx\xf1\xe2\x17\x97,Y\f\xad\x15\xa7t\xba\xf4\f\xbf\xf0\xc2\xcb\u0630y'\x82\xde>$\x06.C\xb2\xbd{x\xeb\x1f\xe9\xefH\x92\x04Q(q\xdd\u0557\u0f37\xadF\xbdaC\x97\xa3\x8c\xe125d\u04ad\x90\xfb?\x9f\xb2QR\xdf\xf22\x1c\xbd\xfdg\xa7\xea\xc8\xcb\x16\x0e\xa5\x14f\u03dd\x8b\x8f\xdf\xf0n,\u83e0\x9d{e7Z_g!\xf7\x90f\xf2h\x89\\\x84Zh:p\xcdQ\u0737\u075e\xe7g\xc3\xf8\v\x89\x8

1\x80\xcaF\x8f\xd41\a
\x94\xd1]=\xdf\xf16\x90\x80J\xd0[\u0144\x96\xd2h%\x06\x8960i\xf7\xe7\x89hR\xf4\xc1\xea(\x9c\x96\xc2\xc0:\xfc\xb9\xc56\"\x85\u0555}x[u\x17*Q\x82$\xb4\x8d\x02k\r\x11U\xd1\x1c9\x82]\xeb\xee\xc5\xc4\xe0\x1e\x90\b\\\xa1\xb2\x1d\xa5\x90\x01\x16\x9c{\x05\x06N?\u07fe+e
\rA\x86UT\xfaf\x81a0\xb8\xfd\x15\xec\xd9\xf2\f\xc6G\xf7CBXO\x16\xe7\u007f\u4670\xa3\x8c\x8a\x98\r;}\xaf\xf7)\xb8\x9e\xed;\"\uee8b\x9atDZ(\xd4\xe5\xa3j\u01ddJ\x83\xa6\x8d\xa3x:\x98\u02a4\x10Jz\xdc\xe1\xabx\xd3y\x1c\xb2\x9cU\x16\x9cE\xf0%q\x1dq<\x81v\x034f\xc6\xec\u06730\u007f\xfe|0p\xca\xfe\xc1\xc1\x15'%\xcc\xf2w\u007f\xf7\xb7\x11\x00\x9cy\xe6\x19\x8f.[\xb6\f\xb6\x86\x11\xfb\xc6Q\x87\xf6\xec\xc4\xf3\u03ff\x88Fl
d\xd0&\b\xe1\xd2\xc1\xd4[]\xd0\x1b\xf5:V\x9f\xba\x18\x9f\xb9\xf1}\xe8\xed\xa9adtlZ\x01\x14~\x10s\x19\x9c\xe2\x17\xdd\xf6\xe7v{\xbd\xa9:\xf5n\x98\xbaOO\xfc\xf0\a\u078dw\xad=\x17\xbaU\u03feF\xd3=\x9e\x85y\x11\xb51\x02\x90m[\t\xc7&\xe3'\x0f~\x9d\xee\xf3E\xa1\x88\x13\xb4#\xfc\t/\x98\xba\x9b\x11\x17M\x8d\xd2vl\xfc\xc9)\x00\x13\xa5\xd0J4b\xcdP\xa6\xe8]\x9e{\x85pV\u0233L\x06\a\a\xe4\x9f\x03\x01i\x9c\x1a\x1c\u009ap?\"RPL
\x19\x02`\xecy\xea>\fn|\x06$\x84\x83\x0f8\vH\u8677\x18K.{/*3\xe7@\xc7-
\x04\bP\xad\xf5\xa3\xda7\x80\xc3{7c\xddm\u007f\x8cG\xbe\xf3\xdf\xf1\xcc=\u007f\x81W\x1e\xf8&\x0e\xee\u0610\xf9\x9bP\xfa\x9e:'\u02c5\x93g\u00d7)\xd3\xf90g\xc1B\xc5\xed\x19O\x1f\x18+37\u0387\xa2\xed\xab\x03\x95\xee\xf4\n\xbf\xde0\xd8\x1bhZh\xcb\x0e\x94\xdd\x06\x06B\n\xab\x1b\x91\x00\x02\xfb|f\x86\xce1\x00\xb7x1\x94N\xa0un\xbf\xe0\xdfS\xbd\xbd}\b\x82\x10\u0198\xc0\x18\xb3\xf8d,\xe6\xc1\x19g\x9c\xae\x00\xe0\u04df\xfe\xe5\xd7\u05ed{\xfc\x95\x81\x81\x81s\x87\x86\x86\xb5\x10\"\xf0\xb9\u044f>\xfa\x04\xae{\u05d58g\xc5B\x98\xc6h\xc7\xcdM\xc7\xd1\xd1\x1d\xed\xf6=\xbd\xd0T\x92\xe0\xfaw^\x867\xb6\xec\xc07\xfe\xe5.\u0109\xea\x1a\xbe\xdc\xceR)+\xc8e\xf0J{`\xc7d]\xf7d\xde\xe6e\xbc\xf5\xf4\xf3+\xaf\xb8\x14\xb7\xfc\xe2\a\xd0\x1b0\x92\x89\xb8\xa4\x93\x9a^\x05M3\x1f\xb9\xacu\xe5Ni\xfdQ/\xba\u053dw+{rj1\xe2\x17x\xea\u03a1(_@\u0494\x9f6V\x8f(\xbc\x0f\xb2\xb1kJ#I4\x942\xce\xf8/}\xae\xb7#1\xb9\xb29+\xdc\u0739\xb0\xd8\xee\x92Q\x81\xc22\x1c\xc6(j\xd8&\x17\x82\xc3\x1a\x06\xd7?\x86\xed\x0f\xdf\x01\xd5\x18w\xb0\x8e\xce~JFU,\xbd\xecz\xccY}\x1e\x8cJ\x10\xf5\xf4\xa3\xda\n\x11\x1f\x1e\u0096\xf5wc\xfdO\xbe\x81\x83\xdb\xd7C\x06\x15\x04\x95\x1e\x1c\u067b\tZ\u0168\xbd{.f\xcd\\\x04\xc4-
\x97\r\xea\x18\xd7e\x91vm\xc3\xc2\x02\x15\x9b\vzN\x97\xf9Yf\xa4\xc0%M\xb9\xe7\x8cE\xed{8*\xc1\xdcK5\xa4\xb9\xf5\x1e\u5db4\fc\xbf!\xad\xb9\x98\x88\b\x81\x04\xa4\x84\v\x87\xb7_W*\x8fLd'\xf6\x82\xb1\xdd:\t\x99\xbd\x96/\xf3\x17\x82\xd0l6\xa1\xb4\x02\x119\u01d8\x93\xb03\u007f\xc7;\xae5\x1f\xfa\xd0\r\x92\x88\xde\\\xbdz\xf5\xed\xa7\x9d\xb6\nJ%$\x84(\u063an}\xf5U<\xfe\xf4Kh*\x03\xc8\xc8c\xb1\x14/$?\xaa\xeb-
\x85\\\xc8\x0eDf\xd4*\xf8\xfc/\u007f\f\x1f\xff\xe0\xb5\x16\xcb\x14\x02A
\u074a/\xa6\xd5A\x97\r?\xa7\xd3}\xd34(\x91\x9d\x03R*\x18\xef_|\xd1\xf9\xf8\x83/\xfc\x1aV.\x99\x87\xa41\x81\"1\xef(\x8e\x87o2W\xb2\x9aNWV?9\xccB]Xp\xd4uqH\xa5\xfd\xb2\x00\xa1\x14\xa5\xfe<\xe5\xfa\xc1\x93\xa2\xea\x86\x19\xb1\xeb\xc8\x13m\xd0NXJ\xc3s\xb4\xb1\xf4S\x93u\xe6^A7\\$\x8b8\x88J\u00a0\x0f-
\xac\xa0\x83X\x1c51\xba\xf3\rl\xba\xf7\u007fcl\xdf\x0e\x80lt\x19\u0600\x84]\xac\xe7\x9f{9N\xbd\xe6C\bzf
\xa8\xf6@\xd7'\xb0\xfd\x99\x1f\xe2\xfe\xff\xf5y<\xf0w\xbf\x89\xfd[\x9e\x87V\tT\xdc@c\xf4\x10\x86\xf7m\xc1\xf6\x97\xeeG\xfd\xf0V\xf4\xf5VQ\r\x05\xa2\xc0\xfa\x10Q\x19w\\x\xe1q\x18\xd1\xfd\x00\x

00 \x00IDAT\xd7\x1eP\x9e\x96\xaf\xa2-
.\x95br\xdc\xd1aSgW\xc0m\u06c2\x12\xdf\x00\x9f\x19\xd6\x16\x14W0@3\x86\xf3\xfd\x18\x01\x14\x10\xc2~\x89hN\x80\xb0*\x10HB
\ba@\x90!\x15h9\xe9.\x93\x89\x11\xd5z\x11Vz\xbc\xeb\xdc\xcd!H`hx\b\x87\x0f\x1d\x063\u01f5j\xed\xe0IY\xcc\x01\xe0\xf3\x9f\xff\x1c\x03\xc0\x97\xbe\xf4\xfb\xff\xb0`\xc1\x82W\xe7\u035b'\xb5\xd6I\x8e\x9d\x03\x80\xc6c\x8f?\x8d\xbdG&\x10Uk@\u6652\u039a\xf3<A\xf1\x16\x16\xf4\xc2n\x93\b\x8dF\x03\xf3\xe7\xcc\u0117~\xe3S\xf8\xc8{\xdf\x01\xad5\x946\b\x83\xa0\xc0\x97'J\x83e\xb9k!\xf7\xc39\xa6b\xaf\xf8,\x98\xc9\xf0\xf9\x82\x0f\xb3\x10\b\x03\x99\xdd\x17\x97^|\x1e\xfe\xe8\xf7~\x1dg\xaf>\x05\x8d\xd1!\x90N\n\f\xa1\xa39\x16>\x83\x8f\xa9[\x97{l\xbb t
\xa3\x9dI\xa2\x9d\xc8k9y\x90\xc0^h\fO\xb9\xab+[\u0532m<\x01\x86\xad\xaa3Nlr\x931~\xa2\xbbU\x19\xb2a\xe8\xf4\x1c\xc1\xb3d5\x1e\u0312\x0eA}\xa6Qv\xae\x193\"\u00bc\xe6~\xec{\xe4v\x1c\u07ba\x11B\x06\xd9@\u03fe\x9e\xc6\xccS\xd6`\xcd\an\xc1\xc0i\xe7\u00a8\x18\a6<\x85\x17n\xfd\n\x1e\xfb\xdb\xdf\u014e\xf5\x0f\xd9\u008f\xd4w[g\u007feR\x1fA%\x14\xa8\xd4$d\b\x04\x92\x10\n\x82\xf4\xb8\xe4)\xcf=KXB.2b?j\u03c3\xda\v;\x9aB\xc25M\r\xb3L\x87\xcd@\xe5\v.\xda\x03\xa6\xdd{c\x00$\t\x95\x9aD83\x80\x9c\x19@\xce\b@\x91\xb5\xb0\x15\x82
#\x01\x11\n\x8b\x97S\xbe\xa34l\x10\xf5\xf5\xa3\xda7\xdb\t\x8cR3\x17\xfbs\xf5\x89:\xb6n\u074aV\\xb3\u065a=k\xd6\xf0I[\xcc\xdf\xfb\xde\xf7\x9b\x83\a\u007f\x06\"\xdaw\xdey\xe7\u07fex\xf1\"\x18c\x82\xf6\u0405W_Y\x8f\xf5ol\x87\x96\x15\xb0\x88\n\x81\xc3\xfe\xb0\xe6X\xb1\xd9c\x83]\x18\x13\xe3\x138e\xe1\\\xfc\xe1\x17o\xc1-
7}\x00RJ\xb4Z\xb1\u06c2\t\xaf\xf0\xa2t\x88\x99\x16\xe70\f2FL\xfa\x9ct\bZV\u0627\x1b\x19GD\b\x83\x00Q\x18@\x1b\x03!%\xde\xf7\vW\xe2\xff\xfa\xaf\xff\t\x17\x9d\xb9\x1c\xad\xf1\x11\xc0$v\x11\xe4\xe3\xc8\xfa,\xe1\x06w\u04d3L\a\x06\xa3\xd2\r\xb6/\xe4!\xb4k4\u065b\xa4\x14\xf0m?K\x93\xa6\u0388\x9d\f\xbeI_[k\xb6<\xf2\xc4@i\xce\nq\xdaYg\x10\x8a\xd7yk\xb6\u0676\x99\xba7\xed\u0239\x04KwEM\xca\x00\xa1\x14\xd8\xf9\xe2:\xecy\xfe\xe1\u0316\u0546Y\x18\xb0V\xe8_r\x1aN\xff\xe8\xe70c\xc9*\xecy\xe6\x01\xfc\xec{\u007f\x8d\xe7\xfe\xd7\x1f\xe0\u0347nCc\xfcH\xe1\x04\xa5@Xz\x8d\xac\xb9\xecz,Ys!\x94j\x81\xa2\x00\x14\x10\x84\x04Bi=\xd2\xf3\x18{r\xd0\x05\x15l\xe3\v\xd4\xe0\x8ek\xa0\x18p\u0285E\xb6\u06d5Q\xb6\x9cs\xd7U\x9e\xdb!3\x8f\xff\x9d\x82\xf9i4]\x10\x10\xc2^\tY\xb5\xa1\xe5\xa8\x12\xb8O\x80+\xb6\xe0KA\bB\x02\x059L\xc3l\a\xcf2\xac\xa0\u007f\xeeR\x04Q\u0545h\xa7\xf5\\@%\t6l\u0600\xf1\xf1\xf1\x04@|Rb\xe6\xe9'\xf3\xe6\x9d\r\x00\xf8\xf2\x97\xbf\xfc\u05ef\xbc\xb2\xfeS\a\x0e\f\xae:x\xf0
\v!\xc8hmW\xc7\xc68\xee\u007f\xe0a\\\xfe\xf6\xf30/\xea\x81i$n\xa8E\xe0\xe3\b\x1cFigw\x14\xc396\x18\x9b\x98\xc0\xbc9\xb3\xf0\a\xff\xf9f,\\0\a\xff\xf8\xdd{\xb1\xef\x80\xddm\x05\x81\x8df\xb3\xa2!\xd3u\x90g;;sTC\xce2\x1c<\xe5\x8e\vB\xb6c\xd1Z#l\f\xe6\x0e\xcc\u00a7>\xfa^\xdcr\xd3\xf5X0w6\xc6\xc7\xc6\xf2N\x9c\x8fn\xd2\xd0\xce5\xeff\x99;\tb\xda\xf5\x98\x97\ty\xca;u?\x8a-
\x85[\x8aT\xc0b\x01\xe0i_\v(\x81s\x00\x02\x1b\x03e\f\x942P\x86a<\xd6\n\xbcn.\xeb\xb8\xfd\xee\u06dd\u007fk\x16\x87,t\xc5\xd7\xe5\x14\xf1X\x81J%\xc4+?\u06c8\xc7~\xf2c4\xc6G\x10\xc8
\v\xb0\x00\x00Y\xa9\"\xac\xf5\xe1\xf0\x1b/a\xd7\x13?\u0111-
\xaf`\xfc\xc0.\xdb};\xac\x97\xbd\xa0\x8a\xec\x98\n\x81\xb7\xff\xe2o\xe0\x1d\x9f\xfdo\xe8\ud7cf\xf8\xd0((\xb0v\xbd\xc4\x02\x82\r\x02\x10X[H(u\xea\xed\xd8y\x15`\x95\xfco\xf1=\xc8;\x05}\xdcf`E\xe8n\xb0P\xb6?\x9b\xfc\xdc1\xe5\xda\x02\xe3\xbar!\x80\xa0B\x10\xbd\x02$`=\xd0\xc1@\x85l\xe6\xed\x98\x01\x9a\x1a$\x800
\xb0\x01\xb4rW\x98]\x851g\xe9\x19\xa8\xf6\xcd\xc2\xd8\u1f50A\xba\xc6\x11\x12f\xbc\xfe\xc6&l\u

07f1#:\xf3\x8c\u04eb'mg\x0e\x00_\xfb\xda\u07e4\xc5\xe9\xc85\xd7\\\xfd\xed\u014b\x17AJA\x00X
x\xfc\xedg\x1f_\x87\xa7\x9e\xdf\b\x84\x15@\xe4~-
\xbe\a>\xb7\xc1q\xddvle\xf9\x90|\f\x85,\x1d\xfc\x8d\x8fO
\x8a$>\xfb\x89\x1b\xf0\xe7_\xfe\x1d\\\u007f\ud568U+PJ\xbb\xf0e{s\n\xf7\xc6|(\x85\x99\v\xde,\x19\
xde\xea\xbc]\xcaR\x84\xca:\xf0J\x14\xa1\xbf\xaf\x17\xbd\xb5\n\xc2@\xba\"n\xe5\xfc\x97_|\x0e\xfe
\xe7\x1f\xfd\x16~\xe7\xd7n\xc4\xfc\x81\x99\x18\x1f\x9b\xc8\xfbL\xe6)C\xe1\u0288\r\x93A)e\x86\x
a74\xc5\xebLVP\xc9y\xae\xf8\xc3\u0254f\xa8\xb3\xef\x14\u007f\xdex\xbd4\x1d\x85B\xb8\xdb{\xd
3\xc6
q\xb0J\xa2\f\x8c6\xc5\x01&\x17\xab\x9d\x0f\xa1\xd8N\x1c\x9e\xab\xa4\xbdn\r\xdb@\x04\xbf\x90\
xb3[\x84\xab\x95\n\xb6\xed\u0703{\xee\xfd\t\xf6\xee\u0647@\x06
\xe3(\x8c\x9c\xefE\x87\xb7\xbf\x8a\xd7\xef\xfa\u007f\xb1\u3c7b1\xbe\u007f\xa7\xed\xf0\xb5\x82N
b\xdbE\xb6\x1d\u0359\x8b\x97\xe1#\u007f\xf2\x0fx\xdf\u007f\xfd\v\xcc^\xb1\x14\xaa\xd2\x02\u03
4d\x80>\t\xaaJ\xa0\"\xc0!A\b@z\xd3^r\x01\xcb\xd4n\x8a\xc5\xc5\x15=\x8dQ\xeb6-
\xe1\u0098\xbam\xc8\xc9\u076e\x94\xac\x15n[`\u06f6\x87\xd9\xf6\x8b\xac\r\x88\xf3\x9e\t#B8#\x8
0\xac\x884\xe7)_mC\x02f\x04\xe0\x1e\t!\tR\x10\"ae\xfaLn\x80\xaa\rf/X\x89\xdeY\v\xb3\xc8>\x9b\xf
4d\x11\x84\x83\a\x0f\xe2\xd1G\xd7U\x01\xcc:\x19\x8byV\xa5\xef\xbb\xef>\xe4\x9f\xff\xf8\xf9\xe7\
x9e{\xe6\x86\u077b\xf7\u032f\xd7\xeb$\x84p\x1d\f\xa0[M(\n\xb0\xf6\xca\xcb\xd1\x1b\x00\xa4Z\xb
9\n\x14>\u038cLxT&\xbb\xe6)\x98\f\u01c2\xa5\x83\bIb\xb7_+O]\x8aK/>\x0f+\x97/\xc5\xd8\xc4\x04
\x8e\f\x8d\xa2\x15\xe7\x9e\xe4\x16O\x9fj/P\xc4\xcdE\xdaq\xb7a\xf1A
\x11\x06!zjUDA\x00\xa5\x12L\u051bH\x94F\x10\x068c\xf5r\xfc\xc6\xcd\x1f\xc1o\xdd\xf21\\|\xee\xe
9H\x92\x04\x8df\xcb.*<=4{2\xfa\xe7\xd1`\xe2t\x14\xc7sRn{\x16\xea-
\x9c\x8f9\n\xe5\xc1\xef\u0667\xc2\u01e7\xfb~\x94f$Z#Ql\x87\x99\x1e\x94\xd2\x0eo\x15\x98*\xe9\
x87\xc9\v\b\x15v]T0\xb8\xb1\x9d%\xa1Z\xaba\xff\xc1#\xf8\xce]\x0f`\xc3k\x9b!\x88\xa0\xb3\xc5\u0
763X\x1a\r\xa3\x14H\xda\x0e\xd3\x18\x03\x9d\xb42\x8c\xdc\u007f\x84=3p\xce{o\xc2\xf5\xbf\xff\x
e78\xfb\xba\x0f\x00\x14
\x9e\xa8[Ec(l!\x0f\xc8S\xef\xe7<x\xe3\xe9\x06:\xf4><\xd5\xe2\xdc\u05a3\xfb\xdc\xc7\u051d\xd3\xf
7\x04\xa0N\x02{\xd1\x1e\x82\u0684`y1OU\xb6
\x82\x11\xd6\a'\f\x80\xea\x8c\x00\xb2O\u61da\xdbV[\x01P(l=\xf1h\x9a\xc6\xc1a
BT\xe9\xc1\xc8\xc1\x9d8\xb8s\xa3\x1b<\xcbl\b\x9a$1\x94J\xa25kV?}\xeb7\xbf\xf9\x12\x00l\xf8\xd
9\x06|\xfdo\xbf~r\xc1,\x00\xf0\xcf\xff\xfcO\xf8\xccgn\x01\x11\x8d\xddy\xe7\xbf\xfd\xfe\xe6\xcd[~8::\
x9aejjG\xd3{\xee\xc9'p\xff\xe3\xd7\xe2c\xef\xba\x18\x11\xd5!\xd8\xf6h\x06V\x19\xea|z\\\u03ae\xb
d\xf0L!\xee\xc4\xcfvo\xff\na\xf2h\x94\x92A\x98\xf7oA\x04\xc3\x1a\xcdF\x1ds\x06f\xe1c7\xbc\x1b\x
ef\xb8\xe2b<\xfd\xdc+\xf8\xc1\x8f\x1f\xc1\xc67\xb6\xe0\u0411a\xc4q\x92]\xb3\xa1sS\u0303c\xad
C\x9b\x10\x02\x81\xb47g\x10H\x84a\x80@J0\x03\xcdV\vq\xac\x1c\xceg\xcbV\xa3i\vx\x14\x04X\x
b0`.N_u*\xde}\xf5\xa5x\xe7\x15\x17\xe2\xd4Es!\x880Qo@+\x93\xfb\xb1\x1cC!o\xff&s\xf7\x8ez\xb
a\f\x16\x9e\xe2y\x9d\xb4F\xfb_\xe1\x06\xe1\x94\xf5\xe5m\xbcc\x86\xe7\x13s\xec\xf0\x9b\xd2\x06
q\xa2\xa1t^\u023b\xc17&c\xaaX\x9a[\xb6\x8bB\x1e\x05\xc7\xce>\x04\x19\x14\x96\xe3\u0646\x81
\xac`\xdf\xc1a|\xef\xae\a\xf0\xcc\v\xafd\xe5Li\x95u\x85\xfe\xae\x10\xb0\xbb\x04\xad\x94\xe5B\x1b
\xed\"\xcel\xd1\uf67b\x10\xa7]\xf5>\\\xf8\xe1_\u018a\xf3\u05e2\xda\x17A55t\x12\u06f8\xb8\xb4\xb
0I\x003$\x10\x11DU\xc0\x8c\v\u0206\x06\xb7\fLb\xe07\xf9T\xb0\x9f\xf5\x8ac[t\\\xc7}\xd7\xee\x91\
xce\x00\x89\xa2V\x81\xb8\x1b\x82\xce\x1e1\xb1Dn\xeae\x86\u06a2.\x11\bF\xb5F\x90\xfdd\xff>\x
c3\x1d\x17Zv<%\x03}\xd2>o\x9c\xed.\x84\xac2\xd7h\x8d

\x8c\xb0\xec\xac+\xb1m\xfd\x03\x18\x1f\xda\x0f\xe9\u03ad\x10\x84\xb8\xd9\xc4\xe0\xe0
~\xfa\xf0#\xb7\x00\xb8\x15\x00\xce9\xfb\x9c\x93\xaf3\a\x80;\xef\xbc\v\x9f\xfe\xf4\xa7\xe8\x95W6\
xe0{\xdf\xfb\xfe\xe6\x9bo\xbe\xf9\x8a\u077bw\xaf\x1a\x19\x19q~\xdb\xf6\xc0\xa9V\x03\a\x87\xc7
q\xee\x05\x17`\xc1\xac^h\x15\xbb\x9c\x14\xeeX\xa9\x05\xa5\x1d\x1c\xa6t\x05i/\x06G\xd3E\x16;t\
xbb\x93h%\x1a-
C\xe8\xef\xeb\xc1\x19\xabW\xe0\x1dW\\\x84\v\xcf9\x03\xf3\xe7\x0e\xa0Z\x89\x10\x86\x12Ji4\\\x1
a\x11{\x16\x9c\x96yb\v\xba\x10\xd6\x14\x89\xd9
I\x14Zq\x828I\x10+\x05\xad\xed\xb6?\b\x02,\x987\ag\xaeY\x89\xf7\\{\x05>\xfb\xc9\x0f\xe3W?\xf9!\
\v\u1658\xdd\u07cb$\x8e\xd1h\xb62\x95
xz\x1e\xe54I\xa7\u0716\xa8\xd5q\x1c\u02e4\xf1S\x15\xf1\xc9\xf6+T\xb6s\xc9(\x87T\n\xed\x1co!\x
87W\xc8\x13e\xa0M\x1b\xe5\xd3\xfd7E\xc7\xd8%c\xa5\xb4C\x98\u0734)\x03\fr\x97\xd7\xccy/\x9f\
xd9\x11\u00a8\x82##u\xdcv\xefCX\xf7\xf4K0ly\xeb\x16.3\x05\xb0\"\xdd\xea\x1b\xada\x8c\v\x84v\x
1d#\xb3\x81\x88jX~\xcd\rx\xe7o\xfd\x9f\xb8\xf23_\xc0\xa2\xd3V@\n\x89\xa4\x11\xdb\u039d\xa8\x
b0\x9b\xc9p\xfb\x80@U\x01\x11\u065d\xa0$B@\x94u\xab\x85\xf9J!\u05c1\xa6X\xa2i\x92\xe99O~
\xc6\u067f\u6f06\u0311\xde-\u00db`\xd2\u0667$
\x14\xa8V\tQ\xbf\x00E\xe4\xe6\f>\xc3\xc6g\x0f9\xf5\xa8
P(\xec\xb9L\x87\xdb\xee7J)Q\xe9\xed\u01e1]\x1b12h\xe9\xa1\xf6\xfc\x11
\x88\xe3V\x93\x84\x103\u05ad{\u426f\xfe\xf9Ww\x01\xc0\x0f\xee\xbe\v\xdf\xfd\xcewO\xae\xce\x1
c\x00n\xbe\xf9f\xfe\u05b7\xfe\x15\x00\xf0\xb1\x8f}\xf4\xcb\x1b7\xbe\xbafdd\xe8\xd4\xf1\xf1\t\x04\x
1c
qE\xef\x8dg\x9f\xc4\xf7\xff\xed^,\xfd\xf5\x9b03\xac\xa2\x157\xec\x1c\x03\xbap\xf2\r\x8a'?s='\x1f\x
9f+\u0798\xdd:\xef)\a\xa1)\xae\u02e9\u007f\x8cF\xac\x12\x8c\xaa\x04\x95\x00\xa8U+\xb8\xf4\xa2
sq\xe9\xc5\xe7ahx\x14[\xb7\xed\u0126\xad;\xb1u\xc7\x1e\xec\u06bd\x17\x87\x8e\fatl\x02\x13\x13
\r\x8c\xd7\x1bh\xb5b(\x13g7\x8f\x10\x840\fQ\xabV1\xaf\xaf\a\x03\xb3\xfb10{&\x16\u039b\x8b\xe5\
xcb\x16\xe3\xf4\xd5+p\xc6i\xcb1o\xee\x1c\xabtT1Z\xcd&\x94\xd6Y
62\xc6\xca\xe4\u007fYG!\xa7\xa2\xa0\x8f<:0g\xdd9\x15\xc6ZeCOB\x99@f\xfa\n\xcc\xf4sS\xe2\f\x
95E{u\x89\x8c;\x96G\xa2\r\xe2\xd8
q\x1dy\n\x9f\x14\xf8\x17\xe9V\xdc}O\xbb\xffS\xc7\xd5d\x1c\xd4R\x98\x9e\x17p\xdeZ\xa5\x82\xfd\
x87\x86q\xfb\xbd\x0fc\u0753/d3\x14\xed\xf2_\x89\xf2\xe0\u2b10\x1b\r\xa3\x13h\xad\xb2\xfc\\f\x8d
\xfeS\xcf\xc4\x19\xef\xff4.\xfd\xc5\xff\x84\x85K\xe7\x80\x12
\x99HlxEV\xccr;Z\xff\x9cQ\u0298\xe9\x11\x90\x01A\x84\x04\x1eW
\xc1\xe0\x86\x81J\xf2\xa2^\x9e\xa1\xcd%g\x87\xbao\xb9\fw\xf2\x89\xdb\x02*\x88\xbaL\xb5\xc8\x
19\x1e\x93o\xfae\xb3R\xab\x91@\xb5_\x02\xbd\x1aF%\x85\xd0\xed,!(C}(\xd7K\x10@\xbd\x02\xa
4%\xc8%/13\x14+D\xb5~\x9cr\u0595\u063b\xe99\xa8\xb8\t#\x04\x041\xa4\x1046>\x81}\xfb\xf6\r|\
xeb[\xffz#\x80'\x00\xe0C7|\xf8\xe4\x83Y\x00\x9b\xccq\xc7\x1d\xb7\xd3G?\xfaq\xbe\xf0\u008b\x9f\
xfc\xf6\xb7\xff\xe5\xbf\x11\xe1\xdb\x0f=\xf4S0\xc0ZI2\x8e_\xfb\xc0=\xf7\xe0\xfc\xb3V\xe1\x86k\xd
7\xc2\xc4\t\x88\x15$\x11B\xe8L\x05\xa6\u0754\xd5\x06t\v7\xec\xcb?'\U000b71b9Y\x97\x1f\x14>]
@\"\u007f\xbeM\x92\xb1#\xb9\x04\t$\x94\x02\x8cj\"q\x98w\u007fo\x15\x97\x9cw&.\xbf\xf8\x1c\x18
\xadq\xe4\xc80\x06\x0f\x1d\xc6\xd0\xe8\x18\xc6\xc6\xea\x18\x19\x9f\xc0D\xbd\x89X\x99lK/\xa5@
\xb5RAoo/f\xce\xe8\xc1\xbc\x81Y\x98;g\x16\xe6\xce\x19@\x14\x06H\x94\x82N\x12$\x8d:\xd8(\x1
0k\x14\x93QyJ\u018ao\xc91\x15\xbd\x93K\t0\x9d\x0e\xe8\x93\xcd*\xcaX-
\x8c\xa9\xb8\xdf\xe5\xfb\xa9N\\\x9f\x8f\xbf\x90;\x8f\x15m\x8a>+\xec\r\u074d[\xe5\xd8+\xea\xe9n\x

b0 i\u271cW`\u007f\x00\x10$Q\xadF844\x8a\xef\xde\xfd \x1e}\xeaE\x18c
\x85\xcc\x02\xb7\x995X\xbbW%\x8b\xe3\x1a60*\xb1\x96\xc5\xeeQ\u96cde\xef\xfc(N\u007f\u07e7
p\xfa\u06ef\u00ac\xde\x00z\"v\x05\x93;\xa9(\xdc~ls:!\x81\x80\x88\x80Y\x04\x84\x04!\x05$'\xd0F\x
83U\xdb\u0312\xf3\xb3GGA'`\ued01!t\xb2f\xca2~\xb3\xb6\xc4\x17$\x11\xa0%
#BuF\x88\xa0\x9f\xa0\xc1\x80&KQ4\\\xec\u0385\xb0Ceo\xdea\xbb'\x80\xfa\x02H\xc3\xc0\xb8\x86
\xd66\xf4\x03\x82\xb0`\xf9y\x985\u007f\xb9\xc3\u0383\xec\xaf\x0ed\xc0\a\x0e\f\xd2\xd3\xcf<\xb3\
x96\x99{\x89h\xe2\x99g\x9f\xa6K\u07fe\x96O\xbab\x0e\x00g\x9ey\x06\xff\xcd\xdf\xfc5\xfd\xf6o\u0
07f\x81?\xf1\x89O}w\u01ce\x9d\x17\xef\u0739\xf3w7m\xdaDA\x18\x98VK\v\x00\x18?\xbc\x1fw\xd
c\xf9C\x9c}\xe6j\x9c\xbeh&Z\xe3#\b\x1c\x13
\xed\x92\xd1\xfd\xb0\x02\xd7\xd1\x11\x15\xbe\u05ae+$.z\xefOBq\x9d\xa4(rF[\x94\x99\xb0\u0240\
xb4\xb6)\xe0\u0292Qc\x87\xeb\xcf\xec\x8d00s)\x02!!\xa4\x04\x89<\x88\u00f8\xad\xb0v]\x85a\x86
QvK\xad\xb5F\u0728\xa3U\xf7L\xf5\xc9\xce\x0e:\xcb\xeb\u051d*\x95\u031e|\xeb\x84BG\x0e\x94\
xda'\x91\x1757\x99\a\x13w\xe9\u0727\xc2\xdd\xfd\b\xba\xf6\xe1X\x99\xac\xe8\x98\n\xb9J\v\xb9v\
x18\xb9\x1d`\x1a\x8f\a\x9e\x0f=\xdb\xff>G\x93\xe4\x1c\n!/\u07cc\xbc\x95\x92a\x17\xf7j-
\xc2\xc8\xe8\x04n\xff\xe1#x\xf4\xa9\x97\xa0\x8d\x81\x14\"\u03cf\xd5\x1a:i\x02\xb0.\x9d\xa9\xb1\x
96\x0f\xf9\xf4\r,\xc6)\xe7\xbf\vK\xdf\xf9a,\xb9\xfc]\x98?o6\xaa&\x86i6\x8b\u0175=z\x8d\\0\xb2\xdf\
u9ca7\xaa5\xb0.\x8f}\x01H\n\xbb;PV\xcd*\xdc\\\xa0]\xb2\xcf\x05\x1f\xfb\x829BfN\xc5\x1e\xbe\xce
p\\u\xea\f\x15\xa4.\xe6l\xedWX\xca}\u05c2@\x01\xa1\xd6\x1b
\x9c\x19\x80\xab\x04\x8auf[\x9b*XS\xc8\xd1\xe6'pg\xdc)\xac\xc8H\xf6\x05\x16F\x9b\xb0;\\\x9d\xc
4\u86fd\x10\xa7\x9c\xb1\x16\awnty\xad\x96\xa5&\x03I\x83\x83\ay||\xe2\u04bb~p\xe7\xaf\x03\xf8\
xab\x17^xA\xb8\xbe\xf2d,\xe6g\x03\x00\xdf\u007f\xff\x8f\x03\"R\xcc\xfc\x95#\x87\x0f_1>>~\xe9\xb
6\xed;
\x85\x84v\x93\xfa\xf5\xcf=\x8b\xbb\xef{\x18\x9f\xbf\xe5\xe3\xa8Uj\xe0\xa4\x01C\x021\v\b\x17#f\x0
3~\xdd\xc5I\x1e\xc7\xd7\x15wIE\x8e:Q\x9b\xf7\x86\xe7?Q\xec:i\x12\x94\x8f3\x9f\x88\x00\xda+7E\
xdd;9\v\xce8\xd6\xe08\xbf6\v\xc6U~\a\x82b\xcee\xfb\xfb\x05\xf9\xdd
:n\x88\xc9\x17\x1f*\xfa\x17qI\xa2\x10\x8a\x05\x9e\x8fb\x91\xa3i\xfe\x9bJn_.-
\xf2\\p\xe9\xa0c\u0499vB>\x896\x88U\u0691[\u06a0q\xaaM\xe31\x1c&\x03\x17I\x01\xf7<N\xb2_\"
\n\u007f\xb7
B\xb5Z\xc1\xb0+\xe4\x8f<\xf5R\u0591\xb31PJ#\x8ec\x18\xd5\xc4\u0095g\xa1w\xe1\x1aL\x1c>\x8
0\xe6\xd8\x11\xb4&\x86\xc1\u0318\xbf\xf2\x02,\xbf\xf0:\xcc=\xedB\xd4V\xaeFu\xe9<\xf4\x06\t*I\u0
74a\xee|\xba\t\xba\x0f&\x88\xda\xf6T\xfe\xf7\x8d\xbb\xcej\x02\x8c\x10\xb1b\x18V\x88Z61\x89\x8d
\xdf\xf8\xe4\x05\xddU\xe9\x8e3\x96z\xb6p\xfb\xc04+\xe8e\x9e\x94\x93\xc08\xc2\xdel\xc6\xf9\xafT
{\x02Tf\x05@\xcd\xd2\x10!\x03\x18\xa1@Bg0hv\x93\t\xb8@haC+\x90[2\b\x02(\x12P}\x12Z\x19\xc
8&\u00e8\x18Q\xb5\aKN\xbf\x04\x9b\x9f\xbb\x17c\xc3\a]XE\xba\x83\x96j\u01ce\x9d\xe1\x03\x0f<
x9\x80\xbf\xfa\xcd\xcf\xffg\xbdc\xc76:\xf5\xd4\x15|\xd2\x15\xf3\xf4\xf1\xeew\xbfW}\xf1\x8b_\x14D4
\xbai\xd3\ubffcs\u05ee\xfb\xeb\xf5\u01b2\xfd\xfb\x0f\xc0\x187\xa9W-
\xdcu\xfb\x1dX\xb5r9>x\xedZ\b0t\xdct|]\x01\x05'*r\u05f2L\v;Q\xdeC\xb1\x87\xc0\x12Cp\xfb\x8d\u0
649\x01s\x89/7\x95l\xffDv;{\x8aEo\xf9/\xa4\xbb\xa4-
qfND\x9eq\x90\xc9::fn\xebx\x8a\xfd,\xe5\xab\xd64q\xfe|P\xdcm\aRz{\x95\xee\xa4;\x91\xe2\u92b1:
\xed\x93\xfc$
.,h\xd4\xf1\f\x00\xc7\u0451\xa73\x81D\x19\xb4\x94\xe3\x90;h\xc5/\xe4\xc6\xf8\xe8\xbc\xd7\x15\xb
2\xb7\x843w:*\xb6\x1fC7\x03\xa9U+8ph\x18\xb7\xff\xe8\x11<\xf2\xf4\xcbH\x94\x82\x94\xc2v\x80Z

#Nb4\xeb#Xu\xd6y\xb8\xe2\x96?Fm\xfeyH&F\xa1\x9bu$\xad&d\xb5\x8a\xdeY\xf3Q\x9d5\x0fI_\x15I\xb5\x85\xaa\x19C\x0fKH):\xad\xc2\u0476\xe5\xccL\xc9\v\x88\xb9-\xec%\x97\x97\x00#\x01\xd5\x1fX\x99\u3a36A\xc9\xcc\u0747\"]\xc06.1\xcf\u028b?\xb5\xdda\u0735=`\u07cf\\\x10\u00aa@uf\x00\u0457\x97\x17!\x05D\x108K\\\x06\x9c\x101SQ\v\x91N5\x1c>\x96\x0f7\x04\x11\u008aD2\x03`N`Z\f\x95\u0118\xb3h5\x96\xae\xb9\x04?{\xf2N\x04a\x04\xc3\fA\x84\xa8R\tv\xee\u0705\u077b\xf7\\\xf7\u063aG\xae\xbe\xfa\xaaw<v\xff\xfd\xf7\a\x00\x92\x93\x86\xcdR\xf6\xf8\xc1\x0f\xee\xc0W\xbf\xfa?\xf1\xb5\xaf\xfd?\x87\xff\xe6o\xfe\xaa\xbee\u02d6\xf7\xee\u06f7O\x1a\x9bL\xb7&\xc6\xf0\xfa\xe6\xadX\xb6b%N[~\x8a\x1bt\xe4C\x1e\x86\r\u007fV$2NQ\bC\x12\x02\x8c\x904\xa4\xbbx\x05\x8a\"#\xe6\xe9a\xe5\xd3\x19\u06b5s<hj\xba\x06:\x8d>\x8b\xa51\xeb\xbb\t\x8e\xdf\xd5\xd9\xc1\x94\xe1\x91\xe9\xff\x05\xca\xe7N\xfe\x93\x89<\u04e5\xb6\xf8\xb4no\xfehL\xb5\xa8\u02f3}\xc0\xc4x\x8e1\xe5\xf5\xe2\xf8#\x03\x13\xa5\xd1Lre\xa7\xf6?<c\xac|~\xde\xe9r\xc9m\uad54\xa1\x92fEZGB;\x9b\x89\xa2\bo\xee\u0707o\xddy?\x1ey\xe6e$\x89r\xbe)\xec<~4\xe2f\x1d+W\x9d\x86w\xfd\xca\x1fa\xf6\xeak!\x83>\xf4\xf4\xccF\xdf\xecE\x981\xef\x14\u0318\xb3\b\xb2\u0683\x96L\xa0k1\xfa\xfb$\xfa#im]SC\xac\xec\xb2)\x02\\\xcc\xe4\xe5^\xe6\uf66d\xd9K\xa6\x1a%\xf8\x0ev\fA@\x18\n\xe8\x90\x10K\xb7P\xb9\xc0d\U000b72e5K:\xa1d\x85ikh\xdaH.\x1d@Z\xfagP\xceZqL\x96\xa0G\xa26\x10\"\x9a\x1dX\xde}\xda\x12e\x17\xb1\xcf\x1a\xa3\x9csN\u0536fPA\xab\"\x84\x1d\x02+\x02X1\x90$\xe8\x9b9\a\x9c\xb4\xb0\xeb\x8dg\xa1\xb5r|s\xfbbI\xa2LT\tjc\xa3\xa3;\x9e}\xf6\xb9G\xef\xb9\xe7\xde\x13\x1dO\xfc\xef\xee1\xa5U\xe4\x82\x05\x8b\xf9\v_\xf8m\x00\xc0\xb5\xd7^\xf7\x8d\xab\xaf\xbe\xfa\xcf\xce?\xff<\xf4\xf6\xd4 \x82+]\xfb\xb6m\xc1\u05fe\xf6u\xac\u007fu3\xaa\xbd}\x99:\x94\x19\x99BP\xb1\x84\xce\xf0\x03\xcbK\x97\x04\x04\x94\x1761\x9d\x81_\xc9\xd73\x93\xfb6\x1de\xc1\\n\x92B\xceSa\xf3T\x9c\xcar\xbb\xae\xb1\x8de\xd1\u0669\x96'\xe8LZ\xc81\xb9\u043a\x13\xbb\xc41\x16T.\xed\xda\xd2\x1c\xa9\xf4\xfc\xa5\xf9\x9d\xa6C.\xc2\xc7\xc5\\a\xb0\xed\u0213\x9cKn\x8c\u0166\xd3b\x9e\x1ac\x15\xfe>\xceY<\u0145Ix\xb517\x81\xb3X\xb1\xd5\x15T\xa2\n~\xb6i\x1bn\xfd\u07bdx\xf2\xf9\r0\xc6r\x95\xad(\u0206p\xb7\x1a\x13X\xb1r9\xae\xff\xe5/a\xe6\xcak\xd0\x18m\xc04&\xa0\x9a\x13\x88\x1bc\x88\xeb\xa3h\x8c\x0f\xa3a\xea\xe0\x19\x84\xfe\xfe\x10\xbd\x01\xb9\x05\x9a\xba\x9c\xb0\x9cuC~\xd7\xec\xfcGT\x12C\xb5\x9aH\x9a\r\xa8f\x03\xaa\u0544\x8acOIj_\xbb&\tsj\x12\xfd\x03\x01\xc4\xfc\x10jV\x80$\x12.\u06d4\u06ba\x1c.\x1d\xccPG\xf4\x1by\x83T*\xccl(\x03\x17\xdb\x1ayw\xc32\u0641g43@8+\x00\x82\xb4\xb1\xf1\x82\u02e5\x84\fB\x9b\x89\x05\x84\x14\x16/\xf7S\xc1<\xf1\x96\xbf\xf4\t\x00\x91\u022a\x04\xf5I\x84U\t21\x16/?\a\x8b\x96\x9f\x03\x157\xad%\x80\xdb\xc1\x85QH;\xb6\xef\u0091\xc3C\x1fd\x93\xac8\x190s9\x9d'\xfd\x97\xff\xf2\xbb\xb8\xed\xb6\xdb\x01\x00\x0f>\xf8\xe0\xcb\xeb_~\xe9\xc2\xc1\xc1\x03\xab\xea\xf5\x06i\xe6\xec\";28\x88m;\xf7b\xd5i+\xb1t\xd1B$Jyy\xb2\xe4\x8a5#\x00[\xb8\x85\x01I\x8c\x88L\xae\xb0\xf4\v3O\xd98w\xed\u01cfv\x19\x9e\nO\xee\xfe[\xf2\xff\v\x0f\x1b\x17(Wm\n\xbf\x1b\xa7\xeeE\x1c]\n\xbd\xefg\xcdDG\xfd\xbe\xa7*\xe59D+\xa1!]G.Jv64\xfd\xf0\x8ci\f;[\xb1\x85V\xb4\xb6]\xb86)[\xa4\xcd>\x98\xba\x9f\x8d\\L\xe8\r\xff2\x15\xb2\x84!\x89\u0219\xa9=\xf9\xc2\xcf\xf0/w\u070f\u05f7\xec\x00\t{\xedi'\xfaI\x92\x04q\xab\x893\xcf:\x03\xd7\xdf\xfc\x05TW^\x87\u047a\x85\b\xadq\x9b],\f\x01\xaa\"@\xb3\x03\u0318i\v\xb9\xec8y\x81J'\v\xa9}\x80Q\n&I\xa0\xe2\x16T\u0482j5\xa1\xe3\x16t\x12\xc3\x18eM\xbc\xd2\x04#cR\x16v\xa6\xb4\x0e\x88P\x91\x02\x95H@\xf6Hk\u0125\xd3E\x8f<7[\xcam\n\xfcXXj\u03dc+\x0e\xec\v$\x84\x0e\x88\x8f<\x81\x90\xb5\xae\x8df\x85\b\xe7\x86\b\xaa\xd2m6J\xae,!,W\xde\xcb\at\\\xdc%\r\x89r\xd6M\x10\tDB\x00\xcd\x043\xfa\xe6b\xec\xf0^\xec\xda\xf2\x02\xac\xed\x88=\x97B\

x10)\x95`\u018c\xbe\u015b7o\xfe\xfe\xbau\x8f\xef\xfa\xbb\xbf\xff:\xdd{\xef\x0fO\xeeb~\xdbm\xb7\xe3\x87?\xbc\a\xdf\xfe\xf6w\xf0\x95\xaf|\xa5\xf1\x8do|\xe3\xf9\xd1\xd1\xd1_\u077bgOPo4\x9cT\u06b27\xf6\xed\u074b\xdd{\a\xf1\xb6\xb3\xd6`\xe1\x82yH\xe2\x96\xdf+\xd9\x13\ufdb9\x12\x8c\xc0\xf5\xea\xe9\x10\x91\xdb:m\xfawx\u040aB\x1e\xbf\xb8qG\xca\x0eP\x12\x14C\xdd!$\x97\x96\xd5Q\xb4:\xdc\x0e3\t\xf6\x89)\xaa\xdc\xe6\xc3\xd1D\x88:\"\x00\u0085x\x1b\b\xb2;)ry\xaf\xe2x\xcd\xd5\x18\x88\xb5\xed\xc8\x13\xa53.y\x86\x8f\xa7|\xee.\u02d4\u007f\xdb3S\x86\u074a\\q`U\x88$`H\"\xa8\xf4@\x1b\x8d\a\x1f{\x16\u07fe\xf3\x01\xec\xde\u007f\xd0F\x02\xc2\xc2*Zk4\x1a\r\x04\x02X{\xd5\x15\xb8\xf6\x93\xbf\x05Zr\x15\x86\xc6\x00i\xf2\x80m\x12\x02,\x04tU@\f\x84\xe8\xeb\x0f\xd1\x13\tH'+\xb7]\xa6/\xcfLC\x8d\x9dB4\x8eaZM\xa8f\xd3v\u07ad\x16t\u0702Q\xca\x1ax\xb1K\xaf\xef\xf0\xff\xc9}|\xb2\x8e\x16\x80$B\x14\nD5\x89 \xb2\v\xaf-
\xea\x1e\xba\xc3\u014e\x82D\x97\xe5\x9f\U000ee768[\xa3\x90c\xe5F\x10\xa4\x04\xc2\xfe\x10r^\x84\xa07\xc8]\x1c\xcb`Ka\x8fQjM\x9c\xedR\xb2\x85\x9a\xbc\xd0\xe7\xce\xf4\x1bA\x96\"L\x02\xe0\x86F\xc0\x01X+\xec\u0776\x1e\xf5\xb1#\b\x82\xa8x\xcf\x18\x8dE\v\x17\xecz\xfe\xf9\x17\x1e\xfdy.\xe4\u04c2Y\xd2\xc7\xfb\xdf\xffA|\xff\xfbVEu\xdey\xe7\xbfv\xd3M7~\xee\xdak\u07e5\xe7\u03db\x870\x8a
\x830{\xee\xd3O=\x83\xbf\xfc\xdb\u007f\u00ae\xfd\x87\xd17\xa3/\xe3\x05\xa7%\u0278N/e\xb6h&hCP\u01a9\xc9\x18\xa5,\n\xfewV\xd0\xf3\xa8Fg@E\x9c\xdd\u0113\xc1F\xecc\xbfmE|\xda\xed3\xbb\x1b\x86\xf9h\xeb\xe7\xa4\xff\xb69\x9d\x12\x89]r\x11@!\x84\x86$\x03\xc9\xda\x0e\xab\x196\xc7~F\xdc)\x8e\x95F3Vh\xc5\x1a\xb1b\xa8\x94On\xfc\xce\xcd+\xd8\xe9U\x94\x168\xcf
\xcb\xc0\xa7(z\x16\xb1L`\x92\x88\xaa5\x8c\x8eM\xe0\xce\x1f=\x8c\xef\xfe\xe0\x01\x1c8x\x04R\bh\xa3\xa1\x12\x858NP\x1f\x1f\u01cc\x9e*>x\xe3\xc7\xf0\v\xb7\xfc\x1e\x9as\u07ceC\xc3\x1a\x82\xb5\x85\x05\xb5\x82V1\x94\x8e\xa1B\x86\x98\x15\xa2\xb7?@o(\xac\x02\x9a\x19\xac\xadD_\xb5b\u8e05\xa4\xd9@\u04a8#\x99\x98@k|\x1c\xf1\xf8\x18T}\x1c\xaaY\x87\x8a\x9b\xd0q\fVy\x17n\xb4\xe5\xb2\x1bcla\xd7\x1aFi\x98D\xd9N\xde}h\xa5\xdc1r\u05c2\x01d@\xa8\xcc\x0eP\x9b\x1f!\x9a\x1d\x02\xa1\xc8!\x17\x81\x82\x01\x96\x95\u06a7\x17\x9d\xef\xf5\xc2\x05\b\xab\xfc\xbe\xe3\fb\t\x04\x10\xf5H\x88\x81\x10\xb2W\xe6\v\xfc$X)\ta\xe3\xf7\x84\xc8\xe1\x15\xe1\xcf6\x04r\xae\xbc{\xa1t`\xe2\u4ed6o\xceh\xc5u\xcc]z\x06\xe6-
Y\x03\xad\x93\x82\xbf\x0e\x91\xc0\xe1#C\x18\x1b\x1b\xff\u022e]\xdb\x17\xd8\x19\xe0\x9d\xf4\xf3Z\xcc\xe5\xd1<\xf9\xb6\xdbn\xc7K/=/\xfe\xfe\xef\xbf\xc1\xb7\xde\xfa\x8f\xeb\xbf\xfa\xd5?\xdf\xd6l6o\u063f\u007f\xbfh4\x9a\f\"2\x8e_\xbdm\xdbv\x8c\x8eO\xe0\xfc\xf3\xcfE\u007f\u007f\x1ft\x12\x03\xccH\xed\x99S\x87\xba\xb4\x935>\u036e,\f\xe2\x18\xa0\x92\xff_Z\xf5\x927CS-
\bS\x04y\xf04\x82\xd2\t\xc7\xfe\x1cn\x1bz\x02V\xf8UC\xe2\x86\u04e6\xb0(\x1d\xaf9\x9a1\x8c8\xd1h\u0116K\xae\x8d\x1bxj\x93+e\u0459\xf1\x99\xee\xb5\xdb\xf9\xd1\xfe\xf85\xb7(qL$!\x11Vj\u0631\xef \xee\xb8\xfb~\xfct\u0773\xa87\x1a\x10B
Q\x89\xa5\x1e&\t\x92V\x8cU\u02d7\xe0\xc37\u007f\x1a\xa7]w3\xf6%K0>\x9c \x022X\x85a`\xc00!g\n\xd4z\tU\x18\xc0\xc9\xf8-
d\x92v\xda1t\xabe!\x948\xceqo\xa5l\xc16\xd6\x05\xd0?\xb9\xd46%\xf1\xfd_\n,\x1dW\xf4\x84\xb4\xc6o\xe9,\x87\xdc\n&\"\x01\x8a\x84\x9d;\xb4l\x97O\xa9\xaa\xd2?\xd7\xee\xa4R\x99\u027d\xc7\xc7\xcf\u0198~(6\x01R\x12\u008a\x00\xe6D\bf\x87\b\x04\xd96\x8d&\xbf\x963\xa7R\xe3\\\x0f\xb9\xe8\x1a\u019eL\x97\n\x9d\\\xaa\xae5@\u02c0[\f\u048c0\xea\xc5\xc8\xe1\xbd\x18\u0739\xd1y\xe2Pv\xce\xc0\x06Q\x14.\u073dg\xf7\xb7\x9ez\xf2\xa9\x03\xdf\xfd\xee\xf7~n;\xf3\xe0h\u007f\xe0\x82\v.6\xeb\xd6=*\xae\xba\xea\x1a\xf3\xcew^\xfb\xad\u035b^o\xd5j\xd5[\xef\xbc\xeb\xee\u0783\x87\x0

e\x19\x80\x85R6\xe0\xf6\xae\x1f\xfc\b$\x04\xbe\xf8\x9b\xbf\x82\xc5sf\xa0>6j\xbb)\x12`\x92\bX\xb7\t`\xc8R\x00\x8b\xd6\r\xedb\xbd\xa3\xc2~\x8f\xb7\bN\xf5\xbb\xd2 a\xff\x8by\xacW\xc1\x9e\xaf V!tr\xc6A\x1d\xf7S\xe9\x0e\xe5h\ni7\x1e~Y\ta\"\xc8\xcc{$\xdd\xd2\u04d4\u01fe\x9d>\u0775\x88\xb3\xed\xbe\x13e}s,Fn\xa9j\xda\xf0\xa4|\x98\xe2\x0e\xadS\xaa\xc8m\xef\xc00C:\xc8\xe3\u535b\xf0o\xf7>\x84\xd7\xde\u0612m\xd5ce\xdd5\x8d\xb1\xa6gW\\\xf1v\\w\xe3/\xc1\xac\xb8\f\xdbFfB\x0f7QqT70\xdb\x10b6@@\x88f\x00\x95\x9aB\xa04r\xe1g\x8aU;\xf8\xc0\x15\xaa4\x9e\xae}H\x9c\xa6!Y
&
e\x05\xb2\xa8\x9c\xf5\xf8\xfe\xc6\xd22\x89\xb5\x15\x10\xb1\x00sP\x84\xb9\U0008a360W\xa2**\x80\x00\xf4\x91\x04\x1csg\u01b7A!g3\xe3\x98\x17|_\xa835\x0e\x80\f\x04\u0090@\xb3\x02\xd0@\x88 @F)\xb6?\xc3]\x9b\x12\x9b8$
\x03\xeb\vo\x8c\xce\x17*a\xbfg\xb7[\x94\xef\xc0\xd8W\xde\x02H\f\xb4\xd2H\f\xc0\x94`\xe9\ua2f1\xe9\xc5\xfbph\xef\x16\x84Q\xd5\xd5\x0f\x01\xa5\x15\xf6\xee\u06c7\xbe\u07be\xcf\x02\xf8\xe2\xcf3\xcc\x12\x1c\xcb\x0f]u\xd55\xe6\x81\a~\"\xae\xbb\xee=f\xf5\x9a3\xbe\xff\u66dbxdx\xf4\xd6\x1f\xddw_\u07e81LD\x94$\t\xc0\x8c;\xef\xbc\x17:\x89\xf1{\xbf\xf5+8u\xf1|\x8c\x8f\x8eBi\xe3\x06E\xe4.t\x82\xc86\xd3%\xfa\xc4\x13\xd8rwS=\x1eMa/\n\x88r\xbc\x9c=\f\xc1\xff\x1c\x9e\xe5g7\x1b$\u007f\xc03\xd9`\xf4X\xf8U\x93\xcb\xf49\xf3\u02a1\xb6R\x99)t\xb9\xfb\xb1\xe2i.\xa6\u06b0\r\\N\x95\x9d\xcaX\xba\xa1\xc9\u04c1\xdam\x1b\n\xdcv\xcfG\x9e2\xbadn!L\x9e\xaa\x98\x19\bC\x89@\x06x\xf2\x85\x9f\xe1\xbbw?\x84\x9d\xbb\xf6@Jk\xbf\x1a+;TTI\x82(\fp\xf5UW\xe2\u068f\u0708#\x8b/\xc4\xd6\xe1*\u0091\x16*\xdad;G\x8b\xef\x02\x88\x80\xa0_
\xecaH\x9d\xc0$\xbeN\x92\xb3\xf3\xc7\xe9\xcfx\xe2\xa6\xe2\xf1\xe2\xdc\u007f\xa4M\xb0S\xb0,\u02c4c\x9c\x15Hr\x8b\x06\x1b\xdb\u0676\ufb6dw\x8c}\x05Y\x13\xa8\u032f\xc0\x04\x02\xeaP\f\u0772\xb1z\xd4\x01\x99g\xb9\x9d\x00\x00
\x00IDAT\u0561\xcde\x8d\v\x17\x1f\u00df\xd83d@\b\x02\x82\xe8\x91\x10s\"\u020a\x00\x99\xa2\x81\\g!\xf7--
\x9c\x96CJ\xabh\u054e+C\x9c\rk\x99\xa8p\xbc\bN0\x06\x02\fC\u0156\xc2j\x1b$\x8d\x81E+\xd1?\xb0\x04\x87\xf6lvL\xa1\xdc\x1a\xa0>\xd1\xc0\xc1\x83\a\xdf\xcd\xccU\"j\x1e8\xb0\a\v\x16,9y1\xf3\xf6\xc7u\u05fd\u01e4\xf8\u04eaUkn\xbb\xfe}\xef\xf9\xb5\xcb/\xbb\f\xfd3fP\x14\x85\x1c\x85Qv\x83\xdd}\xef\xfd\xf8\xef\u007f\xf2Wx\xfc\xb9\r\x88\xaaU\xf4\xd4*\x8e\xb5b\xe9n\x86rq\x05\xa1\x18\x12\x9c}\xdd\x13\xa6\xd1\t(\xe8e]*\x97\f\x1a\xdb?LF\u0663\u008e\xc2d\xd1\xc5\xc5!g\xa1\"S\x113o\x87\xbc\xa9=\n\xac\xa3H\xf21\x17r`\n\xf9\a\xb3g\x00V\xb2s\x98\xe45'\x9bg\xa4B\xa0fK\xa1\xd92h\xc6\x06J[\\<\vW\xce\f\br\xd6\au\x8c\u073c\u063e<E\xad\xe0:\x98\x0e\xa0\xab\x95\x10\x81\x94x\xec\x99\xf5\xf8\u07f7\xfd\b\xdbw\xee\x86p\xdd^\x92X\x98#\x89c\xf4\xf5\xf6\xe2C7\xbc\x0f\xd7\u007f\xf2\x16\x1c\\p\x016\x8dU
\xea\n\x91\xd6)\xa5\xdb\x05ZXH\x84\x04CT\ud015\x95\xce1m\xf7\u007f\xa4\xec\x13\xe73\xa23\x9e\xb8W\xac9\x0f\xd3((\xfa\xb9\x9d\x86\xea\xd9C\xf8\x17J\x1b\x90]\u0282\xca\n5\x10V\x05\u00b9!*\xf3\"D5\t!QN\xdd\x12(\x1a\x03!\xa7\xfa\x16\xba?I\b%AT\x04\u011c\x10\xb27p\xb5\xdf\xebV\n\xd7>\xb9B\xdef\x95\xcb6\x81I\x04\x01HH\xbb\xdb\xc8\xdc+\xc9\xfe\x1bT\\W\xd86}F\x19$\xb1\x816\xe9\xa2n \xa2\bs\x97\xacAT\xed\xb5\xe7\xcb-
>\x82$\x12\xa5\xb1c\u01ce\xd5/\xbf\xfc\xdc\"\x00?\x97\x85\xfc\xa81\xf3\xf6\xc7'?\xf9\t\x9cw\xdey\xf4\xe0\x83\x0f\xe1\xb6\xdb\xfem\xe3\xef\xfc\xce\x17\x16\f\r\r_\xb2\xff\xc0~\xeb\a'\x88\x8c\x1b\\\xec\u0735\a\u03fd\xf8\n\xa4\x108u\xd9R\xf4\xf7\xcf@\xa2\x8d-
\x8c\f\xebW\xec\xd6\x16Q\xd8bR\xe6'\x91]s\xe2\xc4LC\xb9C\xe5V\xd6\x05\xe7\xfc\x94n\xae#^)j+

?m\xc0\x01\xb5\xd5\xf5\x92\xd0\xf3n\x90E\x19\x0419\x0e>\x05V\x8f\x12\x96\r&g\xc5\xf0tw\b\xee
\x05\x94\xb6\xd8x3\xd1V\xd5\xe9\xe4\xf9:\xf3W!\xcf\x12*\xff\x1c(\b\x11\xbb,b\xe4\xf9k;j\xa8
\xd4j\x154[1\x1eX\xf7,n\xbb\xe7\xa7\xd8?x\x18a\x10\x80\rCi\r\xa3\r\x1a\x8d&\xe6\x0e\xcc\xc6\xc7
o\xfc\x10.\xb9\xfe\xa3\xd8Z[\x89\xad\xcd\x1a\x82\x98QmiH\xe5e4\xa4\xb9\xa0\xc4@\x00\x04\x11
l\x82|\xa1\x9b\xf6\xbc*\t\x85\xecQ\x12T`\xc1\b\x19
\x8c\xaa\b\xabU\x04a\x04\x19F\x96\x1dS\x18(S\xc7\xf5\x05\xe1\x162a}\x83D
!\x82\x10\xe4\xaa3y'\xc5\x17\xf8\b\x06D@\xa0\xaa-
\x98B\xc3v\xd1\xde.\x87\xfc\v\x92\xca\aA$l7\x1e\x04\x02\"\"\xc89!\x829\x91\vW\xf6\xad\b\xbc\x06
\xbc\xf4J\xf1\x96\xea,\xe5\u02e4\xc9!E]\x95?\xe8N\xcf\x03\x80V]C7\x8c\xfd\xdb\\\ve\xed\x174v\x
bd\xf1,\x9a\x8dQ\u0220\x92\x89\x88\xb4V`c\xc4\xf2\xe5\u02df\xbd\xfb\xee{6\xfc\xc7\x00\xb4\xe4\
xf1\x9d\xef|\x17\xd7_\xff^\xdc\u007f\xff\x03\x00\x80{\xee\xb9\xf7\u01fbw\xef8\xad\xd1h\x9es\xe8\
xd0!J/n\xe3\x9c\xe6FG\xc7\xf0\xcc\v\x1bp`\xf0\b\x16\u031f\x83\xc5\v\xe6B\b\x91\x19)\x19\x88\xc
cMQ\x90_j\u06f8\xcd\x1e'\xfdXg\x95\u9ad9l\xfe\u07aet\xa4\x82\xea\u0447W|\x9b0\xf2\x1c@\xba{
\xb1{e\x8a&\xe7\xb4w\xe88\xa6\xb9\xbb8\xdeY\xc0t\x06\xa8\xd3}h\xc3H\xb4\xb1~\xf2\x0e\x1b\xb
7B\xa04\x00\xa4\u0762\x81\vXr\x0e\xd9z\xdb|\xcf\x11\u0424\xb0K\xba \b\x81
\bP\xadD\u0635\xef
\xfe\xed\x87\x0f\xe3\xc7\x0f?\x8d\x91\xd1q\b\x17.\x92R\x0f\x9b\xcd&\x96.\x9c\x87O\xdc\xfcK8\xe
3\x1d\x1f\xc0\xab\xb4\x14{\x92*\xa4aT\x9a\x1aA\x92\x17\x12\xf2\xa8\x91l)<\b\x88!B\x80d\xfe\x9e
\n\xe7/\x1d\xbe\x91\xb0\"\x19ae\xecA\x18BV\xaa0l\x82\x03\x9b7`\xcb\x13\xf7c\ubccf\xe2\xe0\xd6
7
\x84D\xcf\xcc\x01\x17f\xe1\x0f3\xbdE\x9e,\xbf\u074aml!\x97A\xe0\f\xe1<\x0e\xbe\xa7\xbb\xf1wz\x
90\x04\xaaX\x93.\xa1\x1c^\x9e\x0e5\x99\x8ab8\xa2B\x87.$\x10\x84\xc2\x05-
\x13\xc4\xec\x10r~\x04\n(\x17\x06Q\xb7\x96\xa1\xfb\xfe\x90<og6\xa6\xc3\x054\v\xd8\xd66\xf8Ci\
x83FK\xc344(\xb1\xb3\t\xc1v\b*E\x88\xa8Z\xc3\xf6W\x9f\xc0\xf8\xd0\x01\x04\x0e\x19
AH\x94B%\nq\xca\u04a5\xad\xc7\u05ad\xbb\x03\x00\x1e\u007fb\x1dn\xbd\xf5\x1f\xff\x033\xf7\x1f
\xbf\xf7{_\xcaC\x8c\x8943\xffZ\x18F\xa8\xd5j\x9f|\xfe\xf9\xe7\xc1c\xe3\xa0\n\xa1\u0572\xc1\f\xcdf
\x13w\xdc\xf3\x13l\u067a\x1d\x9f\xfa\xd8\xfb\xf0\x9e_\xb8\x123f\xf4\xa11Q\xb7\xb44\x10b\x04\b\
u0666JJX\xf9rf\xa2\xe2\x17\xf1\x94\x9b~\f\x86S\xbe;E\xea\f\x93
@\x02\x89\xc0\xc9e\u049e\xdc\u03ef\xcc\x13\xae\x18y\x881w\xc0\x12T\xc2\x12\xc9\xddh\xca;\\\
xc6$\x98\xfa4\x8b/\x9f\xa0\x01\xef\xb1\x14q\xc3\f\xe5(\x86\x896\u06403\x93\xe2\x1b/&\x82\xbcA\
x1fS\xc6\x15G\xc1\b\n\xb9!\x13\xc1\xfb~>\xf8\f\x02\x89(\n\xa1\x94\xc2\xd3/\xbd\x8a{\x1f|\x12\xaf
m\xden}\u0209\xa0\x13e\xe9\x87J#n\xb5\xb0r\xd9R\xfc\xe2\xa7n\u0092\xb5\xef\xc1\xcb\xf1\\\x1cI
B\x04\xc2 L\fd\xe2\x16\x88\x94\x1dbl\x04\x0fl\x17\x15(\x81\x80%\x02- \xc2
\xf7\x01/`\u0294\xc1G\xa9\xc531P\x1f>\x82]\xaf<\x8b\x9d/=\x81\xc17_\xc5\xf8\xa1\xfdPI\x82JO\x
1f\xf6\x9ey>.\xb8\xe1\xd3Xx\xfa\xb9\x801\xd9\f\x83\xdb|\xe13\x1e\xb6\x83%\v\x1eZ\x93\f\x1e\x01
\x17\x06\x1d\x10\u012c\xd0r\xf1\x87\x12P\xc3\xc2]>|\xc5\xf6D\xe6l
\xb2\x85<\f\\JP_\x00\x9a\x17\x82B\xe1<T\xba\x14r/0\x8cPf\u06563[D\x10\xc0h\r2\xc6QI\x1d\x1f]\
b04\f\xacG}#6\xd0-
\x8dP1\xda\x13\xbb\xb5\xd2\b\xc3>\xcc]\xb4\x1a\x83\xbb^\x831\x1a\"\xc8\u0565q\x92`\u04d6\xcd
\x17\xa6\xbf\xfb\xca+\xae\xfa\x8f\x01h\xe9\x85\xe2,5\x17/^@D\xd4`\xe6\xdf\xe8\xef\x9f1X\xadV\x
bf\xb8a\xc3\xcfp\xe0\xc0~03\xc7IB\xa9\xb8\u854do`\xcf\xde\x03x\xed\xcd\x1d\xf8\u055b?\x86\x9
5K\x17\"N\x12\xa8$\xb1\x1d\x1d\vH6\b\xa1\x10\xb0C\xa9\xa9\x8bS\"\x15g/G\xa7\xfcd'V\"w@4\x8

2\f\x15/w;\x14\x05.H\xb1|\xa6\xaf\xa3=\x01t*d)\xf3\xf6\xe7\xe3(\x9e'\xa2\x90\xd3q\x16\xf0\x94q\x94\xe6s*\u05c5\xa7TC\u035c\x8bC\u041ejFY\xdef\x16\x15\xe4\vm\xbcX7\xf6\xd8\x1a\xe4\xb6\xfd\xd5J\x15a \xb1{\xffA<\xf8\xf8\xf3x\xf8\u0257p\xe8\xc8\b\x84
\x186P\x89\u016e\x95\xd2\xd0*\xc6Y\xa7\xaf\xc0G~\xf1&\xcc=\xf7j\xbc\u051c\x8b!\x15
$\x03\xd2\f\xd1\xd46\x04\x81$d\xe4\xe0\x8bD\x83\xb5\x82\x86\x82Jb\x98\xa6\x027\x19aS@$\x01\xa8\"
D`c\xcd\x18PI\x8c\xe6\xd8\b&\x0e\x0f\xa21z\x18I\xa3\x0e\x93\xc4\x18\x1d\u070b}o\xac\xc7\xd0\xeemh\x8d\x8dX\xf5fda\x16\xad\x12\xec}\xf5E\xcc^\xba\x02\xb3\x97\x9d\x86\xb0R\xb5\xf8\xbb\u020b\xac\xf0\xf8\xd7$\x85\x85K\xb2,Z\xce\x06\x9fS^\x1bi\x82\xd1\xcc\xc0\xed<\x14$3\xa0\xbdE6\ud4055\xf4\xaa\x046\\\x19\x82\x80\x1eKCDE\xe4\ucb2eq'\xf0\x98F\xdc\xe5\x9at\u0443BB\x04\x12l\xa4m\f\xd9X\xdfsw\\\x8d\x01Z\x9a\xd1L\f\xa2\x84!\r{a\x16\xee}$\n$\x02,8\xe5llz\xe9'\xd0*\x81\x91\x01\x04\bA\x10bl\xbc\x8e]\xbbv/x\xe1\xc5\u7bbb\xe8\xc2K\x1e\xf8\x0f6\xcb\x14\x05\xfd\u99df\xc4\u06b5\x97\x83\x88\xc6\x00\xfc\xce\x13O\xac{\xfa\xae\xbb~\xf0\u01cf=\xf6\u061am\u06f6\xd3\xe8\u0628VJ\t\xa54\x81\x19\x87\x87\x86\xf1O\xffz\a\xf6\xef?\x88\xcf\xfc\xd2\r8e\xf1\x02\xf4\xd5\"
\xf4\xf7\xf7C1\xa1^o
1\x12\xa1\x90\b\x9chEd\xae\x8a\xdc\xe9}\\t\x01\x9a\xd4\u00ef\xddN\x97\x00DPmE\x9a\xba\xbe\xceT_K\x87\xa2>\xf7\xc5\xef,i\x1a\x05\xf2h\xa0\x93N\xaa!\x1f\xd7b0mHE[HE)\xceB$\xb4\xb6\x18\xa7f.\xaa\\\x19\x1dA\x1d\xa9\xbc\x9d<\u0715\xbc\x00\x8aB2\x8d\u00e1\x830@%\n161\x81\x176l\xc2=\x0f<\x81\u05f6\xec\x80R\x1a\x950\xb0\xbf\u07e4\u0656\x84\x9e\x9e\x1a\xce9\xfbB\xbc\xf7\xa3\x1fEe\xf5Z\xac\xaf\xcf\xc0\xa8\x91\b\xa5\x01\x91@\x90\x00\x91\b!\x03\x03\xd5j\xa01v\x18qs\x1c\xcd\xd1\u00d8\x18:\x80\x89\xe1\xfd\x98\x18\u068f\xb81\x0e\u05b1\x85*\"\x89\xa0VEP\xadA\b\x89\xa4\xd5Bsl\b\x8d\x91!\xc4\xf51$-
+\b2*\xb1>+I\vB\x06\x90a\xe82?M\x16\x1bW\x1f>\x8c\xc3;\u07c4j5\x11\xd5zm\x97\x9aA6\xb6C\xb5B\x1b\x99\xbfF\x10\xa2`\x14=\xcd\u055b\x00\x88P\x00\xb3C@\x10\xe4\b\x01\xb1\x014\xa0\x13\x83\xc4\x15_)\b\x95\x80\x10Fv\xf1@E\x80g\x05@U\x80\x8d\v\x9dqP(qq\xb8\xef\x13\x00\xda=8\x19E\xd6V\x16\xb0.\x03\x18\xa9AZ[k\xect7G\x84D\x03\xf5\x96\x86I\f(6Y\x8a\x14\f@\x9a\x9dy\x9a\x86\x90U,Xr&z\xfa\x060:\xb4/\xfbE6\xec9\xc1\xf0\xf0\xe8\xec\x87\x1e|\xe8b\x00\xffQ\u0327z\xac]{9\xa7\x90\v\x00\\q\xc5U\xdfc\xe6\x9f\xfc\u065f\xfd\xd9o>\xf1\xc4\xe3_\u06bcy\u04ec]\xbb\xf7\xa2\xd5j\xb1J\x12JO\xe4\x8f\x1fZ\x87\x177\xbc\x8a\xd3V,\xc3\xcae\x8b\xf1\xb6\xd3O\xc3\x05\u7781\xd3O[\x0e&\x89\xb1\x89:\x9aZC\x92\x84dv>\xe9\xd6\xf6\x899\x87.\x98\x8f\x06\x05\xee\xe2nX\xb2%\u0325\xc9\xdc\x01\x9f\xf8\xd1\x1aT\xd2\xf5\x17Cp\xbb\x8f\x0e\xdb\xe1\x96\xf6\xa7uc\xa2\xd0q\x94d\x9a\xe6B\u046d\xf0+m\xe9a6\x9b\xb3]\x8a\xdf)\x1fL-
\x00\f\xa7v\r\x94\xb3=\x88\x8b\x8e\x87(H\x80\x00\x00Q\x18
\x8aB\x8c\xd5\x1bxu\xe3\x0e<\xb8\xee9<\xf1\xfc\x06\f\x8f\x8c#\f$\x82@Bi\xe3\x15\x16F$\x03,;u\x19.\xba\xe6]\x98\x98\xb3\x06/\xef\xd7ha\x02\xa1\xb4\xe97Fk\xa8\x91\t4\x8e\x8c\xa21<\x88\xe1}\xdbpx\xcf\xeb\x988\xb2\x17qs\fI\xab\x01\x1d7]\xb6g\xee\xfc\x92\x05-\xa4CP\xa3-
\x17\u06a7\xf39\xae9\tB\xa5\xa77\x17\x1f\xa52}\xad\x11\xd7\xc7\x11\xf5\xf4a\xf1\x19\xe7\xa3\xda7#\x17\xbdd3I\xab\xd4$\x87\x93\xa7CSHYlb&=Y\xd4I1\n\bbf`\v\uaa06hh\u02002\xbb\xe1\x00\xb0\t?\x86\x81\x8a\x80\x98\x19\x80{\x84\x9bW\xf8\xd7iz\x8fP\x9b\xf6\"]\x9c\xfd\xa6\xc8\u3d34[\x05\basS\xc9*\u008d\a\u02754#\xd1@\xa8I\xf1\xce\xe6\x02i4`F\xff4\xe8\x9d1\a\x03\vWal\xf8@\xfeur\r\x03\t\xec\u077b\xef\x1d\x00\xfe\x14\x00\x9e|\xf2q\\~\xf9\x95?7\xc5\xfc-
\x13M\xfe\xe3?~\x93~\xe5W~5\xbb\x95\x1fz\xe8\xfew\xdf}\xf7\xbd\u007f\xf2\xf4\xd3O\xbf}\u02d6

7111\x818l`t\xb9\x99\xd9Y\xa7\xaf\xc4\a\xdes\r.\xbd\xe8<\xbcm\xcdJT\xab\x15\x8c7Z\x88\x95-
\xe6\x15$n\\\n\x87zs\xdb\x14\xfd\xc4<\xba\xb1(\xb8\xf0yqPk\xda\xfa\xe2\xb43\x17\x05^\xcc4@}/v
\xfd\xe8f\x01\u04c7X\x8eZh\u0140r\xcc\x10\xad<{Z\xe7p\x98\xfa\xa9\x982\x17\xc9\xff\x8f\xbd7\r\
xb2\xe4:\xaf\xc4\xcew\xef\xcd\u0337\xd5^\xd5\xfb\n\x10\r\x80X\tb\xe3Np\x81(B\x94DK3\x1cihY\x
d684\xe3P\xcc(\x183\xa6\x15\xe1\t\u02e3\x88\xb14\x94\x1c\x1e\x85\xc2\x16l{,\xc7\f\x15Z\fQ\x96\
xc4m8\xdcA\x90\x00\xb1/\xc4\xd6@7\xba\xbb\x1a\xbd\xd6\xfa\xf6\x97y\xef\xe7\x1f\xf7f\xe6\xcd|\x
f9\xaa\x01nSptF\x14\xd0\xdd\xf5\xeaU\xe6{\xf9\xce\xfd\xee\xf9\xcew\x0e{\xbe;\xec\xf1\xe1\xa9\xe
e\x98\xf3+M\xfd\u0749\b\x81\xab\xc4/\xaem\xe2\xd9\x17O\xe2\u0467\x9f\u01c3\x8f=\x83\xd3g.\x
d8\u079ep\u0563\xbfj\xc0r\xf4J\n\xcc\xce\u0361\xb9\xb4\a\xa3\xda,X\xd5\x10F\x01\x94\x8a\xec4\
xe7h\x04\xbd\xb9\x81\xde\xca9\xf4\xdb\x17\x10\x0f{\xcez\x16\xde|@\xb1kM$\n\x00U\x90(ybs.v\xf
3\n/z\x12[/\x96\x1dW\xbe\x11o\xfa\xb9\xff\x12ox\xeb\xddPa\xe4\x85<\xbb\xa6\xa7\x14\x10A\x00\
x15FPadm3\x84\u020a\x18P\x15oWV\xb5\xe7Z}\u00dc\u075f\xd6\xdf\xc4\x00\xe7G\xa0N\x02c\x
80\x81{OC\x82\x95 \x06\x04\x9a\r@\x8b\x01\x8c\xb4
\xcdd\xfd`R\n_PQ\x1b\x9fs\xe5<\xa6h\xc9*\xf3\n\x9fw\xa3\x13\xc4\xfd>\xe2Q\xec\xb2U\r\xfa\xfd\x
18\xeb\xed>\x92~\x8c\xa8\x1bC%\x06R\x00\x01\x03\xd0\fh\xeb\xf1\x04\x86\xf5e!\xc2c\xdf\xfaS<\
xf2\xf5\xff\xdb6\xa0\x9dm\x80N\x124\xeb\x11n\xb9\xf9\xfa\xa7?\xff\xf9\u03fd\x87\x88.\\\xae\xcc_\
xe5q\xe3\x8d7\xf2'>\xf1\t|\xfc\xe3\x1f\a\x00\xbc\xf7\xbdw\u007f\x99\x99\x1f\xfaW\xff\xea\xb7\xff\xf
1\x13O<\xf9?>\xf3\xcc3\xf5W^9\x83~\xbf\x8f$I\xc6~\xfe\x99\xe7\x8f\xe1\x99\xe7\x8f\xe1\xe0\xbe\
xddx\xe7[\u078c\xf7\xbe\xebN\xdcz\xf3u\x98m\xd6m3U\v\xd8E\u06a0\x18%Q\xb2\xea|\x95\x15ie
u\xea\u0758E\xba\xa3z\u0528l\xc0UUR\xb3\x9f\xc4\x0e\x14$e\x93\x90\xb6*\xddh\xf2>\xa2z\xff\xf
0\x83\xae\xe2)\x96\x19cUI\xb1\xb1\x94J\x9e\xfaS\xde\x15\xe5\xd5P\nt\xe9LA*\xee\x14p\xd9S\xe
9\xf5\trS\x90\xec\xf9WK\x04JA\t\x81\xf3+kx\xfa\xe8q|\xef\xf1g\xf1\u0533\xc7pqu\xc3\u01baI\x91\x
9f\xa5\x1f>\xec\xe8\x1d\"@\x1b\x83\v\x17.\xe0\uc673\x16tDn\x86\x95\xf3Y\xe9X\xbc\xafe\xa7|]\x
a0\xf1\xb6t\xae\xc4\xf0\x16\x0f6\u067f1\xe7u({\x1aq\xa3\rt<\x04\x18\xb8\xf6\xae\x0f\u1dbf\xff\x8f\
xb1\xf3\xaa\xeb0\xec\xf5`t\xe2\x940\xf6<\x85R\x10A\b\x15\xd6
\xc3\u0426\xe9\x00\x85\xa1\x98R:u\xc5\xddIYsZ\xfb*\"\x01
\xb6\u065aF3X\x10\x928o\x88j\x00\xd2]\x9b\x88\rL[\x83[\xd2:\xade}\u07f42v\xaa\x17b\bL\x8a*\xc
9\xefI\xa2\xea\x1b\xdd\xf7/O\xafch\x18\x9a\tR3\x843\x0fK\xef=2\u0206\xaa\x88\x81$\x89Q\xab\x
b7\xb0\xb0\xebJ\xc80\x82\x8ec\xa4\x01DRH\xf4z=\xac\xae\xaf\xed\xfa\xd2\x17?\u007f+\x80/^\x0
6\xf3Wy\xbc\xf9\u0377e\u007f>}\xfa$\xfd\xcb\u007f\xf9\xdbDDk\x00~\xef\xcb_\xfe\u04a3_\xfa\xd2
\u007f\xfc\u0503\x0f>x\xe8\xc9'\x9fB\xbb\xdd\x1e\xc3\u04f4\xa9zb\xf9\f\xfe\xc3\xff\xf39|\xe9k\xf7\
xe3\x9doy3>\xfc3\xef\u016d7_\x8fz\xad\x05\x13\x0f\xc0\xc9(\xe3\x18\xdd~\xad0\xfcR\x05\xdaU\x
80\xe6\x9b\u0231\xa7\xbc\xc8e^T\x01\x9d\x19\xc1\xe3Et\xf9|~9\x93\x87\xe1\xe7\u07fd:@\xe5B\x
d5\u03d7`\u0479:\x93h\xe2b\xb6\x15o\x0f\xd8\x0fN\xa2\xd9\xd2)n\xd8'\xd5P\xe7\xa6W>WJn\xea\
x913\x8b\x00O\xba\xe0\xc9\xf9(\v\xa7'\xa4@.\x10H\t\xa9$\x8ca,\x9f9\x8f\x87\x9fx\x0eO<\xfb\x12
^<\xb1\x8c\x15\xd7\xdc\x14\x04h\u0599\xd5+\t\xdfm\xdbQ7^\xfe\x9e\x90.\xd75G\xe1\u0716\x96\x
b7&\x99R|\xce\x1f\xe2\xe0*\x05{\x81\x82\x97}6z\xee\x82[\u04a1\xa3\xf4\xcb\x18\x83Zk\no|\xef\x8
7q\xe7/\xff\x06\x9a\x8b;1\xe8lzT\x83\x93\x1f\x06!d\x18Y\x10\x972\xcb\xed\u012b\\\xa2\x99\xd2\x
e0t\xbbK\xd2\xc6s\x97\x14\f3b`-\x01w\x12g;\xec\xde\v\a\xa4\x9a\x80>\x01\x92\x19r`
\xe2\xd8V\xc2S\n\xc25@\xb3\r\x87\u06cd\x18\xa6\xcc{\x89\xb2\xa6lu\xd1A\x15\xea\x17\xc0\x99\
u007f\xb9k\x1d&\x8c\xa1\xf3\x99W\xda)r\xd2\xdfk\xac\xda\u021ff5l\xef\xcf\xd6\xccN\xb4fvb\xfd\xe
2\xc9\xcc7\x9e\x04!\x1eiN\x12\xbd\xf8\xf0\xa3\x8f\xdc\n\xe0\x8b\x9d\xce:\xb5Z\xb3|\x19\xcc_\u00

f1w\xef\x01\x86\xcd\x14\x95w\xdf\xfd\x01}\xf7\xdd\x1f\xf823\xdf\xfe;\xbf\xf3?\xfd\xdbVk\xea\x97\x1f{\xec1\x9c?\u007f\xbeP\xeaf\x8d\x11a\xb7v\x17V\xd6\xf0W\x9f\xfb\n\x1ex\xe4)\xfc\xd4{\u078a{\xde\xf7v\\s\xd5!\xcc4\x1aP\xd0H\xe2\x18q\x1c\xdb\u02a2\x94\x95I|\u9294\\\xf52r\x9e\u06a9\x97v\x9ar\"\x05Y\xf5@aK\xe9\u03f9S\u05bc\xf3\x15\x18\x94i\xa3\xd9Q-
T\x92t\xf0\xe4\x91\xd4Kp\xecE.\x1e\x85\xdd\u026b\xa9\xc6'%-
1[\xad\xb8\u05b9\xccP\xeb\xbc\n\u03f7\xec\xe3K\xce\x18h\xbb\xd2^P\x1e\x1dB\u0796\x842><@\x10(\f\x86#\x1c;\xb6\x8c\xef<\xf24\x9e\xf8\xfeQ\x9cx\xe5,\xba\xdd~\x0e\xe2I\x82D\ub8bf\xb9\x11\x05\xc4\xf5\xdb\xd0i)\xca^\xf6l\x1a\x98\x9cJ\xfc\xf2\n3\x05B*\xc4\"g3\xaa~@\xab%\xc5\xf3\x85_k\x18\x13\xc3$\xb1\xa5J`w\x17\xb6i\x19@\xd6\xea\b\xeaM,]\xf1F\\\xf3\x9e\x0f\xe1\xd0-
oG\u051c\u00b0\xb3\xe9\x02\xc4\xedh\xbb\x90\xca\xd1*5\b\xa5<\x83\xa2\x92-
\x00\x97\x03\x1cx\x8c\x0e3\u0657U\x17%\x06\x88\x8d]\x98i3\x81\xe8h\v\xd0\xc6\x01%\x91}I\x8c\xb5j\xd1\x00\xe2\u0600b\xb7\xe0\x8e\fxh\x80\xf9\x00A
\x10\xc0R,\x82r\xd2)\xad\xfe%\xa5!\x1d\xbc\xe5}\xc7^\xd4\"\x93\xe5\xcdA\x02\x9a\r\x06\t\u00f0\x05r\x99x\xbd\x14\xc3
\xe3\xf9\xb6\xe7\xac\x17\xb4\x8eQk\xcc`a\xe7a\xac_8Q\xbc\xaf\rF\xbdn7z\xf9\xf8\x89\xeb\x00\xa0\u055a-
\xf4\xf8.\x83\xf9k8\xee\xbe\xfb\x03\xfa\xbb\u07fd\x9f\xde\U00096df1\xe3\xac\xfe\xe1\xe7?\xffw\xf7\xdd{\xef_\xfd\x0f_\xff\xfa7\xf6-//S\x92$\f
\x91R\x06\xc6\xe4\u06fe\xac\xca?s\x0e\xff\u05df\xfe5\xbe\xfe\xed\xef\xe1\xe6\x1b\xae\xc5\xfb\xde\xf5\x16\xdct\xf5\x01,M\xd5\xd1j\xd4
\xa5@\x1c'\x8e\x8fw\xa6Q\x82\xbcq\xf0q\x0e8\x05\xf2\xcc\xfc\u0254\x12\xe0\xe19\x1c\n\xca\x00*\xad\xd8\r\x17\xedk\xa9d\xfa\x9fZ\xfd\xda\xf1\xe2\x1c\xf0S\xb9b>\x04UE\x10\xf1\x96\x11p\x97\x8c\x87\u02ea\u03ca\xb0\vWys\xf6\xe1\xe7\f\xc8\x13\xe34\xe2\x06^\xcc\x172\x10-
\x0fx\x8c)\xd2\xc0\xce\x10*\xd7\x031\xfb\xe7\x04h\xb6UW\x14\x86H\x92\x04/\x1e}\x19_\xfd\u03a3x\xe4\xc9\xe7q\xe6\xdc\x05\f\x86C+\x8bc\x83\xd10\x81N\x9b\x9b$2\x8d\xb2T\xcaM\x0fR\x9e)Ib\x82\x1c\xa88\x11@.\x8f\x16\xe5k\x01g\v2\x15\xc6b\u0269P,`\xb3\xb1v\xb5iU-
\x82\b\xd1\xcc4\xc2\xe6\f\xa2\x99\x05D\xd3s\x88\x9aS\x98\x9a[\xc0\xec\xee\xfdX8t5\xe6\xf6]\x81\xe6\xec\x02\x88\xadU\xae\b#\x90T\x80\xfb\x12nQ\x13\x82<>~<\x9d\xb5j\xbf\xc9^\v\x82a\x81{\xa4\x19\xa3\x84m\x13\x91\xad\u03f9\xeaj\x04\x1d\x9d\xb9)J\r\x906\x85\xf4:2VVn\x90F\xf4\xb8\x02G\xc7\xd0\xcc\xe8N\xdbs\f\b\b%\xd91\u007fApsD\u064e\xc06\x8bKq\x90^hz\x96\xb1\x9b\xaa_\xa4\x02d\x82x\x94`\xe8V#\x19[\xa5\xa6I\xcd\u0334\xef\x83\u3337\u049d\xa4\x8eQo\xce`i\xf7\x11\x1c\xfb\xfe7=\xd5\x14\x10\x06R\x9d>\xfd\n\xd6\xd77\xaeg\xe6\xeb\x88\xe8\xfb\x0f<\xf0\x1d\xe1.\xf32\x98\xbf\xd6\xe3-
oy\x1b\xff\xe1\x1f\xfe\xaf\xf8\xcd\xdf\xfc\x18\x01\xe0{\xee\xf9\xd0'\x99\xf9O\u007f\xeb\xb7>\xfe\xb1\xaf}\xed\xeb\x1f=}\xfa\xf4U\xe7\xcf_\b\x1c\xa8kr\xdd&\xe6bK\xfe\xf8\x89\xd38y\xfa,\xbe\xf5\x9dGp\xf8\xc0\x1e\xbc\xf9\xfa#\xb8\xe5\xfa+q\xc5\xfe\xdd\u06390\x87\xd9\xe9&\xea\xad\x10Z\x1b\f\x86\xb1\xe3\xe5\xd3j\xae\x18Ll\x00\x9b\x06\x9f\u4cba\x8cB\xf0@/\x05\xea\\vE\x13Io\xf6V\x8e\xbc\x92/\xf2\xeb\xc2q\xc4R\x90\xb5\x0f\x15y|\x96\xc8\x02}\xe9\x92\x16\xb9\x97\xecX\x96\xcf0\xab\xd6,\xff\xad\xbd\xf1\xf3\xb4\x92\xcb\r\xb0\u0704\x1e\xa8\x04\x1b\x18k&\xa4\u065a)\xd02\x8a\xf9\x9c\xe9K\x92\xaa9\xeaQ\b\"\x81S\xaf\x9c\u00f7\xbe\xf7\x04\xee\xfb\u07938\xb9|\x16q\x1c\x83\x88!\x05C'\xf6\xbd3F\x17\x16\xf6\x99f\x1d,\x02\f\x13\r\b\xe5Y\xa9::\x82\xca\xd1\xd4\xee}O+u\xe1\xde\xff\xf2\xf6\x82<\xbb]r`\xa15\x92d\b\x9d\xc460\xb8>\x8d\xc6\xd4\x02\x82Z\v\xaa\u0442l4\x10\x

cd.\xa2\xb1\xb8\x1bS{\x0ecz\u07d5\xa8\xcf\uf08cj\x10B\"\f\x15\x1aQ\x80Z\x18@\bB\x92\xc4\xf6\xfd\xaf\xd5\x1d\x85\"\xa1\x85Mv\"a\x13\x84\xc0\xe3\xadl\x86\xbf\xeb\xc9{/\xc6\x03q\xe3\x16\xc9X\x03\x83\u0120\x9f\u062a<\x9d\xef\t{\x1aa\xcf\x05\x002 \x13\x06\xa5#\xfe\xeey9\r
l\xe1Rx\x8e\x8e\x9a!:\x1a,\bqCbD@?a(\x01\x04\x82\x10)\x81@\x00J\x10\xa4p\xe7lJ\n-
\"+/vU\xbd\u007f\xee
;\xa0\xdfO\xec\xdfe\u0090\t\x17\xed\x1e\xb8L\xdb\xe4K\xb4\xd1\tT\xd0\xc2\xcc\xc2>\x04a\x03:\x199+\x04{K\x8c\x86C\xac\xac\xac\xcc}\xfd\xeb_Y\x04\x80\x87\x1ez\xe82g\xfe\xc3\x1c\xbf\xf9\x9b\x1f\u00d7\xbe\xf4y\xfe\xdd\xdf\xfd=\xfa\xe67\xefc\xa7K\xff\x1df\xfe\xc3\u007f\xf6\xcf\xfe\xe9o>\xf2\xc8#\xbf\xb0\xb6\xb6~\xe3\u0253'U\xb7\u06ddH\xb5\xe9Dceu\r+\xabkx\xfc\xe9\xe7\xf1\xe7\x8d:\xf6\xeeZ\xc4\rG\x0e\xe1\xea+\xf6\xe3\xf0\x81=8\xb0g\t\xbb\x97\x160?;\x05%\x05\x86q\x828N\x9c\t\x12\x9c\u0740\vFH\xb5\xd1i\x13\r\xc5(\xb2t\x84\\x\xe8J\x9c\xfbqT\xc1:gU\x15gF]\xec*>\xe1l\v\x04\x11F\xe9vU\xb8x=7\x8aL\xd9v\xd5U\xf0\x8en \xaf;[H.\xe2\x94#-
\xce\xcc[_\v\xf6>8\x9c\aA\xf86\xa7\xde8\xbdU.\xe4\\0{WS\xf8l2\x8fy\x8f\xb3\xdf\xe0\xf4\x99\xa8tj3\b\xb0\xb2\xbe\x89\a\x1e\xfd>\xbe\xfc\xad\x87p\xf4\xe5eG\x93i\b\xc1\xd0:A<\xb2\x15\xb0\xf1\x16\x84\x85\xd9)\\q\xcb[\xb1\xef\x8e\xf7\xe3\xc2\xc55<\xf1\xf9?E\xfb\x95\x97\x01\x00A\x10\x81\\t\x98]L\xc8k\xc62\xd8P\u039b39\x90\x19\xb7\x84\xe4Tj\xa8ch\x1d#\b\xeah\xcd\xed\xc1\xd4\xfc>\xcc\xee\xb9\x023{\xaeDki?jsK\x90K\x8b\xc0L\x13*j@H\xe5M(\xe7J\x8e\x04@;\x06\x86\x9c\xa0\x11)\xd4ju[\xc9;\x15\x0e\x91\xb0\xc0\n\x86\x12\xbe\xa3\xa2\x0fU\xa5\x05\x94\xfc\xc57\u007fOc\xc3\xe8'\x8c~l{\x1d\xbeY\x9d\x1a\x18\x84=\x03\xa1\x1d\x90\u01f6\xa9\x98V\xe8>\xedg_\"\xceR\x83X\xe44\"%\x8c\xa8\xab!\b\x186\xa4\x93\xa9\x02\xb1f\xf4\x13\r%\b\xa1\x00\x02%\xec\xff\x85\xf3\x96\xe1\\\x0f\xa6\x992/\x11.\x15\tCc\x1b\x9fd\x00\x15\xc36>\xc9\x1a\xf2\x11\x93]xD\xfax\xe1\x8a\x1d\xce\xc4\x05\xcc\x1a\xf5\xd6\x02\x9a3;\xb0~\xf1$dn@I\xa3\xd1\bI\x92\xecy\xf2\u0267\xae\x05\xf0\xcd3g\xce\xd07\xbe\xf15\xbc\xfb\xdd\xef\xb9\f\xe6?\xe8\xf1\x81\x0f\xdc\x03\x00\xfc\x89O\xfc\x1b|\xfc\xe3\xff}\xfa\xa1\xdbp\xa0\xfe\xe9O|\xe2\xdf\xfc\xbd\xaf|\xe5+w\\\xbcx\xf1M\xab\xabko\xbcp\xe1\x02\xf5{\xbd\x89M\xbb$I\xb0\xb1\xd9\xc6\xe6f\a\u03fft\x12\x81\x92\x98j5qx\xff.\\}\xc5~\xbc\xe1\xe0^\\}\xc5~\x1c\u063b\x03{v,`\xba\xd5p\xcd=\r\x8c\x12\b\x9d'\xe80\xa7\xe3\u031c\r\xb5\x14\xa8\x89\u031b\x9c\xf3\x06\x90\xc3\b]\xb007\xd9$\x93\xd7\x12t\x80n}2\xa4I}hr\xd0\xcf\x03+<\xbf\rg\xf3*D\x1aO\xe6\xd1\xf6\x94\x876hf\x1b\x19\xc6~\x02\x0fg\x95O1\u0601=f\xd8\a\xe2\xb1\xfek\xc1\x1f\xc5\xdf\u044cW\u0754\ro1\xb8\xe0\x86)\x85D-
\f\xd1\x1f\x8d\xf0\xc0c\xcf\xe0\xab\xf7?\x82\x87\x9ex\x16\xdd^?\xb3\x9a5\xdaN[\xdaj<\xff\r;\x17fp\xed\xdb\u07cf\xddw\xfe4\xe6n\xbe\vb\xf7aLo\xf4\x10\x1c\xba\x01\u01fe\xfcg\xb8\xf8\xec#\xe8\x9e_\x06b\xbf\xa9M\xb9#\x1f\u0195*Y3\x8d\xf2\xf7\x8b@PA\x1d\xad\x99]\x98Z\u060b\x99\x85\x83\x98\xdbq\x18s;\xaf\xc0\xcc\xce+P\u07f9\a\\\x0fa\x8c\x86\x96\x8c$\x12\xd0!Y\xd9\x1e\xb4\xf5\xe0\xce$M\x02p`\xcdB`(m\xfa-
Q\x93n'\xe6^#\xe9^?\u9a38K\x1a;0C\x1b{\xbf\xc1Q\x1a\xfd\x84\xd1\x1d\x19\x8c\xb4?=k\xd5Bj`,\x00k\v\x922f\x88\x84=\x0f\x1a\x14\xf5/\x8e\xff
\xb2\x14\x87\xf6\xfb\x94\f\x88\x04\bz\x06F\x10t]\x14*\xad\xc40b\x03\x90\x03\xf6H\x12j\xca\xfe_\n\xbf7\xe1\x9e\xd7S\xe4\x8c\f\xa3\x97\u061d\x84t\u7656\xf0\x04\x80\xd2\x05\xb9\x9c\x15\xea\xed\xb6\xb5NPk\xcc`jv\x17\xd6\xce\x1fw\xbb\r\x06\x98\x89\x99\x93n\xb7\xa3\x9e|\xf2\xa9E\x00\xe8\xf7\a\xe2\xdd\xef~\x8f\xbe\\\x99\xff\b\x8e\x14\xc8\xff\xe0\x0f>\x81\u007f\xf1/>\x8e\xcf~\xf6^\x10\xd1Y\x00\u007f\x04\xe0\x8f\x1e}\xf4\xa1k\xfe\u077f\xfb\x93[VVV\xff\xeb\x97^|\xf1}'N\x9e@\xbb\xdd\xc1p8,|\xd8\xfdJ\xd8\x18\x838f\xacmlbmc\x13\x8f<\xf5<\xa20\u011e\x9d\v8\xb8g'\x0e\xee\u06c

9\xc3\xfbw\xe3\r\a\xf7\xe2\xe0\xbe]X\x9c\x9bA-\x8a
\x94\x1d\x821\u0330\x8e\xa6\u059c\xc9h.\x86\x9dL\u0a33\x01!\xbf\xd8c\xf6\x92\x93\xbc!$\xa4`\v
\x8c\x91\xb6Y\x95h\xe0u\x87\\Ui\n\xbc\u007f\xce\xf2\xf8q_9H\x1b\xe61\xe0\xf2\u01f0\xfd\x19W\x
eb#\x9d7\xa4*\x98\x94L\xf7c|\xf6\x99\xf2\x9d\a\xf9i\xf0d\xd5\rRZ\x99\xa11\x8cg_:\x81\xaf?\xf08\x
ee{\xf0\t\x9c\xbd\xb8\x9a-
L\x16\xc4c\x8cFCho\xf6`\xef\ue778\xfe\xdd\x1f\xc0\x9e\xb7\xfc4\xa6\xae{\aF3{\xd0\x19&\xc0z\a2
Px\xc3\xdd\x1f\xc1\xee7\xbd\x13\x1b'\x9e\xc3\xe6\xf21l\x9ez\x01\x1b'\x8f\xa2{~\x19\xfd\x95\xb3\x
88\xbbm\xb0N\xf2IJ\x12\u05a4*\x8clcR\x86\b\x82\x1a\xa2\xda4\x9a\xd3;05\xbb\v3K\x870=\xbf\x0
f\xcd\xd9\x1dh\xcd\xecFT\x9f\x86\xd61\xb4\xb1v\x13\tb\x98@\u0606]\fH&P
`j\xcaV\xd9BZw*\x91Rz\xf9\xffcfl\x0e5FZ\xa0\x19\nD\n\x10\\}Gqe\x13\xdcI\a\r\xb2]d\xac\x81^l\u04
0dm\x8f\xa3\xb0^\t@\xc4\x06AGC\xc6v7*c\x03\x11{\xd5xy\xf5N\xdf>c\xf9n\x86\x05t\xf6\x82V\xc0
\x96\x9e\x89\xba\x1a#\x02\u26a8\xf4\xeb\x1fiF\xac\x19\x83\x84\x10J\xa0\x11\b\u02f1\v.\xb65\x98
\x900\xd0\x191\x86\xc6\u07a3rd\xad\x16\xb2\u05cfE6\x8b\x90G\x8cZ\xc2\\x\xeb\xb3\xd11\x1aS3\
x98\x99\u07d5\xd1o\xe9g3PJ\x9cy\xe5,\xae\xbbnxC\xeao\xfe\xd0C\x0f\x8a\xdbn\xbb\xe3u\u03db
o\xcb6\xeeW\xbf\xfae\xfa\u02ff\xbcW|\xeaS\x9f\xd6y!\u008dO~\xf2\u007f\xffo\x9fy\xe6\xd9\xdf~\xf
8\u11e7\x8e\x1d;\x86N\xbb\x8d$\xd1\xd6\xf3e\x82)\xbe\x10\x02J\xda-
\xad1\x96Z\x10\x044\x1bu\xecX\x98\xc3\u03a59\xec\\\x9a\xc7\u03a5%,-
\xcec\xc7\xc2<v\xeeX\xc4\xc2\xc2\x1cj\xb5\xc8\x03\x1b\x03\xad\x1dw\xab=\xadqa\x94\x99K\x1bl\
xca8\xc8LaNEK]\xf2*g\xce'M\n\xc3(\xfeg\x88*\x02E)\x9b\xd2\xf0\x831\x8a\xfa\xef\x8a\x10\xf6\xac
\xfa\xcf\x15{4a\x81*\xac\a\xde\xf3PQO\xef\xfdQ\b;\xec#\xa5\xc4(Np\xec\xe4\x19|\xeb\xc1'\xf0\xd0
\x13\xcf\xe1\xc4\xe9s\x88\xe3\xc4z\xa8\x18\x8dD'HF#\x8cF\xc3\xec5\\X\x98\xc7\xcd\x1f\xfc\xfb8\
xfc\x9e_D\xe3\x8a\x1b1\xaa/\xa07J\xc0\xa3A\xeec\xeev#2\xacAE\x11d\x00\x8c:C\xf4\xd7/b\xd4Y\
u01e8\xbb\t\xdd\xefB\x0f\x870I\xec\x02#L>\x16o$\x1a}B\xa4\x15\x94\x88\x10\x85M\x84\xb5&T\x
d8t\u05a91L2\xb2\x8dNA`!\x00%\xc1\x81\x80\x8e\x14L(\x01!,\xff+\x04\x92\x86B\u04b0N\x86\u93
89\xb9\x1a\x96\x99m\xe3\xb0\x11\x10\xeaJ
\x90\xf9\xa0\x94\xf4$\x82\xe5\x06\xa7\xa3\xa0\xed\xbd\xec\xaa\xf1\xde\xc8`\xa8y\xec\xfe`\x02\x8
4\x01\x82\xb6F\xd0\xd3\x10\x86!b\x86\x1cq\xd6\x14\xcf\\\x0f1i8\x8d`\x84\x15\f\xb1\xeb5\xd8F\xa
5\x00\xa4\x95\x12&\x11a\u0612\xd05Q\x88\xa1\xe3\n\x95\xad\x10@\xa4\b\r%\x10\xcaT\x82lw\xb
b\xbd\x98\xd1M,\u0146\xde\x00\xc1\xc5.\xe4\xc0\xa6\x96q\x92\x80\xfa\t\x10\x8f\xc0.\x93\x95\xbc
`\x12\x99\ue01d\u02a81=\x83'\xbe\xf3\x17x\xe0\U000df10a\x1a
A0\xf1\x10\xa3a\xd7H)\xc4\xdd\xef\u007f\uf4df\xfe\xf4\xa7>\xb2\xb4\xb4\xeb\xb9O\u007f\xfa\x93\
xf2\xd7\u007f\xfd\x9f\xbc\xee\xabs\xb5\x1dO\xea\xbd\uff5b\x01\xe8\xe7\x9e\xfb>}\xf5\xab_\x13\xb
f\xf1\x1b\xff\x94\x88\xa8\a\xe0\u007f9w\xee\x95{\xbf\xf0\x85/\xfe\xd2\x03\x0f<\xf8\x8bO>\xf1\xe4-
O>\xf5\x14\rG\xa3L\x97^>\x8c1\x18\xb9h0!\x05\xa4\x14\x90B`0\x1c\xe1\xf8\xa93x\xe9\xe4i\b!Q\x
af7\xd0l6\xd0l4\xd0j6\xb1\xb40\x8bC\a\xf7\u16ab\x0e\xe3\x8a};13\u0570\xe3\xe4A\bU\x93\xaeq\x
e3\x05,\xa4\xd5\n\x91\xa7\x84\xb0\x1a\xe71\xb5\x01\x8a\x86\xec\xecM\u0325\xdbg\xe3\xbaF\xb
9\xe6\xdd\xe4A\xbf\xde'%\xeb\xe6\xbb'2\x15\nG\x1aSux\xe9=\xe4\x19&\xf9\xfc{\x19\xd0K\x95\xb
a`\x1fp\xf2\u6c12\x12\xb5(\x80\x10\x02\xbd\xfe\x00\u03fft\n_\u007f\xe01<\xfa\xf4Q,\x9f\xb9\x88\x
d1h\x04\xa5\xec$G\x1c\xc7\xd0\xfbeH\x9c\x00\x00
\x00IDATI\x8c8\xb1\xd4\n\x00\x04a\x88\xeb\xde\xf7a\\\xff\v\xbf\x8e\xa9\xabo\x83\x0e[X\x1b\f\xc0\
xed6\xa4\x80\xf3\xfe\xf6w\x04\xf6C:\x1c\xf6\xc1`\b\xa1Ps\xcdH!E\xee!oRE\x86\x81a\x031\x18!\u0
60c\x11n&\x90#\r\xc4\x1a&\x89\xc1&A2\xec95\x8etS\x98\n\x10\x04\x0e%X);\xd0\x04\x01M\x02Z\

u0694\x1db@\x0e\b\xa4\x80\xa4\x96\x0e\xeeVF!g\u02a8\xd806\x87VO\xdd\f\x05j\n\x19\u0152\xca\xfa\xd2j7\x9f\x9a\xb5\x80\x15\x1bF7\xb6\xb4J\xb9\x1a\xf7\x95\xb3AW#\x18\u0608<J\x182\xe6<]\b[[Ld\x16\xb8\u0300\xd3\x0f0\xa5\xd7f@F\x80%C\xb9\na(\x00\x1d\x89\xb1J\u045fk\xd2\f\xf4b\xc60\u046eB\xa7l\xaax\xe4\x94UBH\x90\xb6\xde\xeb\xf6\x17\xb2\xdd%\xe4\x19w\u07adL\xdeN\xd1\xf5\xaf\x8c\xa52\xebS\v\b\xeb-
h\x1dCP`\xfb\x14$\u0120\u07c71|\xe3\u007f\xfa\xf2\u007f:\x02\xe0\xb9N\xa7#3\xd1\xcee0\xff\xf1\x1c\xd7\\s\x1d\xa7/\xf2g>\xf3\xef\xe9\xa3\x1f\xfd\x15\u07b9s\xcf\t\x00\xbf\xc7\xcc\u007f\xfc\xb1\x8f}\xec\u007f{\xea\xe9\xa7\xff\xe1V\xcf!\x84\xa3\x05\xd8\xc0$\x1a\x80\xcen\x80T9\xc2\xcc\x18\f\x87\x18\x8eb\xac\xacm\x82aG\xc4\x1f|\xec\xfb\x98\x9aja~f\n{v-
\xe2\xf0\xfe=8\xb0g\av,\xcea\xbaUG=\x8a2O\x10)\\\xb5m\xf2\x84y\xadmXp\xa2\r\xe28A\u007f\x14\xa3?J\x9c\x0e\xb7\xa8\xb2\b\x02\x850P\b\x02\x05\xa5\x94\x9dnt\x93\x8aRX\x90\xac\x87\x01\x820\xb0\xbf+\xf5zv\xc6V\x89\xfb\xbd\x99\tQ!\xb9)\x9b\xe2(*<\u01a2\xb9\xa8z\x14\xc97\xbeB.+3\xae\x19,%!P\n\x81R\x88\x93\x04\xe7V\xd6pr\xf9\x1c\xee\xfb\u0793\xf8\xde\x13\xcf\xe1\xec\x85\x15\x8c\\%N`\xf4\a#\x9b0o\\\u2f63Uv\\\xf3&\xdc\U000abfc5\x83\xef\xfcyP=D\xbf\xaf\xa1\xbb]\xab\xec\x91b|w\xe0\xb7\\\u04e1
\x00&\x19\xc1$\xa3\x12(\xe5CJrd\xa06c\xa8n\x02\x8c4Lb\xab<;\xb4\x13B\xaa\xa8\xc0\xa5\xdb\x12\x97\xc0\x81\xb0\xf4\x89\x93\xee\x89!
\xb4q\x13\xac\x96X\x16\x9b\xdaR\v\rY\b#\x99$+e\xd8\x11\xfa\x91\u05b6b\r\xac\"D\x11\xf9Im\x19\u0367\r0L\x18\xdd\xd8*U\xc0\x13\xbcY\x18\b\xfa\x06AO\xdbsM\x81\xdcp\x85\xe72\x8f-
\xe0e\xf5Sj~\x96%~ymK\xe9\xaaa\x100$\x82\ti\uceb3\"\xc0\xbb\x8e\xbe\x19\xb7#H\U000e90c4@i\xff!\xfd\xbd\xec\x87\xc5\x18\x17\xe8\x01/!>\xed\x93\x10t\x12\xa31\xb5\x84\xe6\xec\x0e\xac\x9d?\x01\xa9\xac\xc1\x99\r\xf3`>w\xfe<=\xf6\xf8\x13\xfb\x00\xe0\xaf\xff\xfao\xfe\u007f18\xa4^/'\xfa\u044f\xfe\n\xdf\u007f\xff}\xf8\x9b\xbf\xf9\x1b\xfa\xc4'\xfe\x80\x01t\x1ez\xe8{\xbb\xfa\xfd\xfe\u011f\x99\x9e\x9e\x02k\x8dv\xb7Wb&\xb8\xf0\u007f\xfbB\x10\xa4\x12\x90\xceG\x99@H\x92\x04\xab\xab\xebX]]\u01f1\x13\xa7\xf1\xe0\xa3\xdfG\x18(\xd4k\x11\xa6\x9a\r\xcc\u037405\xd5\xc4T\xa3\x8e(\n-
\u007f9\x8am\xda{\xac\xd1\x1b\f\xb0\xd9\xeeb0\x8c\x11'1\x86\xa3\x04q\xa2\xb3\xada\xda\xe0\xb4\x15\xad\x82\n$jQ\x88F\xbd\x8eZ-
\xb4\xff\xa6\x14j\xb5\x10\xcdz\rK\v\xb3X\x98\x9b\xc6T\xa3\x810\f\x10\x86\x01\xea\xb5\b\xf5Z\r\xf5z\x1dQ\xad\x06%\t\xacu\x9a\xae\xe2\x00\x1e\x05c\xbd\xdcn
G|\xe6\x92\x05\t{4\x8e\x10P\xca\xeehD\xf6\xe5*\"\xad1\x18\x8epqm\x13\x17.\xae\xe1\xf9c\xa7\xf0\u0413\xcf\xe1\u0157\x97qqm#\xa3S\u0229\x8f\xb4\v\xf05i\x90/\x80\xe6\xd2^\\\xf5\xc1\x8f\xe2\xda\xff\xe2\x9f`\xe6\xe0!$CF\xbc\xd9\x05\x1c%\xe6\xef\x0e\xaajH\x1f\x93\xb8\u0510\xcd\xf6/\xcc\x0e\xc8\x19\xe1\xa6F\xd03\x10\x9a
`\xabo\b?\xb0\xb0b$K\xd8\xecI\x03\xdb\x03a0(a(M0\x92a\xdc\x00\x81\x18\x01\xa1\xd1
\x06\xe2\xba\x04\xcb\xf1V\xc8\x18\xa0{\x15\xebH3\x1a\x01\xd0\f,\xf5B\xb9t\x05\xb1\x01:#\x83^l\x90TU\xe3\xfe\a{`\x10t\xb5\x95\x1d\xb2mv\nS\xe2\xcch\xf29\x15-
\x12\xca\n\x1dW\x1c\v7|\xa5\x19$\x009d\x84\xd0\x18MI\xe8\x80&)XK\xef\u0378\x92V&\f\xa9\tl\\\xe4\x9csG\xf4\xcf'\x9d\xfcMUVy7\u01e9\xc6t\x82fs\x1e\x8d\xd6<V\xcf\x1esiM\x96^S2\xa0S'O\xa2\xd3\xe9\\\xcf\u0312\x88\xe2\xcb`\xfe\x13>\xdef\r\xe5\x19\x00~\xeb\xb7>\xfe\xae\v\x17.\xbcWO0\xea\xaa\xd5j\xf8\xd5_\xfdU\\\xf7\xc6k\xf1\xf0\xf7\x1e\xc0K\xcf?\x8bg\x9f?\x8as+\xeb0\\\xad\x86a\xa7~\x10d\a^\x04\x89L\x1eH\x00\xe28\xc6h4B\xa7\xdb\u00c5\x95\xb5,\xce\xcb\xdfG\x8e\xd1\x14\x9eJ\x80\xbd\u01a7?\xafY\u04a2\xb9\x9b]\xe4zv\xb2\u0a64\x95\xf4EQ\x88Z\x14\xa2^\xafaz\xaa\

x85\xb9\xb9\x19,\xcc\xce`nv\x1a\v\xf3\xb3X\x98\x9d\xc6\xecT\x1d\xf5(D\xb3Qszn\x02\xb3\x86N,
U\x834\xa6\x8cr\x85G\xea\x13nw\x03\x02\x81\xb2\xe9\xef\x83\xe1\b\xdd^\x1f\xbd\xfe\x00\xfdA\x
8ca\x1c\xa3\xd3\xed\xa3\xd3\xeda\xb3\xd3\xc3\xca\xea\x06\x96\u03dc\xc7K'^\xc1\xb9\x8b\xab\x1
8\x8eF\x96\x0er\u05db\xc4\xec\x00<\xc9\x00\x1c\x00\xa6\x16\xf6\xe2\xd0-
?\x85\xab>\xf8+X\xbc\xfd\x1d0\x01a\xd0\xeeY?\xefR)KeN7\x1d<)\xb9\x8b\xf9\xfd\x85\x02R\t\xeb
1\x12\xb65T\xdf5\x005;\x90Hw&[E\x9c\xb8II7\x1e\u039e4U\xb8\x0e!K\xcb/\x8b\x84\x10\xb65(f\xc
4M\x01\x1d\x8a\u0262\x94\x12\xb8i\x03l\x8e\fb\xc3h\x05\xc2\xd2\x10\xb0Z\xf1nl\xa7\"\xb94\xa4V
Fb94\b;\xda59\xd9\xc6\xe0\x19.\xc8E\xfd\xac\xd9\t\xb0\x8a\x02\xda\x13\x17\xe5K\x94N\x8d\xda\
xc5M\x18\x00\x9a\xa1\x86\xaeBoI\x98\x90^\xd58N\xf6\x16\xa6\xebfl\x157\xec\xe5\xa3\xda\xe7\x
19\xf7\xd5e\xce\xfbE\u067bG\u05b4\xab\u079cC\xb35o)!JS\x93\x14HJ\xb4\xdb\x1dlll\xdc\u011c\x
ec\x06\xb0|\x19\xcc\u007f\xc2\xc7\xfd\xf7\u07d7\x02:\x9e~\xfa\xfbw\x9e8qb\xac\xdaN\x8f\xa5\xa5
E\xfc\xf2/\xff\x03\xdcq\xc7[\xf1K\xbf\xf4\x0f\xb0|\xfc(\x8e>\xf90\x8e>\xf74\x9e;z\x1cO=\u007f\f\xc7
N\x9d\xc5z\xbb\x9bU\xcb:\u06da\v@*[\x89\xa6C<eg\xbe\xd4Od\xcc^\x94\u01abBx\n\x13_\x93]\xd
0g\xa7#\xe6\xb6\xda0.F=\xdd\xd2j\r\xc4q\x82\xc1p\x04\xd1\xed\xe5\x8bM\x16~k\xa9\xa1f\xb3\x8
1\xb99\v\xe8\xadF\rs\xb3S\x98\x9b\x99\xc6T\xb3\x8e\x1d\x8b\xb3\u0635\xb4\x88\xa5\xf9\x19DQ\
u0359`q\x81\xe3g6\xe8\xf5\a8\xb7\xbe\x86S\xa7\xcf\xe1\u0339\x15\xac\xaeobec\x13\xedv\x0f\xd
d\xfe\x00\xfd\xc1\b\xedn\x0f\xbd~\x1f\u00e1\u0749h\xadsN\u05f8?\x1bc\x17\x10\xb7KH\x8f\xf9=
Gp\xe4\xb6\x0f\xe1\xf0\x9b>\x80]\xd7\xdc\x01j\xd4\u047b\xd8A\xd2\x008\x14\x95\x01\xa8\\\n\xf7\
xf0m\x1a\x88\xb6n\xef3Y\x05G\xd8\xd6\b\x06\xc6R\r&\xa5\x1c\xb82\a\xb6
F\xa5\xbc\xca$O\xd2d\x03\x14\\\xc3\u05cd-
\xb0\x15]\x804\xdb\xe1\x1c\u0348\x9b@R\x13\u0145g\x12\xa8\xb9\xe7\x1b$\x96:S\"w\xaa\xd4\x
06y\x90\xf5\x84'\x101#\xec\x1a\x88\xc4%\x1d9\x90%FiBw\xb2\x1d\x1d\x97UR\xec\xe3\xa8'ka\xef
\xe5q[;b\v\xe8L\xc0\x88$\x8c\x12\x93\xb8\xb1jZ\xd4
W\u06a4U\xb8\xc9\x04]\xee\xf3\xe6\xfc[L\xeeL\x99\xf6\x15\x84\xa3MM\xa2Qk\xb4\xd0h\u0343\x8
4t\x16\xb8\x80\x94\nB(\x8cF#\xc4q|\xeb\xbd\xf7\xfe\xd5.\x00\xcb\x0f<p?\xee\xbc\xf3m\x97\xc1\xfc
'u\xfc\xee\xef\xfe^\n\xdc\xc1;\xdf\xf9\x8e_6f\xf2M\xb2w\xef>\xec\u0673\a\x80A\xbd\xd9\u01357\u0
74ekoz\x13\xd0>\x83\xd5S\xc7\xf0\u02a9\x97qj\xf9\fN\x9e>\x8b\x93\xa7\xcf\xe2\x85\x13\xa7q|\xf
9\x1cN\x9f\xbb\x88\xf5v\x0f\x83\xe1\bZ\x97\xbb\"\"K\x11\u03ebv\u1188\x8a9\x9fy%>>*N\xbe\u01
69\xd7\xc8\u02dd\xf8\xbcf.\xe5\xfc6g\x9ab\x064\xd94\x18!A\x8c\x8c\xf6\xb0\x15\xc7&\x8eq>d\xa
4\x94\xa5\x86\xe6fZX\x9a\x9f\xc3\xd2\xc2\x1c\xe6f\xa71;\xddD\xa3Q\x87\x92\x12Zklv:XYY\xc7\x
ea\xfa\x06V\xd77q\xe6\xfc\n\xd66\xdav\x98G\x97\x06\x83\x1c\xf0\xa7\x15x^\x89\xbb\u01b0\xb6C
7\xa9|4\xa8\xb5\xb0c\xffux\xc3-\x1f\u0121\x1b\u07c3\x9d\x87o\x86\x10\x11F\xbdM\xe8n\x0fA
@#\x81\xe1L\x80$\x12\x05Kc\xc03(D\x01G&\x03c>(\t\x91\xc0\x02y\xca\x1fk\x86(\f\u028c\xc1wE\
xa9_\xfd;\n\x14\x90\xb6*r\x1d\xe4\u0563\x1c\x1a\xfb\xbb\x12\x89\xa4!ad\x81\xa6\x1e\xe3\xd3\xfd
_\x17;\xbdv\n\x98[Y\x880Yz\"\xecj\u0221\x9b\x99H\xc3\x1bR\u0547\xd74\xa7\n\u0687\x8a\xec\x9
a#\xaa\U000f26483O\x98\xbc\x9bO)]\x9e\xff\xb0a\xa8\x81\xfd\xe4\x8cZ\x04\xad&\xa6\u064d\xbd\
xb4\"a+I4\xb0\rh$\xb6\xd9\xee/=\\\xbc\x0fQ\xb8'\xf2\xc9U\"Bsf\x11a\u0530>\xf1R\x82\xa4\x82T\n
F\x8fp\xfe\xfc\xf9\xf0\xa1\x87\x1e\x9e\x02\x80\u007f\xfd\xaf\xff\xe7\u02d5\xf9O\xf2\xf8\xdc\xe7>\x
0f\x00\xf8\xecg\xef\xbd\xee\u0739s\xd7UY\xe7\xa6\u01d5W^\x81\xfd\xfb\x0f!I\x86H\x92\x04\xf1h\
x04&\x01\x11.b\xf6\xcaY\xcc\x1f\xbe\x16\xd7\x0f7\x80\xf6\x1az\xed\rln\xaec\xb3\xdd\u0145\x8b\x
abx\xf1\xc4+x\xe2\xf9\xe3x\xf4\xfbGq\xf4\xc4+X\xddhc0\x8cm\xcf\xdc\x15\x98\xaf\xad\xf5M\x90*\
xb4\xcd\u0634\xbc\xa4t@Hd\xa1\x05

\u06f0e\xa7\x90I?\x18\xbe\xb9\xb6o}k\xaba\x032n\x98HHk\n\xa6\x14\x94G/23z\xfd>:\x9d.N\x9c:\x03\x80\x10\x84\n\xb50D\x10(\xeb:\xa8\r\x06\xc3!z\xfd\x01\x92D[e\x88\xa3w|\x90N\x17\x14\xceF\xfeM\xd1C$=m\x93@\x06\x11\xa6\xa6\x96\xb0t\xf0z\\u\xf3O\xe3\xc0\xd5o\xc3\xf4\x8eC\x10*D\xdc\xeb\xc1\xe8v\xb6\x18R\xcc\x10Z\x83\fa0\xa7,\xa0W,\x86\x85`\x03.J\xec\xc7\x12\x9a\x04\x814#\xea$\x19\x90\x93\v\x00\xceS\x16\x8a\xae\u06d5\u068e2\x8ap\x1emG\xc6n\xe4\xf2\tIKk\x18IY$'%@\xd8\xd5\x10\t#n:\xfa\xe1U\x98\xc9\xd3%\xf8e\x1f\u0205\x01\x82\x9e\x86r;\x8fl\xf4^\x97\x16\xad\x94&\xc2d'\xd1\xfc5\xa6bk\xb420%\ud193\xa5FR`w\xe7C\f\f\x1d\x87Nf\xeb\x8b%\xb6\xcdi\xa1\xd9J \xd3\xe9X/\xac\x84\xbd\xc72\x8d\xf9D{V\x04V\xc9Vo. \xacO\xa1\xdfY\xb3\x86f$U\x88x4\xc0\u0253\xa7\xf0\xfe\xf7\xbf\xef=\x00\xbe\xfew\u007f\xf7\xf9\xcb`\xfe\x93:\u039cY\xc6\xee\xdd\xfb\x00\x00_\xf8\xc2\x17\u007f\xf6\u0739s\x98\xa4-\x97R\xe2\x96[nq\xa0b2\u054a\xbd#\x04b\xaa\x01\x14\x81\xeaS\x90\xd1\x0e4\xe6zh$\x1d\xec2#\x1cIb\xbcm\xd4G\xd2\xedb\xb3\xd3\xc1\xcb\xcbg\xf1\xcc\u0457\xf1\xe4s\xc7\xf0\xfc\xf1e\x9c\xbd\xb0\x86\x8dN\x17\x1d\xc7!\x0f\xe3\x04\xc3QR\x18t\x91B@)\t%\x05jQ\x00\x90@\xb77\xc4(\x1e\u066a^\b\b!a`\xacQ\x94Kf1\u01a0\x16E\u0635k\x17\xc2(\x82q\x89\xf2\xc3\xd1\x00\xf1(vZ\xec\x11\xe2$\xc1h\x94
\xd6\x1aqb\x15%a\x10@\x90\x80\xd6\x15\xed&_\u0152\u068b\x0e\x86\xe8\xf5\xfa.\x1d\a\x19\r\xe2\xeb\x9b\r\x19\x90.\u058by\xb2\x0eg\xbax\u03a6f5\x8c1\x88\xea\xd3\xd8s\xe8&\xec\xbd\xeav\x1c\xba\xe1=\xd8u\xc5\u03687\x16@\x10\x88G\x03\xc4\xfd\xae\xe5\x9aS\xe3a7\x80\x02M\xa0\x8e\xd5\x15\xf7\x17\x03\vz\xba\xbaI\xe61\x1f\x05\xed6\xf2\u034d\x05\xf2v\x82\xc0M>\xc2\x01\x1c\x95\x00\x8e\xb6\xe4h\xb6\xc0\x1f\x93\x03\vL>\x1cF\xda-\xb9\xa9d\xc9X^=\xe8k\u02041jJK\xbb\x88\x92\xf1&.\xf9+'\x82`\xd03P}\xe3\x1a\x9e\x94/Zi\xdf\u0420\xb0k\x9c4\x92\xe4\xb3+9\xb7QvQ+\xe6\u0125\xf4\x910\x06\xd0\xc5\xd7M&\tH3\x06\xb3\nZ\u0456\x15\xbaH\xac\xdd.\x19\x17\x00\x92\x1a\xaa\x15\xe6\xde\xdct\x03\t\x10\xe5\xc5U\xc1\x84\u051d\x93I\x124\xa6\x16\x10\u0567\xd0k\xafdW*\xa4\x02\t\x85\xcd\xcdM\x1c?\xfe\xf2\xad\xcc<OD\xab\x97\xc1\xfc't\xa4@\xce\u0335\x9f\xfb\xb9\x0f}tk\x8ae/n\xbb\xed\xcd\x13\xca\x18\x93\x99T\xb3\x90HD\x1d\x89\xac\x03\xc1\f\x90\x8c@j\x04\x19\x0e\xa1Z\t\xe6wj\xcc\x1f:\x82[n\xbb\x15<\xe8\xa1\xdf\xed\xe2\xfc\xca\x1a\xce^\\\xc1\x85\xd5\r\\\\\xdb\xc0F\xbb\x8b\xcdN\x0f\x83\xc1\b\x86\x19JI\xb4\x9aM\xabYo6\xb1w\xef^|\xf7\xb1g\xf0\xc7\u007f\xf2\xe78\u007fq\xc5\x15\x8c\xd2\xd9u\xdaiN!\xed\xec\xf4p0\xc0\xee\u077b\xf0\x8f\xfe\u046f\xe1\x9ak\xae\xc9\u0498666\xd0i\xb7\xd1\xdeX\xc7\xfa\xfa\n\xba\x1b\x1b\xe8t\xdahon\xe0\xc2\xca*\xce^X\xc5\u064bk\xd0\xed6(\b\x10\x86\xa1m^f\x82\x04\xbf\xb2v\x9bQ\xe3{p\x97'\xdc\xf3*\u06ceG\x9b\x8c\xf7\x96R@\xa9\xc0N\xda:E\n{\xde\xdd\xe9\x13\xde\xf13\xff\f\xd7\xde\xf9\v0\x89\x8dS\xd3#;\xb8\u00de%\xad\x1f\x94\x93\xc9#\x13\xechp@\xe8-\x04[\xf2\xccU \xe4\xf3\xe9AO[EG*\xe1c[\xc1\x96\xcb\xee2\xddP\xf0\xa2'\x94\x02\xbc\xfeFy\xcd\xcc\xc7\x12!\x98a `T\x9e\xbeC\x06
c\x10i\x86H$\u2980\x91Te\xbb\xf2\x9a\x00]\r\fT\xcf5<\xb3\x99z\xce\x16\x9b\xaa\xe9\x9d*\x9a\xc5_,\x8btU\xeeMO\xbe\xe9Z\xbai\x14\xf9\xe2&\xd2\xd73}\x0e\rD\x1d\r\x10a0+\v\xd7[\xb8V\xaf*O\xe9\x1b\"\x012\xf9
\x1c\x15,,+\x9a\xa1\xeewKw\x83i6\b\x1bM\x84a\u0765G\x190\v\x80\xac\xaa%N\x12\xb4\xdb\xed\xbb?\xfd\xa9?n\x02X\xfd\xf6\xb7\xbfEo\u007f\xfb;\xf92\x98\xff\x18\x8f\xbf\xfd\xdb\xff\x17?\xfb\xb3?\x0f\x00\xf8\u02ff\xfc\xf3_|\xf1\u0157\x0e\xb9@\x8b\xca\xe3\x8a+\x0e\u399bn\xb6\xb4\u0224\xe6\v3\xc0:\xbfcI\x02A\x03\x1c6\x900\x90\xc0~_\x98\x04\xb2\x16CL\xc7h\x98\x11\x0e\x1dHpH'\xf9\xa8}V*\n\xe7\xc5!\x01\x15\x022\x00d\b\x84\xf3\xe8\x87\u007f\x01\xf3'\u007f\x91=\xdc\xd2\x13:\xa

3I\xa4R\x102\x00L\x02%%\xae\xbd\xf6j\xbc\xfd\xedo\xc3\xe6\xe6f\xe6p\u021cs\n\xc9p\x80Q\xaf\x8d^{\x1d\xab\x17\xcf\xe3\xe4\xa9e<\xf5\xcc\xf3x\xf6\x85\x17q\xf2\xf4\x19\x9c=w\x01k\xab\xeb%2\u0213tIi\x1b\xba\xe4\xbb4\xe6\x8dd6V}B\x04\u0522\b;w,b\u03de=\u0635s'@\xc0\x13\x8f?\x89\u04e7\x97\x01\u0599w\xb7\xcfs\f\xba\x1bX?\u007f\u00a93b$\x83\xbeK|O\xb7\xda\xc5Qr.\xd9\x13\x90f\x84\xed\x04IH\x18N\xa9\xca\u0127*\xd4\xf3\x15\x11\xaao2j#\xf3Ow\xf6\xa9\xe4\x05*\\r\x04\x9a_\xdd\xf7}\x95H\x8aod\f\x88\x05t\xe0Q.\xec,];\x8eviI\xe8\x90\n\xd2\xcbW\v\xe8L\x80\x1a\xda\ub509\x17rlR*\xc9\x1ad\tT\x84A\x8ceUy\xe2C\xf2c\xde\xc6\xe5/\x05j+\xe36\xec\xa0Nz\r\xe9N\x80\u025eO\xd0\xd30\x02\x18MI\xbb\xc0q\xf1=\x13\xb1\xab\xcau\xdep\x15d+\xf0\x14\xc8\xd9\u07ce\x19\xd7\xc4u;\u0374\x1a'o'J`\xa8Z\x13A\xbd\x99\xdd\u06e9m\x85\x94\x01t\x1c\xe3\xc5\x17_\x14w\xdcq\xfb\x8d\x00N\xbd\x9e\x81\xfcu\x03\xe6)\x90\x03\xc0\x97\xbe\xf4\x1f\xefn\xb7;A\u0784\x1b\u007f\xfdo\xbf\xfdvLO\xcf\u0098W)\x1fe.\xf2\xa1\x19\xaaH\x18)aT-
\xff\x00\xb0\xb1_\xde\u0523eo\xdcG\x86\x84\xdbn\xdbG(\x002\xaaCJY\xa9\xbca\xb6\u0737\xfd\xbd\xc6\xf9\x93\x8c\x9c/L\xe2Gdf\x8a\x93\xb0\xdeD\xd4hazi7\xf6\\q\rn\xba\xdd\xe0\u00ff\xa8\xb1\xb2r\x11/\xbet\x1c'N\x9c\u00a9\xe5\u04f8\xb8\xb2\x82N\xa7k\a\xa2\x9c\xbcpc\xb3\x9d\xfd{\xa2mU\x9dv\xf9\xc30@\xa3\xd9\xc0\xd2\xe2\"\x0e\x1d<\x88\xab\xaf\xbe\n{\xf7\xee\u016e]\xbb\xb0o\xdf^\x1c<x\x00'N\x9c\xc4?\xff\xe7\xff\x1d\x8e\x1f\u007f\t\u02bb\xa6\xf2q\xfc\u026f\xe1\u01bb\xfe+\x84\xf5i\x18A\x99\xde:\xdb\xees\xa9\xa2\x857\xdc\x04\xfb\xe1\x8e\xda\x1aF\n$u1\x06\xe8\xe5\x14x\xdf\xdeD\x0es
\xf7\x91\xd1R\x10\xe3,Ju\x95:)\xabjkb\x86\r\xe7\xd6\b\xce\a\x05,`\x82\u0519\xd2\x01\x9ea\x04}K\xff\x8cZrL\xed\xf2j\x80\\$\x8c\xa0ke\x96\x80\x1b\xb5w\x1c=\x00\x98t\x98\xa9\x1c\u00baU*I%A\xef5\x1f\xfd\u0184\x93\xefd\x03E\"\xc3\xf5\xfc'\xdd\xf7D\xc2\bz\x06L\x84x\xca\u06d1P^\x95\u02c4\u11feP\xea\xf4\u51842\xd5JV\xa9\vgfWm_\xcf`\x04Q\x1dA\xd8\xf0z,\xf63\xa4\x82\x10I\xdc\u01c5\v\x17p\xee\u0739w\x00x\u0753\xe6\xaf\v0\xbf\xef\xbeo\xe2\x1d\xefx\x17\x98\xf9\xc0\xddw\xbf\xff\xb6\xf5\xf5uL\xda%\xce\xcf\xcf\xe1\xfd\xef\u007f\x9f\xad\n\xf5\x0f2\xa1;f\x13X\xaa\xfel\xb50\xf6\xab\xb3F\xa5.\xfc\xac\x8a\xecdd\x10\x04c\x1d\xf8L\xf2\xe84\xe0\x96\x05J\x10H\x81(\n\x11\x85\xa1\u05ec\xca\x03t\v\x897\xee\x1f\r\t,\xee\x9f\xc1\x9e\xc3G\x10*i\xf5\xe1\x83>\xfa\xfd\x81\xd5\xc7\xc71\xfa\xfd!\xda\xdd\x0e\xd6V\xd7\xd0\xed\xf6p\xe1\xe2\nVVWQ\x8bB4\x9b-
LOOaff\x06\v\xf3\xf3\u063d{\x17v\xef\xd9\r%\xc3\x02\xa0\x19\xc3h6\x1b\u0540\xe3}\xa2.,?\x83ao\x13\xb5\xd6|\xb6)
S\xad\xe4\xe0B\x022y\x1fpv[tk\xe4\xe4\x03\x05U\x045A\xb8E\xa0\xe3\x14\x1d&'L\x84\x01D\x89\x9a+\xe7\xbf\xfaCU<\x11\xecP$\xec\xbd+\xc8\xdf%\x9fr\"\xc8\u0100\x98\xa0\xc3<S-
\xa5\"\xe4\xc8 j;\u06a5\xe1@\xae\xd4s\xac\x02rb \xec\x19(w\x9d\x9c.
Nv\xc9\xce\r\x91\fg=\x99\xccLml;\xe3]\u05ebxrz\x9f\x91H\xa7`\xcb\v,9u\x960l\xad\x05\b\x18\xb5\xf2k\x15\x86\xa1F%Z({\xaf\xc9\u04d0{\u0397\xbe\xcf(;\xa5\r\n\xf9\xe0v\xd7+\x03\x84\xb5&\x84\x90\xd9k\x04\x10\x84P
\x92\x88\xe3\x04\xedv\xfb\xed\xccLD\xc4\x0f>\xf8]\xdcq\xc7[.\x83\xf9\x8f\xeb\xd8\xd8\xd8
\x00\xfc\x99\xcf\xfc\xfb\x0ft\xbb\xddk:\x9d\x0e\xa4\x94\x94s\xc0\xf9q\xfd\xf5\xd7\xe3\xb6\xdbn\u035a\x9f?\xfa\x83\xc7\xcc\xf1/u\xa4\xe1\n\x85f\x8f\x1b\xc9O\xc1<=W\x02\xa3^\x8fP\v#\x98V=W\xba\x94T\r\x13\xf3\x9f\x19\x18%\x89\x05\x91\xb0\x8e\xe9Z3\xf7A\x17\x02\xb9\x8b4\xd0\xedn\xa2\xd3\xe9bn~\x0ea\x10\x00\xf0+m\x8d8\x1ea\x10ws\xafv\"\xec\u0631\v\xd7\\s\r\xbe\xf0\x85\xad\xf3p\x87\xbd\r\xac_8\x8e\xe9\x1d\a\x1c\xf7\xa9\x8bzB\xaeh\xf9yZ?\"\xbb.\u02a1A\xe8\xc0/\xad\xd0=?\xb2\xfc\xda\xc9V\xdeaGC\r\xd8\xf1\xd3\xde\xebm\xbc\x99\xf8J\x19\xe28\xe5P\xdd\x13%\x94'\x9

7\xa8\x9c\xef\xea+/8\xe7\x93e\f\x18i\x95\x1a~\xcf@\x8c\x80P'\x19\xa0'\x91\x18\xabp\u02e7\x915
<\xd3\x06d\xdah-
{\x94\xbb\xc1\x1e\x9atY\u07a9\x17\xbc\x1a\xc6\x1a\xa5\xbe\xd8\xdc\xf9\xa5\xf89\xa0n\x95)\x18\x
ba\xa5\xef\x95qrA\xe7Q\x1e\fL\x01\xd0\xd5\xc8\xca\x11\v\xc6\x12\xee\x9c\x04\xa8@\x162\xb9\x
b6W\x99=,uA\xd2A?A\x12Q\xad\t\xa9\x82\xcc?)-
\x88H\x06l\x18\xb4\xbe\xbe~\x13\x80\x10\xc0\xf0\xde{?{\xb92\xffq\x1d.\xa3\x8f\x01\xe0\xfe\xfb\xb
fs\xd3\xc6\xc6\x06\x88(!\"U\x06r\xa5\x14~\xfe\xe7\u007f\x0e33\xf3`N\xb6\xcd5\xc4\u039f;\x05D\xf
2\x06\x91R\xfa%\xbd\x96\u010d\u29c0\x9fz\u02cc\x93\u00d3K&\x9f\re\x9d\x14\xa3\xbf8\xe7\x0e
G\xa3\xa1\xb5!0\x06\x83\xc1\x00\xfe\xe2\xe8\x9fgff\xe4\x16\x9c;\xef\xbc\x13\xd3\xd3\u04d6\u04d
ft\xcd\xc3\x1e\xd6\xce\x1e\u00fek\xdf\x01\x01\x01\u05ba$s+\x0e\xe2p\xa9$\xe6\xd4\xc3\xddyZ\x
a7M\xb4\xa4N\xb9%k\xa9\xb8\fz\x1aA?\x1fa\xf7\xb7\xfad&\x039\x95\xaa\xf3j\u04ba\xb8\xf8PE\r\x
cf\xf0\ar\xc6w-
V\xa5\xe1\x93\xea\xf9\x0f\x92\xb6<?%\fj\xb2\xb5\x02(\a\u007f\xbb\x1fUC\xcbA\v\x9b\u049c\rh\x8
a\xc4y\x94\x9bTzY:\xb1TN\x99\xb1\x89\x159nTR\xafT*\xe0K\x04\xff%z\r)0\xa7\xfe*\xd0\f\xd5\xd7
`\x01\xc4u\xe1t\xe5\x9cW\xe2\xc8\u04cf\u021f\xae\xf6\xa98\x93\xae_\x9c\x85W\xa7V\xb8%\xa7^
\x04Q\vB\x05H\x86\x03/$\x05PA@\xc3Q\x1f/\xbcp\x94\xef\xbb\xef\x1b\xd7\x02x\xfc\xf7\u007f\xff\
xf7_\xb7`.\xb6\xf3\xc9}\xef{\x0f\xe0\x8b_\xfc\xbc\xcd\x14a}\xd3\xfa\xfa\xfa\xddg\u039cA\x10X\x02\
xb2\f\xe67\xdcp=\xee\xb9\xe7\x83\x0e@\xb7\x8f\u0742U\x83\x98\x02\xb5\xa2\x94D\x18\x86\b\xc
3\x00\xd25$\x01`0\x18\xa0\xeb\xbcd\xb2\xc6\x0e\xe7\tA\xec\x15\x98\x93\xbe\x8a\xe0\ue0f2\xc8\x
16\x10!\xadSd\xbd^/\x9c\x97\x942;\x9fr\xd0-\xb9O\xc1-
\xb7\u070c\xab\xaf>\xb2\xf5\"l4:kg\xdd(5\xd2`\u0209\xe7X\xacx\x91i\rS)\xa1\x1c\x1aD\xed\x04j`\x
8a\u041a6<\x9d\x17\x89\xf0\u0324\xc8G\xe8\t\x9e\xe1\xe3\r\xc7\xd2$\u0418/@\x99\xf2'g\x04@\x
05&\xa6\xfc\v\u04a6+\x19[9\xdbp\b\u05f0t\ra\xe1\xc6\xe1\xa3M\x8d\xa8\xe3\u4525\u0758\x8c\x1
9a'\xb1\xa1\r\x8cl\xb1SCc\xadm\xc7v
<F\xd10\x95\xae\u008bB,<\xd0\xe7z\xa8\xf4vy\xefW\xf1\xdby\fK\xc1r\xc1_\xb8\u063e\x06AW#\xd
a\xd4\x10\xc3\xdc:\x80\xb4\xb7\xbb\u0206\x95\xbc\xe9k\xce\xdf\x13.u\xbe\ty\xfaRZ\xb4\xc00\xa2\
xfa4TP+\xd99\x1b\b!\x91$\x8cv\xbb=\xf5\xf8c\x8f\xbf\ue8c6\xb65\x98\xdf~\xfb\x9dx\xf9\xe5\x97\x
05\x00|\xe63\xff\xe1\xc6\xf5\xf5\xf57t:]H)E\xf9\u62e2\x10\xf7\xdc\xf3A\x1c:t\xa8\xc4Y\xfe\xe7?\x9
2\xa4\xa8COu\xe6\xf5z\r\xf5z\x03J\xc9\f8\xbb\xdd.666~\"\xe7\xf5ZS\xc9S\x8f\xef\u00c7\x0f\xe3\x
c0\x81\x03\x97||o\xf3|\xb1\xea+\x15v\xc5\x05\xa8\b<i\xb5I\x0e\x00\x89\xedV<h\xeb\xcc\xff#\x05\t\
xa1\xed\xe4\xa3\x1a\x8d\x0f\x90\xa4\xcd\xc6q\xcb\xd7-
\xb8\xf1W\xc5\x18\xa3\xb0\xa3\xca\u0399*\x00-
\xc5L\x93'\xcb\v\xcd\x10\xb1\xc9\x12\u007f2k\x01\xb6\xbc\u007f\xd0\u0548\xda\xda\xd2\x0f\xa9E
\x80\xb1\x96\xb6i?\xc0.\n\x06\"6N~\xc9\x05\xb9g\xa1\x97\xc1\xbc\u014e\x8e=\x15\x8b_N\x97\xa
b\xec\n\x13\x9c\t\xfd&\xae\xd0\x1f\xfa\n\xa6t'a\xdf7\xe3\xed\xa0r\xeb\x01\u02e1\xa7\xc1\x14\xfe\
b\xac\xb7+\xa2\x1c\u0229\xd4\xc2b\xe3f7\xea3\b\x82\x1a\x8a\x96aY\xae\x17\x12mp\xf2\xd4\u02
77\xa6?\xfb\xc2\v\xcf]\xa6Y~\xd4\xc7p\xd8C\x145\f\x00<\xf5\xd4\xf7oXYY\x013'\u032c\xca|\xf9\x9
e={p\xf7\xdd\xefG\x18\xd6\x11\u01c3mu\x1d\xfd~?\xdb)\xf8\xd4E\xb3\xd9B\x18\x86\xd0:\xc1h4\x
ca\x1e\xeb\x1a\xbc\xdb\xf20\xc6
\bj8p`\xff%\x1f;\xe8m\xc0h\x03\x91\"o%\xe7_\b\xb5\xcc=V\xbd\xe6Y\x96\xbda,\xc5@\xd0\x18\xc2
\xfa\x9d\x10[zB\x0e\xbd\x91\xf5\u0313e\\\xcd\xc1\xd5L\xc0\x04\x9c\xa3\xf1^g\x15\x1d\xe3\x11[\xf
e\xa3*\x1d\xc2M\xf1y\u0205J\xb0\xb4\xd2M\xe3\xe8\x17\vt\xc6\r\x19\x01I$\x10\xf45\xd4\xc0NU\x

92N\x17\x81*\x8e\xc8\xef\xed\xa0\xd2{f\xfcJ&\u0343\xfa=\x8e\\\xc0\x88R\x03\xba@:\xf9\x95\xbf\xe3\xd1\xd9\u04e2s*\xfc2p\n\x16O?\xc0\x1e%\x95Z\u0792\xa8x\xf2rR\x91\xd7\x1b@\xaef\x01\x03a\xad\x05\xa9B\xef\xc7)\xdd\x12A*\u0143\xc1\x90\x8e\x1f\u007f\xf9:f\x8e\x88hx\xe4\xc85\x97+\xf3\x1f\xf5q\uff5f\x15\x96\x16\u3e53'O\xbe\xf1\xf4\xe9W\x10\x04J\xa4^
~\xa5\xfb\xaew\xbd\v\xd7]w\x1d\x00\xfdcj|\xfe\xe0G\xbb\xdd\xc6`0(\x809\xc0\x98r\x8e\x87R*\xe4\xd1b\xc0p8\xda\xd6=\f\x00\u063f\xff\x00\xa2(\u06aahE<\xe8\x01\xc6\xe4\f2\x97\x84\u0528\xb2\xf0\xab\xa2[\xf2\xc7\b\u00d0C\xa7X\x19\xd9\u051c\xa0\xaf]e[\x1a\vu\xe0^V\xd0T\xa7)Up\u0095\xc9;\xe3\xe1\xd5)\x9c\xa7\xbe\x8b\\\x1e\x83\xaf\u068a\x94+\xf6t2\u0564\x8b\x92\xfd\xb3\xa5\x974\xea\xeb\x89uB4(6w+\x1a\x80\xd9\x16!\xf5\xab\xa7\xf2\xe3h\x02\x17\xce\xc5\xef\x8f\xfdL\xfe\xbdI\xe2L*?\xbf\xef\xf2\xe0\x14/Y\xe5\xads\xc33\x91\x18\bc\xb2\xc57\u02f8\xf5\xa7\xb7Q\xf9\x16\xa3\xb8\x01b\x18\xef^e6\x88\xeaS\x90A-[\xf5\x84H-
4\bA\x10Q\xb77\xc0\xa9S\xcb\xf3\xdf\xfd\xce}w\\\xa6Y~L\xc7\xc6\u01ba\x02\x80\xcf}\xeeoo%\xa2{677@D\xc27u\xb2U\xf9n|\xf8\xc3?\x8f\xf9\xf9%l\xe5o\xfe\x93>\u049b\xf0\xe2\u014b\x8ef\xf1\xd2\xe2\x19XXX\xc0\xe2\xe2\x12\xa4\x14\x85\x9b6\xbd\x86\xd7J\x83\xfc$\x8f\x03\a\xf6cvvv\xe2N\x1b\x00\xe2a\x17Z\xc7VZF^\xfd\xc6\xe3UVUm\x9c\x8e\xc9Sb\xb2*\x94\x9c\xc4P\x8e\x18\xd1f\x82h3\xc9\xd2s\xc8\x14<(\xf3m\xbb).\x1a%bd\x8c\"\x98h\x81[\xe2\xceil)\xe0\U00045a7a\x9e/^s\x89\x82\xb1\xf6\xafy*\x85\xdf/
]\xc4V\xf6\xa5\x92\\\u0352\xe4\x97E\xd5\x15\xfaV\x8c\xe4\x18\xc5R\xf4\n*\xfe..\x18;\xfbt\fy\x14P\xba\xab\xf0\xcd\xce\xc8 7\xec\xf2{&\x94\xe7\v
Wv\xe6\v\xa6\x97\x9a\x95\xf7\x96\u0606\u0478>p\u0618B\x10\u05b2\xe9\xe7l\x91\x80}nm\x80~\u007f\xb0\xe3[\u07fa\xef\xb6\xcb`\xfec:\xbe\xf2\x95\xaf\x02\x00\x9e|\xf2\xc9\xfd)
\xa6\x1e&>\x98\xdfu\xd7]\xb8\xf3\xce;\xb0\x9d\x92\x9f\x98\x19RJ\f\x06=\x9c;w\xce\xed
(\vv`6\xbc\xb8\xb8\x88\xc5\xc5\xc5\xec\u3402w\xa7\xd3\xde\xf6`\xbeg\xcfn\xcc\xce\u030c\u05fa^E8\x1at0\u8b3b\xc1\x92\xbc:\x1foA\x96A&\xafb\xd3\x11|\x91\x18\x88\xc4d`G\xda\x0e\xa1\x84\xa9\xef\x8aO\xf1\xfaO\xa7\x91+;x\x12R\x95\xbb{\x15
_Q\x02\xf3%8\xf7\x94N\xf0\xab[\xda\nI=u#a\xbcj\x17\x89\xe5\xd9\xc9\u016be\x8a\x98\x8ak\xcb\xcd\xcf\xf2\x05\x94+\x17\x13.]3UL\xc7\xfa\x12L\xaf\xa3K\x97\xc0\u007f\xef/i\x837\xdd}X\x130d\x8d\xf1|\xb9\xa3\xc2\xe2d?\x18\"\u05da\xfb\x97\xe4}>\xb2\u0701\xb1\xb4sF\x10\u05ad\xd6\xdc\x19\xd51\xe7=\x9c\xd4\xcb?I4\x8e\x1d?\xfe\xe6\xcb`\xfec:>\xfb\u067f\x1e1s\xb3\xdd\xee\xdc\xf5\xd2K/e\xda\xecL\x93M\x84\xc5\xc5E\xdc}\xf7\xdd\u0631c7\xfa\xfd\xbe'\xe5\xfb\xcf\u007fH)\xb1\xba\xba\x82W^9\xe3\xe1\\\xf6\xa1\xa1\xe9\xe9\xe9daa^KY4\xac\xe8t:H\xe3\xed\xb6S#7\xa7Y\x18\a\x0e\x1c\xc0\xae\u077b*Y\x96\xf4\x03\x95\x8c\xfa\xe8wW*\u22f6\x9c\xa1D\x85p8\xab\xca-
\x18X\x0f\x0f\x998\xe5CR\f]`\xd7=M\xab\xdd1)P\x01Z\xab\xd4*\\\xcd%s\xd5u\xe4\xb1\x19T\xb5\xb3\x98D\x85\xa0\x94\xfa\xe4I\xf7\x84W\xb9\xca$\xe7\u01b3\xa6\xb03\r3\"\x9f&\x9d\u073b\xad\x18\x10\xaa\u0715x\x8f\x13\xe4\xd3\xe4[\x97\xf1\x13\xa2\xf1\xaa\xa2\aS\xed{A*\xca\\\x99\x15\x9aSE\x15\xd4\x11\xd1\x18;\x97[\xafsa\xc6K\x1b\x06\x84Dcz\x11B\x05\u0678?\xdc\xf0\x143
U\xc0k\x1b\x9bX>\xfd\xca5\u033c\x04\x00\x0f=\xf4\xc0e0\xffQs\xb3\xcf<\xf3\xd4\xe2\xf2\xf2\xf2\xf5\xa9\x17K\xd97\xfb\x86\x1b\xae\xc7-\xb7\xbci\x9b\x9e\xbf\xc4\xea\xea*.:\x83-
\xf6R)\xa2(\xc4\xe6\xe6\u01a3\xccx\xb1^\xaf;\x89o\u0299\x0f\x91$\xf16\xae\xcc\rv\xef\u078d\x03\xfb\xf7W\\w~\x1d:\x1e`\xd0]\xf7\xb24=\xdeSL\x029_\xcb]\xaa{9\xa5S\\\x95\x97\xd2'\xee\xef\xb6\x1a\xf6\xb7\xff\xec\xe5\x97Va\xdb\xe4\x99\xcfJ\xe8\xa5\xf2\xa2S\x04\xc1\xf2d%\xf1\xf8\xba\xc0\xd9uQ\xe15K\u007fF8n\xc0\xf2\xc8\xc8\x1d\x1eK\xe1\x12\x82\x91}\u007f\xe2\nI\x84KxB\x8e\u007f\

x9f\xca\xc0?\xb9M\\\xa4\u04cb\xa4\x13\x97h\xa4\ft\x99KE}\xc9*\xc0\xf5\v\xb2\xc1*\xf7\x021sa\xea\x17\xdeD4;\x9bg\xf6\u04d1\xdc{\x94\xba\x826\xa6\x16
\x83\xc8\xd9L[\xf9\xadt\xf2[)\x15u{\x03\xb4\xdb\xed\xbd\xdf\xf8\xc6W\xdf\f\x00\x0f<\xf0=z\xfe\xf9g.\x83\xf9\x8f\x92o\xfe\xeew\x1f\xb8\xa2\xdf\xef\xdf<\x1c\x8e\n\x14\x8b}\x13$n\xbf\xfdv\x1c9r\x04I2\u0716\u0dfe\xbe\x81\x8d\x8d\xf51\xc0SJ\xa1\xd3\xe9.\x03\xb8\x10\x04AA\x156\x18\f\x11\xc7\u0276\x05sc\f\x84\bp\xe8\xf0!\x04a\xe8\xde/Q\u0612\x03\xc0h8@o\xf3b\xb6\x95\x1d\xa3A\xca3\xf4\xa8\xaa\U000386be5\xe34\xae0\u0784\x9f\x86\r\xd64\\fn\xc6+\xd3J~\x9b*P\x8bQ!\x8es~+\x13\b\x94\t\x8bU\xc1E\xdd\xef\xe4\x19/&\x8d\xb9\xe8\x1b\xe4-
\x19\xe94#U\xbdp\x15\xd5\xebX5=\x19\xc1+\u007f\x80\xb6\xdaHU\xf6\x19J\xa6\xf3\x85]VYic\r\xc1\x8c$\xbb\x18g\x93\xab\xe9\x1f\u075bm\xf2\xd7\xc3.\fn_\x94\xe9\xd1}\xab\x02\xcf7\x97\t\xf5\xc6\x1c\xa4T\xf9\u0535\x90Y\x1c\xa3\x10\x02Z\x1b6\x9aw<\xfa\xe8\xa37\x02@\xb3\u0660\xab\xaf~\xe3e0\xffQ\x1e\x17/\xae\xdc\xf1\xd2K/\x15\x12m\xd2\xcaw\xff\xfe}x\xeb[\xef\x84R!\x92d{\x82\xdf\xc6\xc6FiR2o\x82\xee\u0631C\xd6\xebuYV\xdf\f\x06\x03\f\x062\b\u0272\x00\x00
\x00IDAT\x83mE\x19U\xed\x9a\x0e\x1f>\x84\u9a69\xb1\x0fl\xba\xd8&\xc3.:\xeb\xe7,\xb0\b\x0f\\\x04\xc1\x10U\x83\xdd\x16\xdc6y\xab\xa1\x9f\x9eC\x1e\xc0g\x14E\x95\x05,\x97\xaa\xec-
\xe9\x0f\xae`\x87h\xac\x06\xcf]B\x18[\xea\xd4y\x12oQE\xcf\xe4\x8dP\xcaV?\xf7{\xd3T\xab\u0523\x9e\x8a_\xd5\x05\xb7G@\xf3$\x90/7\xa7/\xd1[(}\xffR\x8f\x18_\x84\xb9\xb0\xf8\xb0\xb0\xbb5#\xc9\xd2G\xde\xee\x80Y\xdbIO\x14\x15\xe5\xe9k@yW\xd4\xfa\xec\xa7\xf3\xfeY\xc5\x0e\xd4[s\x90\xd2N;[\x03/k\xd6k\\\xffA\xca
\xd9\xd8l\xe3\u0631\xe3\xd7\x01\xc0\xaf\xfd\xda\u007fc\xb6\x1b\xc5\xf9\xba\x04\xf3\x17^x6\x05\x85\xc6\xf9\xf3\xe7\xefX]]-
LC\xa6\xc7\xcd7\xbf\tw\xdcq'\x00\xb3m\xab\xd8N\xa7\x83^\xaf\xe7\xed8\xd8\xf1\xe9\n\xbbw\xef\x0e\xa2(\x94eJ\xa5\xdb\xedf\xfc\xffv\xbc\xa1\xd2s\u06b3g/Z\xadVa'\x95]\xa7\x10\x88G=\xf46\xce\xe7\x8a\v\u05c02\xe9\xcc5Uh\xb1\xc9\x17\xf6Q>8\xefeQ\x92\xcb\xde\x1c\xab&\x9dv9\xe5\xd0s\x19\x1bc\xa2\xeb\n\xf3\x18\xc8\xfa\x93\x9dE\xb8\u326al\xa2\t\xca\xf2R(\x84O@\x8c\xd7\xc3\xf9.B\xf8\u050a3\x912
(E\b\x03\x81P\xd8\xec\x8b\xc9q\xa0<q\xb1\u068aIz\x8d\xfb\xe71*\x87'\xf9\xed\xa0\x9a\xb11\x04hE\xd0\xd2\x02:$\x81\x94\u02f5ufY\u033e\x04\xd4\xf3\xdaO\xd5+\xee\x9e4\x99\xaf>r\xa0vE`\xad\xb5\x00\x19\xd4\xc0Z\xc3\xc0\a~[\xf1\x87a(\xce_\xb8\x88\u04e7\xcf\x1ca\xe6\x03\x00\xf0\xb9\xcf\xfd\xad\xb8\f\xe6?\xe4\xf1g\u007f\xf6g\x00\x80\xfb\xef\xbfof}}\xfdm\xa3\xd1(\x03\xf2\xb4\x8am6\x9bx\xe7;\u07ce\x9d;w#I\xe2m\xfb\x02\xf7\xfb=\xc4qR\u079br\xad\x16A)\xb9<\x1c\x0e/R\xe6\xe4fo\u0635\xb5ulll\xa2h|\xb5\xfd\xc0|ii\x11\xadV\xb3\xa2\xbe\xb4\n\t\x93\xc4\xe8\xb5/
\x89\a\xce\x1a\x18\xa5\x00\x1f\xb6\x91k4\x89\x06\xc8a\x8f\xfd\xea\xab\x04&\\\x1a{\x1c\x93\xe5y\xcd4\xa6I\xac\x82\xaf\x14/\u007f\x8b\x8b\xe7\xbc\x05A\xe1\xab\xce\v\xc3<\x15\x89B\xd5\xc4\xc6\xf8\xa4,\x180\x12\x90M\x89\xa8\xa5\x10\x86@
\tJ\xe6>\xf4\xc4\x15\r\u05can$\x8fq\xe2\x97F\xf7\\\xc7R\x150W\xfe\x15\xc5&\x01c2{\xc5\x00H\x11\u0114\x84ll\x04\x01!\n\b\xb5@
\f\x04TV\xa1;\xddb\xd5$+\xe5\x9c:\x17\x9a\xb1\x949*\x02@\x106\x11\u0567\xb3\u0140\xa4\xcc\x1dH\xad\xfcQ\f\x87#lln^\xfb\u007f\xfc\x9f\x9f~\x03\x00\xbcp\xf4\xe8e0\xffa\x8f\xdf\xfe\xed\xdf\x01\x00\x9c={\u6593'O-
\xd9!\x1a\xce\xfcM\x00\xe0\xdak\xaf\xc5\a>pw\xc6\xe1nW\u03bf\xdd\xeed&[)\bZ\x1b\xd9&N\x9dZ~\xe4\u0529S/*\x15xJ\x17`mm-

\x1b\xe9\xdf\u0395\xf9\x8e\x1d;0==3\x0emi\xf1\u010c\xc1\xe6\n\x86\xed5\xab\x15N\xa9\x11m\xd
cT
\x95\xfa\x88\xc5\xe4\xf7\xb1\xd6${\x9c\xabo\x8d\x9am\u0569dKR\x04\x95,6\x8f\xb6\xc6\xcf\u025
c\x0f\x15\x95\x13\x19\x85K\x15\x04L\xe9\x19y<gt\"xzC6
\xa7ZiJD{\"\x04\x8b!d$A\x82\x90Z\xe8\xb3\u007f}\x19h\x93w\x05v1\xb41u<A;8\x89\x17\xa1\t\xec\
x17c\x12AUY\xecsi\xffAn\x97Q\x13\x88\xe6\x034\x96B4[\x01\x1a\x8a\x10*
\f\x05\x02E\x90\x82\xb2\x1dV\xb6\xa0W%\xfae~\xff9\xfd\x92\ue10d1
\xa1\u041c\xd9\xe1b\x13\x01!\x03{O\xe6\x13\xd9\xc4$x\xb3\u075ey\uaa67n\x06\x80\u007f\xfb\x8
7\u007ft\x99f\xf9Q\x1dG\x8f\xbex\xd7\xf2\xf2\xa9\xac\xf2\xf6md\xef\xb9\xe7\xa7q\xed\xb57\x809\
u0656\xf2=!\x04\x8c\x19as\xb3\x8dr\xf043s\xb3\xd9\xc4\xec\xect\u007fccs\x98\xe7n\xd2\x04\x9e}
{\x82\xf9\xdc\xdc\x1c\x16\x16\xe6|[u\xf7}\x93\x01\u06f0\xbb\x81A{\x1d\x82\xf3\xc6V:\xaa]\xfc\xca\
x05\u02570\x85\xf4>\xacN\xf5\xe0\x9agF9n\x9e\xaa\xc2\xe0\x90\xed\x02\nU$\x97\x99\x80\x8am\
x02\x95\x1a\x9e\xa5\tN\x06\x17\n\u0182\xbb\x1f<\x1e\xa6b\xa2t\\UB\x85\x06\xb1\x0e\bIK\"\xdc\x
11!\x9c\r
f$h1\x80\x98\x92\x10\x81\x00I\v\xeae\x0e=\xd5a\x17TC\\E\xdb\xd3\x18\u060eW\xdf\xc5\xf0\x16\x
e6I\xb2E\x14'@\xa9\x94?\x97>\x8b\x80\xa3\x8b\bAK!\x9cRv\xc71\xab C\x01!\t\"
\u021a\x80t\xa9D\xcc\xc5\x1d\x12\xfbrt\xdf\n\xdf\x05\x84\xa3\xe0\x9eh
d\x80\u9e7d\x10$a\xb4v\x01\xebA\xf6\x9c\x86\x19R\x05|\xe6\xcc9\xf4\xba\xbd\x9fb\xe6\xc5S'N$\
x97\xc1\xfc\x878^~\xf9X\xf6\xe7S\xa7\x96\xdf\xea\xab:\xd2\n|\u07fe}\xf8\u065f\xfd\x10\x00\x8c\x0
1\xe5v\xaa\xccG\xa3\x18\x9dN\xb7\xb0s
\u05d4i4\x1a8t\xe8p\x90$\x894.\x9d&\xa5Y:\x9dv68\xb4}\x9b0\x8c0\f\xb1\xb8\xb8\b%U\x11<\xbd\
xff\x8d\xfam\f;k\x99\u01ca\xc8\x1a\x96\\P5\x14E-
\x13\\\x14\xab\xceB\x10\x92\x9aDR\xb7\xde\xd8,\x8b\xcd@\x1f\x18\x99\xa80\x0e\xee+\u034b\xd
4CU\x1d\u0293Kx\xf67\xf6\x1e\xd7N\xbeR\x851\xa9\xf6\xcd\x11\xa9\x88\x8a\x86\x80$\x10\b\xe7\
x034g\x94\u0376\x04A\xb4\x14\u010e\x104\xab\x104\x04\xa2\x86DX#\xc8\xc0\xaa3\xca\x16\x88\
\-
\xb6\xc1\x98\x82\xe7UD\xe4\xb1'\v\xe2rhK\xc5N\xa6J\x97.\xa5@(\t\"\x14\x10S\n*\x10v\r\xae\vPS:
\x87D\x86P\x04\n\x90\xc9\x14\x99`\x1b\xe7\xee\x1a\xb3\xf5?\xf3\ua9ca\u05d6\xc0\xc6@H\x89\x
a9\xf9\u0750*\x84\x8e\x87\x10BB\x85\xb5\x8c\x8f'\x10\xa4\x90\xd4\xeb\rq\xf1\xe2\u0291o~\xf3k\
xfb\xf1:;\xb6\x1d\x98\x1f:t\x85\x03\xb4\x8d#\x17/^\xbc15\xa0\xf2\x03\x9c\xef\xba\xeb\u0778\xf1\x
c6\x1b\xb7-\xc5\x02X\xbf\x98\xd1h\x84^\xaf[Y\xed\xd4\xebu\xec\u06b5\xdb-
F\xc5\b\xba~\u007f\x80~\u007f\xb0\xcdo\x1d\x03\xa5\x14v\xed\xdam\u524c\x92\x0e\x98A$0\xe8
m\xa0\xbfy\x01\u0485Xg\xfc\xae\xa3A}\x8e=W`O\xe0\xae=\xe0Kk\\\xa9\b\xc1\x9cB\xb8+B4-
\x11\x04\xae\x9a\x93\u00abP=m\xbb(\xf3\u9503\x00y\x89N\x15\xa0\xc7>\xe0\x15x\x96-
p\xb24\tC%v\x19%\x83)\xbf\xfa\u0512\x10\xce(L\xcd\a\x90\x8a\xb2\xea\x94\x00P]\x82\x16C\xc8\
xf9\x10\xb2%\xa1\x9a\x12a3\x05v\x01\xa1\xc4Xq\xfc\xc3\x05\x9e\xd2$W\u0742\xb4r\xac\xd7P\x9
eG\x12\x84@\x00R\xa6\\\xb9\xb0k\x9e\x81m~\xb6\xa4\xbd6W\xd8hg>f\xdf?\x02K\xf7\xe5/Zd\x1b\
xa6\xe9i\xd8\xde\x1a\xe7\n8m@$\u0418^B\x105a\x92\x11\x88\x04\x82\xb0\x9e5p]\u07ca\x12m\x
10\xc7\u0261\xfb\xee\xfb\xf6^\x00\xf8\x8b\xbf\xf83\xba\f\xe6?\xe4q\xff\xfd\xdf}\xfb\u0673\xe7\x9a\
xddn\u01fd9:\x03\u024f|\xe4\xefA\xa9\bI2\u07364\x84\xb5\x8b\xd5^s\x96\xb2b\xcd\x18\u00edV\v\
xfb\xf7\xef\rF\xa3xl\ua0d91\x1c\x0e\v\x00\xbf=\xafQb\u03de\u0748\xc20\xf3\xd1@\x01\x9f%F\x83
\x0ez\x9d\x15\xa7C/\x19\xb2O\xf2*\xa9\xa0_\v\x01\ri\xb8\x80\"\u0526\x14\x9a\xf3!\xa6f\x034\x97

B\xd4\xe6\x03\x845\t)\x01#,\x18he\xbf\xd8\xe3\u05ab\x16\x86q=J\x95\xab\xa0w\x19c\xdb\u007f\x06ck\xba\x82_\x05l\x02\xb6*W-
\x85\xd6R\b\x15\n\x97\x1f\xee\xe9\xaa\rCD\x02b.\x80\x98Q\x10\x91\x80T\x02A]
\xaaK\u051b\x12Q]Zj\xa9\\\xa5W\xf6\x05\xb8\xe2\xef[xLr\xb1/Q\xb4\xdf\xe2\xca>\n\x13
\x05!\x10\x04D\x041\x13\x80\x94p\x13\xacNG\x1e\x120\xa3@\x91\vr\x16v\xc7\x01o\xc7e$`\x04\x175\xf5Y\x95^\xbc\fv1zF3\xc2h\n\u0359\x9d0\xc6\xc0h\x9di\xcd}\a\xc6
\bp\xf4\xa5\x97\xf0\u02993\xb7\x00\xc0G>\xf2K|\x19\xcc\u007f\xc8\xe3\xf1\xc7\x1f\u007f\xc3\xfa\xfa:\xe28v\xa1\u01f6\x8c\xbb\xf9\xe6\x9bp\xd3M7\xd9U[\xebm\xfd\xe2\xa6\x15\x02{\x83\x1f\xcc\xe0(\x8a\xd4\xc6\xc6F\xbb^o<U\xaf\xd7\xeby8q\x0e\x18v\xa4\x1f\xdbzp\b
\xec\u06b5\v\xb5ZTIK\x10Y\xb3\xad~g\xd5\xf3\xb2\x9b\\\rV+\xe9r.8\x93A3AJB\u0610\b\xe7\x15\u009a\x80\x02C\xd5$\x82\xc5\x10\xe1B\x80\xa0!\x01E\xd0d+\xdcT\xbfl$\x81\x95\x03t\x81\x92/I\x85Sb\xe5_\u02a1\x11\xb4%{\x81\n\u00a6\x9c\x0f\xe1\xafq\x9a\x19\x14I4\x96B\xa8\xa6\xcc\xf3;\xcb\u02dea\x90\x02\u4302\x98\x0f
\x1a\"+\x18d@\x88\xea\x02\xb5\x86\x84\x94\xe4\x16\x03L\x1e&\xe2\xf2\x8b<\u10b8\xb8\x1c\xb1G\x92\xe5\xeb\xf3x\x9b\x97\t\x10d\xabr\x92\xb0TQCT\x8e\xfdS$\x80)\x05\x04\x04\tF\x10\x10T\xe8\x02*\\C\xd4\b\xbb\xe0\xa5\xd6\r\xf9g%\xd7\xe1\v\xca-
\x8cu\x92@\x85u\xcc\xed<\x04!\xadlV\b\u5a16|GDRbmm\x13Z\x9b_4\xc9p\x0f\x00<\xf4\u0403\x97\xc1\xfc\xb5\x1e\x8f=\xf6\xb0\af\xed;\x8c\xd1.?\xd3d\xc0\xfd\x9e\xf7\xdc\xe5\u0329\f^\x8f\a3\x9b0\x8c\xb0\xb6\xb6\xfer\xb3\xd9z\xa9\u0468\x85y\xc0s\xfe\xb8^\xaf\ufa0a\xed\f\xe6\xc0\xee\u077b\xd1l6\xbd\x81\x0e\xff\xb3N\xd0I\x8c\xee\xfay\x8c\x06]\x90\x90@\xc9$v\f.x\x1c$\x8b5!AH
jH\x043\n\xa2)\v\xce|\xa4\br.@8\x1f
j*\xa8\xd0\xf2\xb1ieh$A\a6\\\xd9\u0212\xf7\u01d8\x81k9\xad~\x92\x94\xe2U\xf4P\xbc\x85)\x83?.\xf1\xf1)\xa8\x93\xa5\x8e\x82Y\xe5R\x88\x8a\x96\x01\xec{\x95\xbb\x1c4j)\xd0|\x00jH\x90\xcc\x01;\xacI\x84\r\t\x11\x12\x8c\vv.\xbe\u063e\xddm1:\x88\n\xb4\x10U\xd2]D\xe5\f\xd4j\xbf\x17\"@\t\xabNA$A\xd3\xca\n\xe5\xb9\xc2!\x8c\x004$\xa8!!$
\x89!%\x01\x8a`\xbckO\x9b\xbb\u0185\xd1d\xbaq\x1e\xff\xddl\x12\xa8\xa0\x8e\x99\xf9\x03\x90*@<\xe8B\xaa\x00a\xd4p\xe7m\a\x89\x88\x04\xb4a\x9c^>}\xc3'?\xf5\xe9]\x00p\xdbmw\\\x06\xf3\xd7z\xbc\xe9M\xb7f\u007f>~\xfc\xe5\xb9^\xaf_\b;\x06l\xfed\xad\xd6D\x1c\x8f\xf0z>\x1a\x8d\xba\xba\xf2\xcaCJJU9\xe9ii\x16\xb3\xadi\x16\x00\u0635k\x17\xe6\xe6\xe6\x9cE)\x8d}\x98\xd9\x18\xb4WO\xa3\xbf\xb9\x02!TA\a\\\xa5N\xe6\x02\x90\x97\xab}\a\xe4u\x89\xa0\xa5
f\x02@Z\xc9c\x01\x9f\x14AN+\xd4\xe6\x034[\x12\xb5P
\x90\xb6R\xcb\xf8h%\xa0C\x01\xdej\xf2\x06\x93H\xe2\x8a\xc7l\xe5\x9a[\xad\xad\x19_\xcel\xe61DC\"\x98\x0f,(\x1b*\x00\xb9\xf103[z\xd8y\xbd\xd6$h!\x00MI;x\xe3\x1e\xa4\x02BT\x97P\x01\xc0i\x8a=\xa1\xdc\xe4\x98@\x00q\xd5*\x9b\xe3n6\xfcU\xa5\x9e\xf7\x8c\x81\x89
\xa5}ohZ\x82\x1a\xa2\xba\xdf\xc0.\xf89\xe5\xcf#@\x10CR\xdab\xe5,z\x8e\x05\xdb0\x8f\xec\xb51\xee+U\xa8\x98\f\u05d95\x88\x14\x9a3\xbb\x11\xd5g\x90$#\x00\x04)\xc3,\xfc\x82\\\x9f\aDx\xf9\xc4I\xb4;\x9d\xd7U\x94\u0736\xa4Y\x8e\x1e}\xf6g\xd6\xd7\xd7\x0f\xb6\xdbm(\xa52\x8a\xe5\x96[n\xd9\xf6\x8d\u03f1\x8a\x8c\xaa\xff-
\bB\x04A\x1dJI7\xe9\t\xf8\x01\x15\x9dN\a\xa3\xd1h\u06cf\xf4\xcf\xce\xce\xe0\xe0\xc1\x83PJ\xa0<\xff\u01ce\x1f\xd8X9\x89\xf6\xdaiH\x15\x8cU\xabT\xfe \x13\x15\xb6\xee\x19^\x01
\xc1\b#\x81\xb0!
f\x94U?\x94'@]\u04cf\x14A\xcd\x06\b\xe7C\xd4\x1a\x12\x8dH\xa0\x1e\t\x04\x82\x9cc\x1f\xc3(

\t\x85\x954z\xa4|\xbeo\x98\xe0\xa4H\x15Z\xeaI:\xed\x82+
O\xa0c\x9cZ\x8b\x192\"\xd4v\x84\bZ\xaa\x10f\\\x984\xa5\xaa6\xad\x03\xb9P\x00s\x01hV\x01\x91
\xbd.A\x84@\x11\xa2\x86\xb4\xfc{\u5fa1\xca[\x86's\xe0\xe9>\xa3\xf0\x92P\xf5\xf3\t@)XuJS\xdas\
x134Y\xaa\xee\x16(\x0e\x004\xad\xa2\x85Rq\xbfa\x10\xe7\x03D,a+\xfcBw\xc26=\xcb=\x11\xd61\x
ea\xad\x05\xcc,\x1e\x86\xd1\x1a\xc3A\a*\xaaC\xa9\xc8=.=s\x81\xd5\xd55\x1c\u007f\xf9\xe5w1st\
x19\xcc\u007f\x88\xe3\x1b\xdf\xf8\xd6\xc2\xc6\xc6f=\xedF\xa7\x14\xcb\xdb\xde\xf6V\x1c<x`\xdb\x
c5\xc2\xfd
\x87\x94\xf6\xa5\xb7\x03CN\x99\xec\r\x0eu\xbb]\x8cF\xa3m\xedi\x0eX\x89\xe5UW]\x05\x95\x01u
1\x1fS\b\x81\xf6\xda+\xd8\\[\x86\x10\xaa\xf2\x03L^eJ\x13\\>\x18\x80\x90\x02A$
\x9a\x12\xd4\x14\u0632\x1cf\xab\x83\x14-\t9\xa3 k\x02\xa1\"\xd4\x03B-
\xfb\xf97\xae\x99\x16\x88\xdc\x0f\xa4\xb8ba\xe2\x84\xd1V&]\x99/9\x15(j\x1a{\\i\xb1\x92\x84p.@8\
xa7\x1c54>y\xeaOb\xd2\xd8\xf9\xb0\xe5\u0195k\"\xce*
\xccJW\xdbg\xa8K\u0210\nz\xf4I\x84?a\xf2\xda\u00d4S>\xbc\xc5N\x86\x85U\x1cII\xa0@\x80f\x03
P]nyO\xa5.\x88\xd0\xda.H5\xe7\xa3b\t\x95\xac\x02O'?\r\x8c\x1d\xfb\xf7\x16\x1f\xbf\xa5\xcd\xc6\x
c0\x18F\x12\x0f\x11\u05a60\xb3x\x10\x00c\xd0Y\x87\njPQ\x03\u0326d\x8fK\u8d3b?u\xfa\xe5\xa3
!\x00|\xf3[_\xa7\xcb`\xfe\x03\x1cI\x92\u0726\x94\xac\xc7\xf1\xc8v\x9eMZ\x99\xbf\taX\xc7h\x14\xbf
\x8e`\x9b*+\xf5\x14\u0327\xa6\xa6\x10\x04jLOnm\x00\xe2m[\x99\xdb\xddQ\x82(\x8ap\xf8\xf0!DQ*
O\u0307P\xec5\t\f{\x1b\xd8\\}\x05Z\xfb\x1e4\x8cr\x13\x8d&#\xbdU5\x84\x04\x11\t\xa0\xa9\x00)&\x
b3\x1f)mj\xec\x1d.\xa6\x14\xc4l\x00\x8a\x04\xa4\x04\xa2\x80P\x97\x84\xc0\r\xdah\t\x18\xe5\xa9\\\
xbc\xc5e<\x1d\xf9\xd5|\xbb\x8a\xed\xa7\xc2w\xfc$\n\xbb\xeb\x00\xa2)\x85p)\xb4\xea\r\xe6\xca\n\
x9a\xb6\xb8\xb7\xb2\xbf\xa7\u0297\x96\x02\xcd\x05v\a\xe3\xaaW)-\x8f.BQ\bw\xf0\x17\x84-
]\x12\xdd\x0e\x8a\u0299\xae<~\x9eL\xee}\x13\xee\xf7OK\u0434\xac\xfa\x81\xf1\xb5\x80\x19Fk\v\
xcbu\x01\xad\xa8\x90\x13j\xc1\xdd\x01\xbb\xb6`n\xd8@\x1b\xe3vC\xaer7\x9c\xed\xe4u\x12C\xca
\x10\xd3s{\x10\x84\r\xc4\xf1\x00\xc6\r\x14\xf9\x94\x93\x10\x12q\xac\xf1\xfc\v/\x04\xdf}\xf8\xd1\xd
b\x01\xe0]\uff0b/\x83\xf9\x0fp\xf4\xfb\x83}\xcc\x06I\x92p\n\xe4\x87\x0e\x1d\u0111#G&R\x17\u06c
bZ\xb1\xa1\x12J)\x84aP\xf9\xfd\xd42vqq\x11a\x18\x8e\x81y\x1c'\u06deJ\xd2ZC\xca\x10\x87\x0e\
x1dB\xb3\xd9\x04\x8d\u036b\xb8\x0f\x92N\xd0^[\u01b0\xbf\t\x92j\xecs<1<\xce\xeb5\x92\x84mf6
%P\x17\x97\xd4L\x97{|4%!\xe7\x02\x88\x9a\xb4\xcd8E\xa8)\xb2L\x04;\xb3\xa7\xb0\u031f\xfb\x13
G\xfe\t\xb9\xa6$U\xc78\xd3\x18\xdf_0\xcb\xf5*\\\xf7h\u05e8\x8c\x96\xacf<\xdd\x11\x14\x1d\x82+\
xf25'\x82:\xf2\u0160)m5\\s\xce\\l\xd7\xc10\x12\xa0\x80,%?&\xe8\xe1\xb1\x19\xa6\xe2\n\x96n*\x8a
*\xa0\u00a2E\x16]\xa4
(bP]B\xcc\a\xa0\x80r[\xe2\xca\xe7\xf7\x831\x9cQ\x96\x02\x92\x1a\x90\x10g\xcdN\x18\xb7+H)\x97
\xb4]K9\xc5\xc7\xc6d\xbd\x01\xb2Sy`N\u041a\u074d\xe9\x85}`\x93`4\xe8@\x05a\x1e\xf8\xec\xce
A\x1b\xc6\xda\xda:^|\xf1\xa5\xb7\\\xa6Y~\x88cuue4\x18\f\x01\x10\xa5\xc3BW^y%\xf6\xed\u06c7\x
ed\xdc\x14,V\xad\x06Q\x14\xa1\xd9lU\x9e/;\xbd\xe2\xd2\xd2R\x10E5\x18w\xa1\xe9c\xe38\xb6\x9
e\x12\xdb\xf8Z}E\xcb\xc2\xc2B%\u0352n\xc6\u06eb\xaf`\xd0\u07c0P\n\x13`\xa2\x1a\xc5\x1df\xaa
@@\u0525\xad6\xabT\x10\x18\xe7\x97\xcb\xcc\b\xb5,\xa0\u021a\x84
\x82\x14@\xa4\x04\xea\u03b0J+\xabt\xa9<1*k\xce\xc7\u03c1\xb6\xa0\x0e\u0199\xe8\x94\xcf\x06
\u0080\x10\xcd)\x88Y\xe5=\xfc\u0485
OlV\xe6\u007f%fPC\x80\xe6\x02\x88\x86\x84P\x04\x99r\xe85\tR\x02\xc6\a\xeeB\x9a\x0f\x8d\xc5
o\x94\u007f\x0f\x8f-
d~\x8f\xc35=\x03\xf7\xfb\xeb\x12\xd0[Y9\x96^'\xf6<\xcc#\x01Q\x17\x10\xa2\xd8*\x17\xe9Tq:8$\x0

4\x84\xb7\x98\x12R\x99\xa2\x80
\x01\x9d\xc4h\xcc\xee\xc4\xcc\xd2!\x18\x9d`\xd8\u06c4TA\xd6\xcfI3D53\x12mp\xee\xdc\xf9w\xa5
g\xf4\xf0\xc3\x0f^\x06\xf3\xd7z\f\x87\xa3l\xe23\xb7[\u074d\xa5\xa5\xc5m\x9e\xc0S\x04\xba0\f1==]
:\xdf\u051a@\x13\x00Q\xafG\x17l\xd1P$mm\xdaP\xf2:\xb8V\xc6\xcc\xcc4\xf6\xef\xdf?^\x99{
\xb1\xb1z\n\xfd\u038a\xfd\xd0\xf0\xa5\xdcW\xbc\xa7 @*\x01U\x13\x10-
\t\x8e\xc4%p\x8e'>+\x11\xac\xdcm.\x80\xacK\xeb\x01B@(\b\ra\xa9f\x1d\n\v\xe8\x84\n\xbd\xb8\x1f
\r?l\x01R\x95Z49\xb8\"\b\b\xe1t\x00\xb1\x18\x82\x14Ys@\xc7Qp%\xe8\x95'-
\xcb\xf9\xa5\x9e\u076f\xff\xad\xba\x80\x98U\x10N\xbah\x9b\xa2@X\xb3>(LEs\xafl\xec\xc7\xd8N\
x83\x8a\x8b({\u0222\x84m\xbe\u04b4\x82\x9cQ\x15\xd4\u04c4EW\x00`\xe3\xeck\xed\xffCEh\xce(
4Z\x12\xf5\x10\b$l#\xd4x\x80\x9f\xf2\xf8i\xc2\x14\x18\x865\x8cvT-
3\x8cI\xa0\xa2\x06\xa6\x16\x0e@\x86u\x8cF}h\x9dX\xd9,\t/\xb1Np\x1c'X^^~#3\xd7\x00\xe0\xd6[\x
ef\xb8\f\xe6\xaf\xf5\x90Rf>%)\x98OOO\xa3^o\xbc.T,)\xcdB\xa4077\x8b
\b\xe0W\u0756\xa20\xd8\xd8X\t\x9e\u007f\xfe\xe8\x97766\u06f5Z\xbd\x10H;\x18\f\u0728\xff\xf6\x
06s\xadc\xcc\xce\xce\xe2\u0211\xab\xac?\x06\xa1\u020b;:\xa2\xbd\xf6\n:\x9d\xf3v\xab[\xa60\xaa\
x02\x9d\xddC\x94\x04TD\x90-
\t\x9aRn\xc2o\x82\xcf\t\u00c5\xf5\xa2Z\x80\x91\xfe[\u04daU\u0256\xb2\n\vE\b\x94@]Y\xd3*\x1d\t
\x98\xc0sW\xf49\x9b\t\xa1<\x05m\xfa%\x17\x18\x02\t
L\x1b\x92\x8b\x01\xe0\x86\x83\xf2\x11\xfc\xaa&g\xfe/\x95\xfd\x05\x1e\xcf\x02%\xa2|\u0769\t\xd0|
\x00L\u0641*!\xad7z\xbdf-
g\x89\xc6s83\x9f\u022cb/\xee\b\xd2\xde\x04yzI\u06ec&H)\xecnh)\x80\b\xa8b\xc7Q\xd4\xed3\xf2\x
b8?\xd6\x1aF'\x96\x0f7\x06\x82\x18A\x04\xa8\x96@P\x13\x88\x027\x80\x94\x017\xf2&\xa9\xf1\x1
6\x027=\x9e\xfa\x9a\x83\x01\x9d\u0118\xddq\x18\xd3\xf3{\x91\x8c\xfa\x18\r{\x102\x80H\xc7G\x8
9\xa0\x94\xa2N\xb7\x87\x13'OM\u007f\xfb\xdb\u07f8\xe72\xcd\xf2\x03\x1e\x8dF\x1dR\xca\xd2M)\
xb6=\xb0\x95h\x14\x00\xc0\xcc\u0334m\x0e\x96\x8e$\xd1t\xe1\xc2*\x1a\x8d\xc6@\b\x91\x94+\xf
0tXj\xbb_s\x92h4\x1a\r\xec\u06f7\xcf)Z\xaag8G\xc3\x1e\xba\xed\x8b\xd0&\xa9l\x82\x8e\xcdQ\x12
\x89\xa0\xa4\xb0\xb4\u0234\x0f\b\xd5\xceQ\xa9\xcd\x15\xc0\x93\xeb\xf3\xf4W\xd6\x04h^\xd9\xe6
\xa8\x04
\xac?xMX\xa0M\x02\x01\x13\x8a|\xb0(\xe5^\xb7|?\xe8U\xbc[\xf6\xf9\x02A\b#\x011\xa7\x80i\x8fz\x
e2\\\xbf]\xad+a\xcfY\xb2\xaa\x84\x9eddnA\x17\x01\x81f\x953\xb4\xb2^)2\x14\x88\"\xfb%E\x89P\x
a9Xo\xa9\xfc_\u02af\x9d\t\x80$(a\x9b\xd5b1\x80\xac\x8b\t@>.+\x05\xb9f\xa5N\xc0\u0680\x8dv\x
92D\a\xc6\x01@!A(\x810\x90P\x822\xee<S\xbf\x19\xed*\xf1\xb4?\x95\x1a\u0739\xe7\xd7#L\xcf\x
ed\xc5\xcc\xc2\x01\xb01\xd0\xf10\xf54\xcf_b\x12\xd0\u06a0\xd3\u9d3e\xfd\xed\xef\u0718\xf7\xb2\
x06\x97\xc1\xfc\xb5\x1csssQ\x14E`\xce;%\xb6J}\xfdM}\xb6Z-
\xd4\xeb\xf5\xe2N\xf8\xffc\xef\u0363,\xbb\xca\xfb\xd0\u07f7\xf7>w\xa8[Swuu\xf5(uK-
\xa9\u0552\x00\v4\xa0\x81Qf\xe6a<\x80\x00\x93x9\xb1\x8d\x89\x87\x84\x10\x9c\u007f\xf2\xfe\xc
9K\x9c\xbc8\xce\xf3\xb2\r\xbc\xb0\x1c\x9c\tX\xd8\x01\x83\x9e\x102\xc6\x0fd\x03\x92h\t\xa3yhuK
=O\xd5\xd55\xdc\u1733\xf7\xfe\xde\x1f{8\xfb\u073a%!/\xbf\u040a\xea\xacURu\u056d{\xcf\x1d\u03
b7\xbf\xfd\xfb~\x83\xff\xc8\x14E\x81}\xfb\xae\xcc\xda\xed\xb6\x1c\xee\u078a\xa2\x80\xd6\xe6\x82
/\xe6\xd6\x1a(\xd5\u0116-
[\xd0h40\xdam\x10`]\xa2\u05dd\x87\xe6\xd2\x05\x03\xd4:\xb3a\xba\x9f\u00f33\tg\xf3:\xa9\xa2T\x
bd\xa6\xeec\x1a\x01\x05$\xf0\x04\x0fA\xde4T\xd8\x1a\x02\xb4A\x81&\x14D\xe6.\xfaL\x12\x1a\xb
e\xa0[E\xb0\xa9RT8\xb5\xe1\xaa::T\xf0h\xcd e\x1f\x94

\b\rE\x10\x93\n4S\xc1+\xa9\x00sT\x02Sb\t\x81\xe7\x97
\r\x95\xfa\xa1\xe7N\xbe\xa0cR9\x11\x8fpP\x8bj8\ua9d4U\x90G\x88s\x1b\xbe\xb3\xe1\xfdA\xb4i!o\
xa2\x95\x11\xc4F\x055\x95\xd5\xf8\xe8\xa3\xf1\x9b\xfaN.v\xd26\xd0\x0f]\x97\rk\x01\xc1\xe0&9\xd
a)\xac\x0f\xe7\xf0\xcc\x16k\xc1\xda8.zl\xaal\xfc^\x04\xd8\xc72\x94ja\xc3\xe6K\xd1lO\xc0\xe8\x1c\
xd6j@\u0238xy\xb1\"\x17E\x81\xe3\u01cf\xbf=\x9cn\x96\xb5\u058b\xf9\x8b:)!\x9e\xb1\xd6B)E\xa1
\x8b[ZZB\xbf\xdf_\u0571_\xe8G\xbb=\x86V\xab=\xf2w\x83A\x1f7\xdexC699!\x86#\xf1*\x98\xe5\u0
09f\r\x00\xc0\xec\xec,:\x9d\xb1\xd5\xe5\xc5\xeb\u05ad5\xc8{K0\\\x823\x91fP\xa0\x96\xa4\xe9\x8b
A&\x9d\x9d+\x8fI\xe7\xd3!B\x9e\xa3\xaf2\xc4\xd1pkTM\x8dl\x88U\xbfO~`\x01(\xe1:\xd5\t7\f$\u12a
d\xf4\xf7a\x15\xc14\xbdR\x14\x14\xdd\xfcj@|(n\xabL\xc4\xea\x10\x05\x93\x83\x1f2E\x0e\xbb\x9ei
8\x0e\xb5\xad\xa0\x82\xf8\xf4j\u0557Gw\xe8#\x81\x9e\xbai\x18Q\x8a\x87$\v\x99\"\xc7r\x99T\x80r
;\x13\xca\x04dS\"k\b(\x95\xb4\xca\u00b3n\u05b0\b\x8e*K_\x04\x95$\u0229\fjc\x96\u030d+\xc6Im\
xf1\x1e\xe6wz~9\x9b\n/\xaf
\x12\xcfb\x91N\xf0\xc5`\bb\bI1g6\xae[\"\x9d\u06cc\xb04\xb6%f\xb6^\x8e\xf1\xa99\xe82\x875\xdaw
\xe6\x14\xbby\xa5\x14\x9d=s\x06\x8f<\xf2\xe8\xe6s\xe7\u03bcj\x1df\xf9[\x1cJ\xa9\xfd\xd6r\xb7\u0
468\xe0\x89\x03\a\x0e\xe0\u0529S\x90\xb2\xf1\x92Q\u007f\xbab\xdeF\xab\xd5J0\xf3\xc0\xf7\x15
$\x84\x90;w\xeez\xaa\xd9l\x9ck6\x9b\xb5\u03bd\xdf\x1f\xa0,_:|\xfa\xa9\xa9)l\u0738\xd1C\xe4\xc3
v\x87\x04f\x8b\xa2\xbf\fkJgO+\x87\x82\x13\"F\ue2dd$\xa0)\xc0S\xd2\tG,\xeav\xe2UtP\x1a(\x14\xb
b;Ak\x80\x1ea\u06ddt\x82\x94\t\xd0T\x06\x9aR\x10\x99\x80\xf4\vJ\xac;\x020\x99\xa8V\t\x1a\u00a
91\xdc\xfe\xf3PG\x8dJ\xb4\xd3$\u020e\xb3\xafE\xa0!bDm\xe3\x90\xda1\\\xbbG\x81/\xc3\x05\x96
F\x92\x06k\x1d\xb4u\"%\x9a\x92\xa0\xa9\f\u021c\xf1\x18Ig\xd2\xd5P\x02*]O\x12\x16\xce(\x83\xe
2`Q+$
\xc7\xdd\x1c\x802\xb1\xea\xbcG\xa6\x15%\xef_eN\x97\x0e@\x1d\xccb=\x06\x0e\x82\xa3Z*\xb7\x
83\x12\xd2\xef.\x84\xa3\xfc\x92W\x11\xdb@]\xf4CR\xf6n\x89\xe1\xff\x9d\xa99L\xcf\xee\x82\x10\x
12V;C?!\xbd\x93\xa2\xef6\xba\xbd>\x16\x16\xce\xed\xfc\xecg?{C\n\x9f\xae\x17\xf3\x1f\xf1\xb8\xe
d\xb67\xcf\xcf\xcdm6Y\u0588\xea\xee\xfb\xef\xff>\xee\xb8\xe3N7\xc7i\xb5\xf0RI\xceN\x8byz\xd5\
n!!\xa5\x90Dt\xa8\xd5j\x9d\xe8t\xc6\xfds\xaa\x02*\xf2\xfc\xc2W\xba\x06\xf5\xea\u018d\x1b\xb0m\
u06f6h\xff;b\x88\x80\xa2\xbf\f\xa3K\x10I\xb7}W\xae\x80\x84\xad\xbc\x94\x84,#\bE\x80\x02xR\x82
\xc7%\xb4e\x94\xb6N\x88s]\xa7+x\x14\xc1Z\xae\x93\xe6\x88kt\xb9\x14\xaap\x85\xb2\x12\u0650\
xf2l\x8f\r\x95Z\xb4\xa5\xc8\xd58v\xe7g2\xaa\x02.j\v\x11\xd7\xf0c\x1e^\xc8\xe0\ng\xd6\x12\x90\xb
e#\xa7\r^\xa8bQs.X\x9bQB\xabb\xa9y\xd5\"B\u03c3\xdb\xd7=^\x9c\xe4\x9da\x05\x81'=\u44b9\xe
2H\x19\x81\u048cQN\x92\x93\u05a2\xb9{\x18J\x8dI\xa8M\x19\u460c\x8b\x00%\xbb\xa2\x10\xc2<
\u0681\u04a7\x18\xd9\xca_\x98\r\x83\xad\x89E\x18\x81?.\x1d]\x91}\xe83\xc8\u07f7\xf7\xe7t\x8fc\
x9cR\x14U~\xb0\xf1\"Dk4\x88$6_\xf4\n4\u06d3\xd0e\xee\xc2S\x84\x02\x91\xf0<u\x03kM\xd9\xed
v\xc5C\x0f=r]\xf8\xbc_\xa8\xb5\xe7\x82,\xe6\x97\\r\u065f\xef\u0739\xf3\xc0\xcc\xcc\f\x88D\x84Z\xf
e\u077f\xfb\x1d\xdcq\xc7W
D\x86V\xabs\xc1\xa6\u05e7E\xae\xd3\x19[\x05?\x00\x801\x1a\xa7O\x9f\x06\x00\\z\xe9\x9ef\xa7\
u04c11&\x8a\U000965d7\xd1\xef\xf7_\x12\xc5\xdcZ\x8d\xa9\xa9il\u077a\xb5\x96\xbf8l@\x95w\x1
7at\x0e\x9fs\xe6c\u037c\xf22#\u0226p\u0635$P\u01e96\xa1\x04r\xcd\u022d3\x99\xa2Z\xe5\xa3U
\x1d0\x8d\xe8\xc4G\x03\xb5\xc1X\xa9bP\x10\x11\xc4d\x06\xb91CcL\xa2\x99\t\xb4\x94@[\t\xb4\x
a4[dB\x92\x11\x10\n\x1c\xd5pn\x1e\x02\xa9C!WmW\xc8\u0174\xf7M\x11p\x1dc\x12T\xbd\xe6\xc
1\xb4\x1a\x03\x1f\x82\u04c7\xbbt^\x13\x9f\xf63(\xcb\xd0\x1e\x86f\x01\u0414r\xb0KS:\xb8B\xf9\xf
7A\f\xfd\xe1H

\xdeY\xd2f\r\x81\xc6\xc6\x06\u0114JnZ\xa1\xeb6\xc9\xf2\xe0\x98\x182t\u007f\xb1\x88\x06\x88\xc5\xd4
\x96\xf8\x05\x06+g\xb6\xe5^\"\u007fG\x89L\x94@\x10\x95\xa5fT\xa3\x8a0\re\x8b\r\xdb\xf6`|\xe3\xb6\xf8\xb8\xe13\xc3^\xb0\xa4\x94\xa4\xb3g\u6c7c\xbcr\x033_\x06\x00\xdf\xfd\xee_\xd3z1\u007f\x81\xe3\x87?|0\x14\b\xbe\xee\xba\xd7|a\xeb\xd6-
\x18\x1f\x1f\x8f*\xca#G\x8e\xe0\x1f\xfc\x83_\u00a7>\xf5\x87\xe8v\x97\x90e-
\xb4Z\x9d\x18\x04q\xe1\x15v\x8b\xf1\xf1qt:\x9dX\xf8\xc2d}0\x18\xf0\xd3O?\r\x00\x98\x99\xd9x<\xcb\x14\x88D\x04\x00\xfa\xfd\x01B\xca\u0485^\u03356\x98\x9e\x9e\u008e\x1d\u06e1\x94\xac\x158J'\x90\xe1\xfaU\xbe\xb3\r\xdc\xe0\xe0\u06e1\b$\x05\xa8!
&%DGB\x06\xebT$\xd71%\xfe\xd9k\x18N\xd5\xca\xf6\xdaNX\xfe\xd4\xea\u014a\xc6%\xe4\xc6\f\xaa#\x915\xc8w\xe9\xc2y\xba4\x84S\xfa\f\x19N1\rei&6\x04\xaa)\\1\x1f\x97\xa0)\xd7\x01\xaf*\xb2k\u060c\xa7\v\x16\x0fc\x12\xabj4\x0f\xc5\xd6\xd5\x1f\"\xb8\x97\x18\xbf\x1b\x10\u00a9AE\xa8\x04\x13\x12\u0618\x01c\x12\xdc p\x96\xfa\xbe\xfb\x00\xee\xa4\v\x0f\xef\xa7\x11~\xe71\xad
7fn\xae\xc0\x95\xe2U\xf8B\x1es\xb8\x13\u03da:=\xd1\xf90Y\xad\u0757q\x9duE3\xb4n8\x1a\xa0\x13l\x10-
\x01\xf2\xce\u0281\xcdR\x83\xd4D\xb2c\x1b2\xf6\xb4VC\x8dM`\xf6\x92WB5Z\u043a\x88!8l\x8dc^\x81D\xb7\xd7\u0179s\xf3;\xef\xb8\xe3\xcf\xf6:\x94\xe0\xfe\xf5b\xfeB\xc7+_ym\xfc\xfe\xf6\xdb?\xf8\xf9W\xbc\xe2\x15\agg7\xa1\xd1hF\xae\xf6\x993g\xf0\u044f\xfe\x1a~\xf1\x17\xff!\xbe\xf8\xc5/\xe0\x89'\x1eE\x96)\x8c\x8dMDY\xfc\x85R\u052d\xb5\xe8t:\x18\x1bk'\u015c\u0207m\xa8^\xaf\xdf\x04\x80\xa3G\x8f\xdd5\x18\xe45\nc\xc5f\xb9\xf0\x8b\xb91\x1aY\xd6\u009e={0>1Q\xcd4\x88
\xa4\x84\x94\n\x82\x04Z\x9di\xa8\xac\xe9\xe9f\t\xea\xc1\\\xcd\xc0\x04\x1cF>\xae\\\\\x183\x94/:af\x99:\xf5\xb9\xf7\x9bF\x16?pR\xf2\u05fc\xfc\x86\x13t<&;\xe6\n\xba\xecH\x90r\x85/\x13\u0085\x10\xd7d\xff<\x94\xe7@\xb5\u007f\u02cc\x905\x04dS8\x8c\xbc)j\xf8\u007f\x8c\x81[\u02dbf\xad8\xb6\x11~\xf0\xc3I?\xe9e`\xbdK.\xfb\xc7S\x92\xa0(q,\x0f\x1ds[\xc0N+7x\u039c\x1f\x8ejJ\xc8\xcc\r\x87S9\xbf\x15\xf0A\xda\x02j\\A\xcdd\xa0\x86g\xe6\xa4^\x91\x01M\xe2\xe1\xd79\xa5\x1b\xb9\u0398u\tk48\xc0*)~\x9e|\x05?\x16d\x00)\x00\x1eR1\x9ek\x1e`5\x1b=n\xea0\v\xb3Ei4\f\x03\x1bw\xecE\xa3=\u9b54)\x16sv\x9fcQ\x14\xa5VJM\x1d;v\xe2\x06\x00\xe8v\xbb\xeb\xc5\xfcG9\xee\xbc\xf3\xab\xa1H\x1c\xbd\xe9\xa6\x1b?q\xc5\x15W\x94\xe3\xe3\x13h\xb7\u06f5b\xf7\xc5/\xfe\t\xde\xff\xfe\x0f\xe0\xfd\xef\xff
~\xe37\xfe\t\xbe\xf4\xa5?\xc5\xf2\xf22Z\xad\x0eZ\xad\x8e\x17\x1e\x89\x1f\x9b\x822\xf8\xb0w:\xe3\x18\x1b\x1bK\xb0rAy\x9ec\xf3\xe6\xd9\xcbv\xee\xdcq\r\x00l\xd92WfY\x16\xa5\u0100\xa3b^\x88\x03P\"7h\x12B@J\t\xa5\x14\xb2\xccm\xabw\xed\u0685,s\x03j\xe1y\xbbB*\x88,\x83l4\xd1\xecL\x82\xbc\x02Tp\xe2Sb\x00.-
\xacq\x96\xa6<&\xc1\xde\xc3#4\x86\x888(\xa2\xfc\x9c_\x88\xe4\xf1\x02\x1d9^\xa0\xbeS\xdb\xcb\xd0;N\\$\xa4\xc3\xf5\x95$\x84H\u04c0\xa1\xa7\x8f\x19\xfe-
\x95\xa3\xfb\x89\x967\xbdZ\xcd@\xad\t\x84^\xe4;1\xa2\x93\xe7\xbammH\u0671\xce^W\x80\xbc\x8d\xc1h\xbe|@3\xa8%\x9c\xeb\u2e02jJdaA\x92\xc1m\xd1\xedDB\u0407h\t\xa8i\x05j\xc9\x1a\x1e\x9e\x12i\bnA\xaco\xd9\ua2f0\xd1\x06\xa6(\\Wn\x8d/\xa8\xa6\xf2_I\xbe\b\x0e#bb@q\xa2/p\x98y\xa4*\xc2\xc2x\x13.\xf6!7\x96\x03\xben\xc1l06\xb1\t\xd3s\xbb@B\x80\u0640\xad_\x04\xb8Zu\u03df?\x8f'\x9ex\x82\x00\xe0\xe0\xc1C\xeb\x98\xf9\x8fr\xec\u06f7\x0f\x9f\xfd\xec\u007f\"\x00\xf8\u065f}\xff\xff\xb8\xed\xb67\xff\xdb\u05fe\xf6\x06\x8c\x8d\x8d\xa1\xd9lal\xac\r\xa5T\xbc\xfdC\x0f=\x84?\xf8\x83?\u0107?\xfc\x9c\xd2\xd3\xda\x00\x00

\x00IDAT\xf7\xf0\xa1\x0f}\x18\x9f\xfc\xe4\x1f\xe0\u0211g\xa1T\x13Y\xd6B\xa3\xd1F\xb3\xd9F\xb3
\xe9\xba\xfb\xe0\x1f\xfe?\xa7\xc83\xa4\x14\xd1\xf90\xc8\xf6\x8d1\xba\xd9l\xa2,\xcd\x0e\x00\xb8\x
f6\xdak\xdb\xe3\xe3\x1d\xdf\xd1V\x17\xf6\u0673g]Q\xf81\xd11]\xe1\x16PJ\xa1\xd9l\xa2\xd9l\xa3\
xd1h#\xcb\xda\u0232\x16\x94jB\xca\f\xd62~\xf8\xd0\x0f\xf0G\xff\xe9\xb38\xe3\xe7\x00
\x82\x94\n$\\zK\xa3\xd3A{n\x1b\xa8\xd5\xf2\x83\xac\n\x12\x11p\x81\u077a\xb0\xc8\x19\x184\x05\
n\xf6A\f\xd1\x1a\xb7^\x1c\xd2\xee\x94\xd2\xeb<\xc1Vb\xafK/\xae\xa6\a\x9a!Y\xb8l\u034d\nb\xdc\
xcb\xe0}Z\x8eLW\x19\u0090\x01\x95\xc7\xc9[\x12\xa2)@-
\t\x1a\x97\x80\xac\x9a\xd1:\uf15egQ\xa1\x17t\xb2\x19B\xb5\x93\xe2\xcc0\xbe\x0eJr\xc3Lz\x9e\u0
5c2*<\xc6\x19\x92M+g5\x9c9\x93.\x12\x95\n4\u0707\x90@c\\B\x8cK\x17\xe3\x96t\xe5<4\xba\x88
\"\x8fp\x8ed\x86-K\x18]\xbaa\xa7\xa9\xe0\x15\xd808\xf5
\u007f\xf2of\xeb\x87\xe7\x0e\xce\t[>\x1b\x86\xa1\\\xd9\xf4R\x98eQ\xb2\xb33\x1a\xaa\xd9\xc6\xc6
\xdd\xfb\x00!\xa0K\rcu\x14+1;\x87\u04d5\x95\x15<\xf7\xdca\xed\n\xfb\xe2\x05Y\xcc\u0545vB\xbb
w\xef\x01\x00~\xcdk\xae\xa5\xfd\xfb\x1f\xe4\u007f\xf4\x8f~\xfd\xb7\xdb\xed\xf65\x00\xdes\xef\xbd
\xf7\xa1\xdb]\x81\x94\x92\xad\xb5d\x8c\x89)\u073d^\x1fw\xde\xf95\xdcu\xd7\xd7\xf1\xd9\xcf\xfe1\
xae\xbd\xf6Z\xdc|\xf3M\xb8\xfe\xfa\xeb\xb0s\xe7E\xe8t\xc6
D\xe6\x8bc\xfa\x81\u05c9\xda\xf2\xef\xbc\x1c&\xd3\xfb\x8a\xb2'\x84\x10EQ\xa0\xd7\xebZ\x00\u0
631c\xfb\x9f\x13\xd1B\x96e\x1bz\xbd\xea\x12x\xf2\u0267P\x96\x03dY\xf6\xa2\xf3N]\xe7,\x9c\x87\
xf8\xdf\u025am\xb0\xb4t\x1e\u01cf\x9f\xc0\x91#Gp\xfc\xf8\t\x9c<y\x12\x87\x0f\x1f\u0191#G\xf0\u
88cf\xe1\xe0\xc1\x83\xf1y\x86\xe1
3C\x00h\x8dOaj\xcf\x1e`C\a\xf6\xe42\x94\x97\u07a7\u0233e@\x17\x8c\xa2\xeb\xba')\bMIh\n\x8a\
xc5
\xad\xa1i?Z\xeb.\x87\xa0\x98\x17W\xc69v9!\x10\x813\xc7\xc9f\x00\xb4\xc8qg\xc2\xe4Ph\x81!;X\x
e9\xb0uj\xf8Y\xc0\xb8\xc7\xc9m\x15FM\xab\x1e\x92\xd6\x00\u007f\u0590\xbf\xbf\xc0\xf9[\xb8\xb
aG\xfe\xf3N\xbc\x06\xda\x1e\u051b\xecY
q\x81d\x88&\x81\xa7\x14\xd8j\xf0\xb2\xae\xf8\xf0\x1c:AFF\x02\u0658\x9bs\x00\x80\xb1\xec\xe8\x
82\xc3\n\u0580\x97\x87\x957\xe5\xa32`u\tS\x16\x11+\x8f\xa2!O)D`\xb2\xf8\x02\x1d\xbd\u030d\x0
5$\x83U5r\xae\xa2\xf5\x1c\xb6DI\xf3Vy\x88\x91_|4\xa81\x86\u03b6]h\x8cM\xa2\xb7p\x06\u031e\
xb1B\x95\xe2\xb8(J,..\xb2\xbb\xb6.L\xc3$\x85\v\xf4\u063f\xffA~\xfd\xebo\x15D\xd4\x03\xf0S\xbf\xf
f\xfb\xbf\xf7\xb1F\xa3\xf1\xaf\x1f~\xf8\x91\xe6\xe9\u04e7\xa9\xd7\xeb!\xcb2ffbfdY\x06k-
\xf2<\xc7\xfe\xfd\x0f`\xff\xfe\a\xf0\xb9\xcf}\x1e\xbbw_\x8c]\xbbvc\xf7\xee\xdd\u0633g\x0fv\xec\u06
0e\xad[\xb7`nn\vfg7a||<\xaa\xba\xcar\xf0w^\xd4\xd7\xc0\xf0\xd9\xc3\x14\x12\x00\xf6\xec\xb9\xe2\
xc9-
[\xb6\u0627\x9f>\x00cl\xfc\xa4\xff\xf0\x87?\xc4\xe2\xe2\"6m\x9a\x83R\xcaE\\\xbd\xd8\x12l\n\x94e\
x89\xb2,\xdd\x16\xd3\x1b\xf5\xaf.\xfeT\xf3\x86)\xcb\x12\xcb\xcb+8y\xf2$\x9e~\xfai<\xfc\xf0#8x\xf0
\x10N\x9e<\x89\xf9\xf9\xb38wn\x01\x8b\x8b\x8b\xab\xa0 !\x84\x8bw\x13TQ-
\x85\xc0\xc4\xec\x16l\u06b5\v\xa2\u04c0!\x86\xb4\x94^\xe3\xae\fy\x86\a\x9d\xd70Z`0&\xd1\x17\x
04E@K\xba`\t\xe1;+Z+K\"tb\xb6\x1a8\xf3*\xb1\xca\xf3\x15B\x1ajS]e\xe4\xcc\a>\x18G\x97C\x11\f\
xa1\xbc[\x9f\u007f|+\xe0\x84Ha\xa0;&\u0717/\xb0\xc4\u0568r5.\x9e&\x85R\xa2~J\xb924\xf2\x8c\x
e3z\xc0\xe4!\x06\x1f\xecL\xf0\xd1s<\xa2\x9b\xf7\xc3\xc1\x04\xbe\xaa\xa0\x1f\xbf\xc06%\u0318\x
85Y\x04\xac\xa9\xcf\x17\x049\aFa\xbc{\xa1?5c\x1dN.\xd3A\xb1\x1fZ\x83\xc3{R\xedI\xac\xd10y\x0
eS\x140\xba\x04[\xedZ\xfc\xe4\u06b1\xfe\x04+\xd3,[\xb9*\x02n\xb1\x94\x00\xca\xc4x+\xce\u0753
E\x88\x03'\xde\xc2\xc2\xe1\xfd,,\x1a\x13\x1b1\xb1e\x17z\vagY\xf1\u0549\x88\x8a\"\u01d6-

\x9bw\x00\xc0\x9f\xfe\xe9\xff\xd0G\x8e<K;w\xee\xe2\xf5b\xfe#\x1e\xdf\xfa\xd6=\xd6\xf3\x12\xf9\x
d7\u007f\xfd7\u007f\xf7\x81\a\xee\xfb\x9b/\u007f\xf9+\xff\xfa\xde{\xef\xbb\xee\u0631c\xe2\xf8\xf1\
x13T\x14y\xd1l63f&c\f\x84
\x94\xa5\xeb\xb6WVV\xf0\xf0\u00cf\xe2\xe1\x87\x1f\x05\x00t:\x1dLOOabb\x12\x1b6L\xe3\xe2\x8
b/\xc6\xf5\xd7_\x87\x9bo\xbe\x19W^\xb9\x17\x13\x13\xd3\x00\x18y\u078f\x9dp(p/\x16\x96\xf1\nV\
f\x06\x03\xf4z\xdd\xda}\x01\x84F\xa3\x11Y.\x00\xe4\xf6\xed\xdbOH)f8\x99\xe6\xddw\xdf\xfdx\xee\
xb9\xe7\xb0i\xd3\x1c\xa4l`ii\x01\xddn\x17\x83A\x0e\xad\u02f8\x05t\xb8u\x16\xe1\xa3\xc1
\u01c9\x13'p\xe0\xc0\x01\x1c8\xf0\f\x8e\x1e=\x8a\x85\x85\x85\xa8*]\xdd\xe5{S$\xff\x1c\x8d1\x18\
f\x06X^^\xc1\xd2\xd2\"\xce/.aqq\x11\xf6yv\aB\b\b\x95\x81H\xc0Z\xe3!\x1a\u9f22\x1bMl\xbb\xea5\
x98\xd9~\t\xca~\x0e\x16\x04\xab\x00a\xaa\xe2\x14\xa5\xf8\f\xa8\u0082\x96\x19\xc2\x00yG`@\x8
4B3\xfa\x9a\xd0\xc9\x18-
%\u0410\x14\x87\xa2qx\xe7\xb1\xe1\xc0N\vN\xb9\x15\xe6\x1e\xba\xb3Q\x85\x8dc\xbdI\xfd\n\xe3\
u06d1\xaaE\x1d\r\xde\x15\xeb\u042d\x1a\xc0\x04\xb5a(\xd8M\x01LHG\xbf\xe4Qv\xb5\xb4z\x96IC
\xbf\x1b\x81\x8e\xd3Z#\xd2d\x9e\x10\xa4\ubafb\xfb\xa1\xbf\x8c\xb5\xaf\xb2\x8cu\xfe(\x8c\x82\x19
yia\xfa\x16\x82\x192&bq\xb5\xc6\b\x00}\x03,k`:s\x96\a\x89\x86I\x10\x0f\xa9Y\xa9\xfe8\xc6@\xe7
9L\x91\u00d6E\x84W\x90\xc8\xf9\u00d0\u0630\xf5E\xbe\x1ab\x82\xfd\xfb.\tF\x02\x9a-
`\xac\xb7\xbd%\b\xff\xees|lO;\x14>\aV\t\b\xa3\x91\xb5'1\xb5\xfd2\x9c~r\xbf\xff<+\b\xa1\xc2g\x99\x
8a\xa2\xc0\xec\xec\xecnf\xdeBD'\xf1\u0084\xd2\xf5b>\x8c\xd9\x02\xe0\xdf\xfd\xdd\u07e1\x8f}\xec
\xe3\xfc\xeaW\xdf\xf0\x97\xcc\xfc\xe6\xff\xfe\xdf\xff\xeb\xffv\xcf=\xf7\xfc\xd2\xe1\xc3G\xdfx\xfc\xf8\
xf1\u01a1C\x87\xc0l\xb5\x94JJ)I)\aKh]\x87P\xba\xdd.\xba\xdd.\x80\xe3\x00\x80\a\x1f\xfc\x01\xbe\
xf1\x8d\xbf\xc0\xb6m[\xf1\xa67\xbd\to{\xdb[q\xcd5\xd7`\u06f6-
\x1a\x8b\xf0B(\x80U\xd7N\x95q\u03e8\x8e\x87\x19\x8dF\x03R6\xb0\u007f\xffw\xf1\xc8#\x8f\u059
eO\xa3\u0450y\x9ek!\xe8\xa0\xffy\xf1\xab\xbf\xfa+w\x18c\xaf\x96R\xc6b;??\x8f/}\xe9\u03d0\xe7\x
05\x1ez\xe8!\xfc\xe0\a?\xc0\xc1\x83\x87p\xee\xdc9\xb8\xb0k\xa0\xd9la|\xbc\x83\x89\x89\t\xb4Zm
0[\xcc\xcf\xcf\xe3\u0109\x93\xe8vW\x90\xe79\xca2\x9c\xbf\x19\xda)P\xf4\x1c\x0f\u07ff\x18\x8a'\t\
x01!\xa4S\xcdI\x05!\x1c~EF;xE)\x90\x14\x98\xb9x\x0f\xf6\xdc\xfcV\xa8\xc64\xfa'\xcfC\x90\x80\x95
\x1e\u03b0U:\x1bK'\x19g\x02d\u0260\x15\x03bF1.a\x05\xa10\x8c\xd202a\xd1\xca\x04Z\x8a\u041
0\"F@\xba\xady(h\xec|\xae\x87\xdc7WA\x15\x9e\xed`9\x91\xfb\xd7\xec\u0513n\xd8\xd3\xe10\xa9
\\\xa1X6\xe0\x81\x01t\x05MXCN\x8cSX\xb0P0\xd2CM\xab\xb2,|a\f\x16\x05\xb5\x1a\xef\u07cb\b\x
f7\xa4[\xfe\xba\\?-\xd6a\x14\x11\xb1\xf1\xe1\xa0\xe9\xb0\xc8P\xeaT\xb8\x8aM\tm\x19}m\xd1-
\x19\xb6k\xd0X1N\xdf\x14\x16\xaf`\xe3\fr\xe7h\x18\xb4l\xdc\xef'$\x023\xd1V(\x8e\xa7L\xa6\xe9\x
ae\x04k\xdc\xc0\xd3\x149LY:Z\xa0\xc7\u016d\xad[\xe0Zk1l\x04C\x9c:(8\x89\xbf{\xc1\xb8\xc2\xe6\
xfd\xfb\x1b\x94\xa3a\x12\xab%\xa0\xa5\xff\xbb\xa2\x84\x18\xeb`|\xd36d\xcd6\xcaA\x1fBH\x90l\x0
0\xa6\x04\xc3\xc0\x18\v\xa5\xb26\x80\v6\x13\xf4\x82.\xe6\xe1\xf8\xd8\xc7>\u039f\xf9\u0327\xe9\
x97~\xe9#LD]\x00\x9f\a\xf0\xf9;\xee\xf8\xca\a\xbe\xfd\xed{\xfe\u0791#G\xde\xf6\xec\xb3\u03e9g\x
9f}\x16\x83\xc1\x00D\x8c,SPJ\xc2\x18\x05c\u071bQ/\xc8\xce3<\xcfs\x9c?\xbf\x80\x83\a\x0f\xe1+
\xf9*^\xf3\x9aW\xe3u\xaf\xbb\x15\x97~9\xe6\xe6\xe60;;\x8b\r\x1b\xa6166\xf1\xa2\xceY\xeb\x1c
_\xff\xfa\xd7\xf0\u06ff\xfdoq\xe0\xc03\xd1\x1e\x16\x80m6\x9b\xb2,\xcb\x03\x9b6\xcd>\x19n\xff\x8
aW\xbc\xe2\xf4]w}}U!\xfd\u0527>\x8d?\xfc\xc3Obqq\xf1y\x16=8\xfa\x9f\u007f\x8c\x88iz\xe8$\xfc?\
fR]\xb3\x13\xf2U\xd9\xc3.\xbc\x9a\xa7;bq%\xe1L\xfe\x11\xd9B\"^_\u0194\xf1\x82\x93J\xa23\xb5\x
01\x9b\xf7\\\x8d\xab\xdf\xf5al\xdc{#V\xe6{\xe0\u0702\xd8\xfb\x1bJ\xc0\x12\x83\x02\f`\xad\xc3.|\xb
c\xa7`\x8b\u018a\x1bD\xe6\xe3\xd2\xd9\xd22PX\xa0\xc8-

\xba%\u0412\x16c\x99@S\x0e\x85\v3\xc5b\u00a3\xe4\xf7\xb4\x1ax\xb1pp\x81\xa5\x91\xe4\x16\xf7\xba\x19\xef\t3.\xdd\xcfK\xeb\x8a9\x85N\u0531G4\xbb\x8eU-
\v\xf0\x84t\v\x95M\x89\x1c<Zk\xef!\b\xe3\u03ff\x1a\xfer\xb4\xc7\x1dAV\x84\x89\xefy\x1d!\xaaM\x15(\x88r\xe0\xc56T\xf3\u0231\f\xe4\xa5EO[\f,\x83KF\xabo\xa0\xb4\xdf}\b\xb8\xc5\xcc\xc3(L\x0e\xd6p;N\v\xb1\xa8\xbd\xe0K\xd6c,\x98\xeb\x93W\"\a\xad\fr\xe82\x87.\x8a\x04/\xd7Ug\xee\xa5\xfcQ4\xc4\x0e\"\xb4!\t\xc9\xc7\u00b1\x0f\xe1h\xf8HA\xee\x01\xa6tx9{\\\x85\x02F.\\\xeek)\xdd\xeb+\x8c\xeb\xd2\xd9X4:\x1b\u041a\u0704\xa2\xff\x1c\x84\u02a0l\x03F\xf7aJf)%\xba\xdd\xee<\x80\xf3\xe9\xc7h\xbd\x98\xff-
\x8e\xb7\xbf\xfd\x1d\xfc\xe9O\u007f\x12\x1f\xf9\xc8G\xe3\xcf\xde\xfd\xee\xf7|\x1e\xc0\xe7\xbf\xf9\xcdo|\xf0/\xfe\xe2\x9b7\x9f8q\xe2\x8d\xe7\u03df\xbf\xf2\xc0\x81g0??\x8f\xa2(
\xa5\x8d9\xa2\xcc6v\xec\xc6T\u0662\xc6Xt\xbb]\xf4z=\x1c9r\x04w\xde\xf95LOOc\u01ce\xed\u0631c'\xb6n\u0742\xed\u06f7ann\x0e\x9b7o\xc6\xc4\xc48\x84\x90qq`f\x14E\x8e\xc1\xc0-
\x0e\xa7N\x9d\xc2C\x0f=\x8c\xbb\xee\xfa:N\x9et;2\xa5TP\xac\xb2\xb7\x8c]y\xc3\x1b\u07b4\x10\x9eK\xa3\x91}gnn\x0e'N\x9cDQ\x14\xb1\xa8.,,<\u07ee\xc5\xc1\x1b\x9e&XAB\xd5\xef+\xc3)\x8f\x8bZ[\xfbZ\xabx\a\x8f\v\x12\x94\xd8\x0f\xb3\xef\u0185\xc3\x14\x85\x80j4\xd1\x1c\x1bGgf\x16\x13\xb3\xdb\xd0\xd90\x8b\xe6\xf8$Tg\x12\xed\xd9\x1d\u0630\xfb\x1aL\\\xb4\x0fK=@,\xf6\x91\x19\xac\x9a\x00\xd6\xc9\r\xec\xf2%\xfdV\x98\f\x90u\r`\x81bR\xc24\x84\xebf\xddn\x1a]\xcb(\xacu\x98\xba\xb7\x96M\x85!\xa1C\x14\xabL\xb9\xea]+E\u067a+\xa4\\\xf3#\xa9\xa2\xc8\x02D\xcb\xc2]=\xc1\xc6\xc3Z\x8e\x1eV,\x1dvn\xfa\x16\xda\x14\x10ZAN(\u020c|\xe3H\xf1\xf1\xac\xc7E\xc2p\x17\xfe\xb1-
\x1c\x13\x05a\xf8\xeb_2\x9bb\xdbT\x15a\xf6\x03\u3c01\xa0!*|x\xba6\xb2\x88\xc2\xca\xe0\"\xd2\x06\x861(\x19\xb9\xb1n!\"
\xcb\x19\xb2`P(\x98\x02\xb0YP\n\xf9\x9d\x88f\xcf<\x12\xc8\x18PK\xae\xa0sS\xa4\xebfmg\xc2ZC\xe7\x03\xe8|\xe0\x18,\xa13\u05fe\x90'\x9f\u05fa\xd9V\xf5\uf2ad\xe3v7\xc2\u007f^\xdc{\"
\xc8\u0090\u007f\rQ\t\xbb\xb4\x00J\xc1q\xe1v\x10\x8e\x00\x1b\v\u065a@kj\x13\x96N\x1c\x82
\t)3\xf7;\x06g\x8d\x06\x0e\x1e:\xf4\x04\x11-
\x02\x10\x8dF\u04ee\x17\xf3\xbf\xe5\xb1c\xc7E\xab~\xf6\xf5\xaf\u007f\x8d\xde\xf6\xb6w\xf0\x9b\xdf\xfc\x93\x9f\x03\xf0\xb9S\xa7\x8e]\xf9'\u007f\xf2\xa5W\xed\xdd{\xc5\rg\xcf\xce_\xdb\xeb\xf5\xae\x9f\x9f\x9fo\x9e:u\n\xddn\xcf\x0f\x01\x1d\xf4R\x14\x05\xfa\xfd\x01\xf2|\x00c\x92\x0f\x871\xe8\xf7\xfb\xe8\xf7\xfb8q\xe2\x04\xbe\xff\xfd\xfd\xa8\nn\x03SSS\x9e\x19#\xfc\x82`\xc0\xec\x06\x86E\x91#\xcf\v\xf4z\xbd\xf87\x81\x8b\xad\x94\fjI\xbe\xe4\x92K\xb0o\u07d5O\x11\xd1\xe1p\xbbV\xab\xfd\xec\x8e\x1d;\x8e>\xf9\xe4S;F\xe3\xda\xf5\">\xfc}\x1a1\x97B)\xe1{_\xb8\xf9\x85HzD\x04\xa9T2\xfd'\x80\x84\x13\xffd\x19\x1a\xad1\x8co\u068a\x99]\x97c\u64bd\x98\u06be\a\x9d\x99Y\xb4'\xa7\u045a\xd8\x0096\x01-
[0B\xc1\x88\x06\x8a\xdcb\xb9?@\xd6+\xd1.Qc2\x90\xf5A\x04\xc1\xf1\xce\xc0o\xe3\xb9R\x1b\u00b5\xca\xd9\xc0\x80\xc0\xc8'\x15LC\x80\xa9\u2a57\x86Q\x96\x8c\xbee\x8cenP\x9aIJ
\x16\xdfuS:(\xad<\xae\xeb\x905\xd5\xcc\x0fC\x17\xccL\xae`\xfb\xdd>k\x06V\f\x88\xbdp\u01ba\xc1\x9f\xf10\x06\xc1y\x8c\xe8\x9c\xc1\xe7J\xa0\xb0\xa0I\x05\xd5\x10\x88\"R\xaeZ\u22bf\xed\x9e/\x0f\x85\x98\x86\x1al\x93?c[)+\x89\u073f\x1dN\\\x87t\x1c\xb3eH?J\xee9\x15\x86\xd1--
\xfa\xda\xc2\xf8\xa1)\t@\u632co\xe2L
\x0e\xab\xc3\xceE\x00d\u072e\x89\x19\xd0\xd6u\u045a\x19\x12\x80\x9c\xc9
2\x01k\xbdM\x82p\x85\u0616\x1a\xba\xc8aB!\x8ftD\xed8\xde\xc1{\x85\x19ll\f\xa7\b\u0316`o\xcb\x1e\x9b\x93\x82<\x1f\xdci\x15\xb8t\xb7\xa3j\xb5\x84\xa7\xa0\xc0\x92t\x1d\xb9\xa8\x16n\xf2\u05c7

1\x06\xb2\xd9B\u0599\x8a\xf7\x97\xc2TB\b\xf4\a\xc5\x00\x00^s\xfd\r47\xb7e\xbd3\xff\xbb:\xee\xb8\xe3+x\xdb\xdb\xde\xc1\x0f=\xf47\xf4\xe0\x83\x0f\xca_\xf8\x85_4ss\xdb\x1f\a\xf08\x80\xcf3\xf3\xb6\xff\xf6\xdf\xfe\xcb\xdc\xe1\u00c7ggf6\xbe\xe9\u0739\xf3;\x1ey\xe4\xe1%)\xd5E\x1b6lx\xeb\xc1\x83\a\xd5\u0463G\xb1\xbc\xbc\x82\x95\x95\x15,//#\xcfs\x0e\n\xcd\xe1\x8e\xd5u\xdf\x05\u039c9\x833g^\x98\x16\x18:q)E\x847\xac\xb5\xbcy\xf3\\v\xf9\xe5{\x06\xb7\xddv\xdb\u007f\x01\x80\xc7\x1e{\x98\xf6\xed\xbb\x86?\xf8\xc1\x9f_z\xf2\u0267\xfe\xea\u0211\xa3\x1fx\u4447\xd1\xeb\xf5G\x16\xde\xf4\xbc\xc2\xf7C\x85\u007f\xe4\xdf\x11\x11Z\xad\x16\t!\xe2\x1c!\xfd\xbb\xb4\xf8K\x95\xa13\xbd\x01B5AY\x86\xf6\xe4\x06\xcc\\t\x19\xb6\\~\r6\xee\u078b\xf6\u62d0M\u03601>\r\xb4\xc6\xdd6\u057a\xf3\xe8\x97\x1a\xa56\xb0\u0682M\x0f\xb64\x10\xc6uy\xc2$\x92\xf9\x9a\xf2\x93]\xa7g\x03\x8c\xe0)e^\x90\x130c\u0577\x00k\x14\x13\n\xa6\xe9\v\x9cE\xbc\xb8\xf3\u04a2,\x81AF\x18oJ\xb4\x1aU\xa7\xce5x\x82V\xe1\xc4\xe9\x8bfb\x91\xe4\x88)k\xc3\u0425+.\xc62d\xcfB,\x1b\x90u\x85-\x868\b\xef\u07c2D\xe0h\x00\xbbl\u07023\xa9@\r\x82\x02\x90\x81\x90\x81#\a\x9c\xfd\xf3\x14*a\xa2\xa0\xdaYp\xbaCH\xb6\xfa\x82\xea\u04c0\u06ae$\xb0K\xb8>-
\xd0\x06\xe8Y\x87\x8b\x97&Q\u0252\xdf\r\xf5-
T\xe1vI,\xfc\xfba\x00\xd2\xd6\xe3\xfc~\x87\xe1'\xa8\x82\x00\xd6@I\x8cr\xd9@\nB\xb61\x83\xf23\x83R\x1bp\xe9\xf0q.\x1dk\xc5j\r\xa35\xd8w\xe4)\xac\xc2\u0182\x8d\x89\x83P\xa7\b\xe5\xa4kO\xb6\x1b\xecp&1\xb0\xb0%;s-\xef\xfbB\x11\xa7r,\x1f\xaey\xe1x(\xce/
\xa42H\xcflcx_u6\x00;\xe7\x97M\xb3\x9b'\x01`\xff\xfd\xf7\x99\x93g\x17\xd6\a\xa0\u007fW\u01fb\xdf\xfd\x1e\x8f5\xbf\x8a\x01\xe8\x13'\x8e\u04b7\xbf}\x8f\xba\u77bf\x12\x9f\xfc\xe4\xa7\n\":\x1e&\x9d\xcc\xfc\r\x00-
\x00\xdaZ\u04fe\xf7\xde{7\x1e8p\xa0\xdd\xedv_\u007f\xe4\u0211\xf7?\xf6\xd8c?q\xf8\xf0\xe1\xa9n\xb7K\v\v\xe71\x188_\x14\xb7]\x17\x15\xc5\xcdo\xfd\xa8>?\x1b\x8aJ\xab`\x8e\xa4Sf\"A[\xb7n\xa3\x9bo\xbe\t\xb7\xdcr\u02ff\u0677\xef\xea\xbb\x01`\u07fekB\x90s~\xd7]_\xfb\xbd,\xcb\xde=>>>\xfe\xc0\x03\xfbiqq\xe9Gz-
\xaa\xee_\xa1\xd9l\x12\x11\xa1\xdf\xef#\xcb\x14\x8f\x8d\x8dQ\xbb=\x06\",l\u07fe\xddn\u06f6m\xfe\xfc\xf9\xf3_\x1b\f\xfao\xfc\xde\xf7\xee{e\xea\xff\x12\x1c\xe1\xc6&7\xe2\u019f\xfbUl\xdc\xfbJ\xa09\x86\xf1\x99-
hM\xcd@\xb6;0\xaa\x05C\x02eYb\xa5,\xa1\xcf/:\xb8*\xf0\x95\xa9JI'\x02\x98\x04Di\xdc@3TTI\xd5\xf7\x9c\x14\xf6\x00lp\xc5\xf1vX\xad\xc3iI\x00j`AF\xa3\x98\x94\xd0-
\xe1=^b\x1d\x85\xb5\x8cA\xc1(5\xa3m\x04&\xda@K\xc8\xc4\xe3|m<\xbc\xb4\fm\x19\x03mQ\x9a\nk\xb6\xceb\x1b:wb\x16\xa9\x19\x8d\x15\vU\xb2\x87\xa1\x82\xa0\x06\x95\xe8\xa66\xcfd\bKP\x03\xbf\x18\x8dK\x14\x19\x01\xdaB\x18\x8bL\xb8\x9d\x84\x92\x04\xa5\b\x92\x9c\u06b2\xdaI\x84\xe2\x9d\xf8\x8cp`\xab\x84q\"\xc7\xcf)\xaa\x97..H\xe9B\x95kF\xcfX\xe4NG\x19\xcd\u0442Y\x95\xea[\u0201\x89wB\f@3\xa4\xf1,\x16\x03\b\xe6\x88\xd4
@\x16\x9eQd4C\x9f/Q\x12\xa31)\xa1\xd8\xc0\x16\x8e\xb1\xc2Z\x03F;\uee9fc\xc1\x98\xaa\x90\a>\xb9\x87A\xd9$\x1cs\xb6\x91c\x8e\x80\x99\xb3\x05\x97\f;0@a\"}2:4\"\xb11`@X\x86%\xaa\xbbf\x06\xdez\x80\x1d\xbdk\xa21\xda)G\x01\xc7\xcc\xf2\xa1\xcf\xd5\u02b7\u0799\xff\xffrl\u077a\x83\x01\xd4\xd2\x1c\x9e~\xfa\t\\v\xd9^\x90\xe3\xa2\x05\v\xc2\x12@\xa8\x92\x8f\x03\xf8\xf4\xf1\xe3G\xf6\xfd\xf1\x1f\xff\xe7\x9f9|\xf8\xf0\xcf=\xfd\xf4\x81\xd9~\xbf\xbf\x19\x80XZZ\u00993g\x90\xe79\x8a\xa2,=.-
\x84\x10\x94b\xd1\u00d0\a\xbb\xc3Z\u02d4e\x99\u06b0a\x9a\xb6n\u074ak\xae\xb9\xfa\xec{\xdf\xfb\xde\xdf{\xeb[\xdf\xfe\u007f\x00\xc0\x1f\xfc\xc1\xef\u04ef\xfd\u06af\xf3\x17\xbe\xf09\xba\xfd\xf6\x

0f\xf2\xdb\xdf\xfe\x8e\xfb\xfa\xfd\xe57\\t\xd1\xce\xffs\u01ce\xed\xb7<\xf1\u0113\x8dS\xa7Na0\x18
@)\x85\xb2,K\xa5\xd4\xf2\xe4\xe4\xa4PJ\"\xcb\x1a\x12@\xbe\xb0p\xee(3z[\xb6\u0309]\xbbv5\xa6
\xa6\xa6N?\xf5\xd4S\xff\u03d5W\xee\xed\xddt\xd3M\xadK/\xbd\xd4LMM|\xf3\xaa\xab^\xf9\x8c?\xb
f\xe6\xdf\xff\xfb\x1f\xfe\xcf\xdf\xf9\xcew_\x19,k\x03\x9c\x026
\xa9\xb0s\u07ed\xd8y\u02ed\xe8\x19\x86\x06A\x1bF^\x14\x18\xf4s\xb05\xb1\xd0H)!\xa5\x88\x1d\
xec\xaay\x1e[d\x85u\x83&O\x13$\x9bx\v&xFJ\x19\x8c\x9e\xdb)\x89#p\xa7K\x8b\xe6\x12\x83\x8c
B\xd9\x11\xb1[Mi\x88\xdaX\xac\f\x18%\bc\xd6A/JT\x9d\xab\xf5x\xb5\xb5\f\xc3@a\x18\xb9\x1f\x06
\u06e4\xcb\r\x1d\xb01\xae\x90\x90\x01T\x9f]!\x0f<f\x06\x84\xae\xd8\x16\xc2T\u0404\x15\x04\xab\
x1ch,\r@\x03G\xf3\xcb;\x12\x85
h\x03\x94\xdaV\x83l%\x905\x81\x86r\xe7\x9by\xb5i\xe0K\x8b\xb0\xb3!D\x8a
\xa3>#\xa9\xcdU\xfdBg\xd8-
p\xfd\u00a2[X\x18\x01\x88L\u01a4\x9ep{\x953\x1a=S=\a\x06\x84f\x90\xf6\xbb(\xeb0\xf4\xf0^\xa4\
xf3\x06\xb7\xbb\xb2\xb1\xf8\x17g\v\xf4\a\x80j\x1ad\xa6pE\xdcxz\xa1\xadb\xe1\xc2\xc03\xf8\x98#t
\xe2\xa6\xe2\x92G\v[\xe3\xfd\xcd\x03,ZZ\x98\xae\x86\xcd\xdd\xcf\xe3{b\xc9\u0767_\x04c\xb0\x89
\x00l\x18\xe4\u05f6#\xc2\x0f_\r\xa4Tn\xbeV\x16a\x17\xc0\xadV\v\xcdv;\x0f\x9bo\xb7\x8f\/\xe6\xff\
u04ce\xcb.\xdb;\xf2\xe7\xcf<\xf34.\xbd\xf4\xb2\xf8\xefm\xdbv>\x06\xe01f\xfe\x97\xf7\xde\xfb\xdd\x
eb\x1ex`\xffM\v\v\xe7_\xf5\xec\xb3\xcfN//\xaf\xec[^^\xdeND\x9ds\xe7\xce\xe1\u0739\x05\xf4z\xbd\
x04/\xaf\x8a\xb9\x10\x8e1\xd2\xe9\x8ccjjRNMM\xa1\xd9l\xf6ggg\x0f]u\xd5U\xdf\xfc\xf0\x87\u007f\
xfe\xd3ss[\x1f\x03\x80\xf7\xbd\xef\xe7h\u04e6\x19\x06\x80\xdbo\xff
\u007f\xedkw\xd2;\xde\xf1Nn\xb7'\x1e`\xe6\u06ee\xbb\xee5\xb7\xdfu\xd7\xd7\xdf\xf5\xdd\xef~\xaf\
xd9\xedvinn\xae\xdbh4\xbf{\xf3\u036f=\xf2\xaew\xbd#k\xb7\u06f0\x96\x1b\x9dN\xe7<\x91\xba\x87
\x88\x06\xcf=\xf7\x1c\xee\xbb\xef\xfe\xf8\xbc\xbe\xf5\xado\xe3S\x9f\xfa\xbfG\x12\x8c\x17\x17\x97
D:+H\x9b\r]j\fV\xfa(\x16\x80>\xaf@+G\xb7\xd3\\\x9f]\x06\x88\xe4\xf9\xecE\x84f\u021c\xbd\x89Q`\
xa9$\x86F\x89\xc9\x16\xfb*\xcabd\xc4g\xac\xea\xc2\x12\xa8`4\xac\x06A\xa2\x1c\x93\x18\x8e\xbe
\x16\x9e\x9fg\xc0X),\x06\x06hJ\x82\x0f\f\x82\xb1\x0ek7\x89\xe4=\xb0A\x86\x1e\xce\r\xf0\xfc\u03f
2\x82\x91\xe5\xbe`\t\xf7\x1aH\xcd\x0e~\b,\x99\x94\xefg<\u03ac\xdc\x0e\x83\fC\r<\u04e2I(%\u057
0m\xb0\x13\xe7\x14\x96=\xd4\xe2DS\xd2\xf3\u01a5p\xcf\xc1Y\n\xa0bQ'\x9c\xf2\xb8T\xfa\x81mQ
Z\xf4s\x8b^\xcen\xe1 @HQ\xdfa\x12 \r\x90\xf5\rD\xe9\x19
\xcc\x0e\x177\u026e*\x95\xd8&\x19\x9b\x1c\x02
`\x00m\xc1\xc2\x02\xb9\x81\x19\x18\xf4\xdb\x06J\x19\xb4\xc8\xe1\xe9\b\xbe*1\x1e\x8e\xa3\xdd-
\x12qP\xc57\xf7\x81\x15\x89X\xc8\x18FQX\xe8\xae\x06\xe5\xa6fU\x1f\x98J\x14w\x1d\xceg\x80-
\x03\x86\xc1B\xfa\x1cU\xf6\r\x84\xdbY9\xb7\xc6\x02\xc2\u04c3\x1d;\xcb\xdd\xf1X\xa7\x83\xe9\r\x
1b\a\x00\xf0\v\xbf\xfc\xab\xb4u\xd3\xf4zg~!\x1c\xa1\x90?\xf6\xd8\xc3x\xcb[\u0789\xa3G\x0f\xa7\x
dd\xf5\xf7\xfdW(6\xaf\xf8\xeaW\xbf\xbc\xf7\xe8\xd1c\xaf}\xf0\xc1\x1f\f:\x9d\xb1\xab\x1a\x8d\u019e
\xc5\u0165\xb2(\n\x18\xe3\x042Y\x96\xa1\xd5j\u0271\xb16///?\\\x96\xfa\xe8\x15W\\Q\xec\u06f7\xf
7\xe1w\xbd\xeb=\xf7\x12\xd1s\x1f\xff\xf8?\x03\x00\xbc\xf7\xbd\xef\xc1\x87?\xfc\xf3\x1c\xa0\"\x00x
\xc7;\xde\xc9\u007f\xf5W\xff/n\xbd\xf5\x8da'\xf1y\xffU;>\xf3\x99\xcf\xfc\xad\x9f\xf7\xa1C\a\x82]BK)
IB\xb8\xc1\xdd0\xe8`u\x89\xbc\u06c7^\xd10\xd0>\xb6\xcdC\x1d\x9c$\xb1F\xaa\x9cg\xba\xa0^\x1
c\xc8\x02\xaa\xe0\xd8\xe5\x11\xfb\xa1\x19\b$\xb8\x1e\xda3\xd4\xd2\x13\xb9A\x1bQ=U,\x82\xd1\
x02\x10\x1ah,\x1b\xc0\x00&\xab\x84\x8eQ\x96\xe2\x17X\x80}\a\u0315\xfd\xe9pr\x19\xd5a\xb3d\"\
x8a
#R\x1a\xc8r\x1b!\x05b@\x1a\xf7\xfc\xaa>\xad\xae\xd4$\xf8\xae\xd6\x12\xac\xf4E\xdd\x12Tn\xd1,\

x9dK\xa1n\xb8\xd2#\xe06G\x94\x88?K\u02c8\x1a[r\x9d\xb9c\xbe\xb8b.C\x81\x17\fEa1r\x85\xbcd
F^X\xe4\x03\x8b\xbc0\xfe\xf5&\a\xfb\xd0\xea\x18S\x99[\x88\xbc\xfa@\x90\xf5\x1c\xf2Z\xa6\a\xd5\
xf0*f\v\xd6\xd6\v\xe8\r\xac\x1fb\x82\f\x98\f\x044\xa8\xb0\u8d41\\0Z\x02\xf0\x96\xe9\xc9{jcQG*\x1
8\x8a\x85\xdc\xcd*\xc2\xe3j\x03\xf4\xfb\x1a\xbao
\n\v\x19\x86\x99\xfe=g\"\x18\xb8\xc5UD\x18T@\xa02\x1b\v\x10Tx=!\x04t>@\xd1[\xf2\xbb\x00\xe3
rA\xfdy\x8e\x8fOb\xe7E\x17K\x87\x02l\xbb
\xeb\xda\u02f2\x98\x87c\u07fek\xe2\xf7\xa7N\x9d\xc0\xdc\xdc\xd6Q\uc387\x00<\x04\xe0\x8b\xbe
\xb8w\x00\\\xe2\xe1\x9a\xe1\xfeQ\xfa\x06\xebq\x1a%3\x04\xf0\u05b7\xbe\x15\x9f\xf8\xc4'\xf0\xda
\xd7\u07bc\xeaw\xb7\xde\xfaF\x1c=\xfa\x1cv\xec\xb8\xf8E?\x97\x83\a\x9fF\xa3\xd1x\u07bf\xf5\x8
5\xdcw\xa0>\x1e\v\xa6>\x00\x00\x90\xf7\x96\xd1]8\xeb@\xabA\t\xce\b\xba)`\xd9\x0f$\x13\x18\x8
2\x13~\xb5\x1drV\x92\u05ba\xe2g\xfd\xb0\xcc\xfa\xa2\xe7)\x11\xf4|#\xdbT'\x13\a\x9d\x9c\xdc\xd6
A\x17T02m\xc1\r\x81A[\xc0\u02aa\xa0\u04c8D\x1c\x1e\x01y\xa64>\u02d1\u679c\n\x83J\x8b\xac\
xe7\xb0\xffpr\u00b3o\xc8\x0e\xe1\x1a\xc3\xd6\x00\xfe\x8eC\u0383\x95N %4\xa3\u0673
\x16(\x9b.FO\f\xf1\xdf\x05\ra\xfbn\xd6\a\xcd\xd1\xd36\xce'\x94\xa0\x1ac\xc70\xc3h\xf7\x05\x10\x0
4U\\\xc1\x9a.\u04cb\xb4T\xdf\r\xab\x01\a\xa5\xb8E(Y
\xd9\x01O\x96C\x87\xec\xbai\x1b\x92~\x9c\x91\x80\u01f6\r\x98]3 4C\x82P6]\x91-
\x89\u0410\x80\"\xe7\x9e\x19\xb8\xe2A\xbe\x1f\x86\x9fV\xdb8\xec\f_\x85\xb6\xe8\r\\\u01df\x95\xb
e\x90'\xde6\x14\x06\x9d\x91zX\xa7\xc2:h\x8f\xa3\xe5\x04E\xc30\x05\x9d\xf7\x90/\x9fw\x9b\x03S\
x82\x8d\r\x8c(j\xb7\xdb\u0631}{\x1f\x006\xcdlZ/\xe6\x17\xf21\xaa\x9033\xee\xbd\xf7\xbb\xf4\xe0\x
83\x0f\xd0\xd2\u04b2\xbb\xb6\x9ch\xe9\xe1\x17\xba\xbf\x8f\u007f\xfc\x9f\xcaM\x9bf055\u016fz\u
056b\xf8\xc6\x1bob\"\xc2\xddw\u07cd\xbb\xef\xbe{\u037f\xfb\xdb\x14r\x00\xb8\xe4\x92\xcb^\xd4\x
ed\x83#b\r\xef\xf7\xb4\x0f\xf2\xdd9\x9b\x12\x18\x14\xe0e\x06\x93\x02\x87|DO\u007f\v\x85Cx\xdb
\u046a/\xa5\xd8\xe9\xa9\xc2\xe3\xab~\xcb.\x18#r-
\xeb\xb5/0Z(\xf8\xb7\xc8\x04\xc0\x0e\xd7f\x80\xa8\x85S\x916z.\x95=\xefH\x87Q\xfb.7\x8c\aG\xfa\
x13&4\x97\x10~\x1c\u050b\x14:r\x02Dn\xa1V\f\xd4\xc0-
L\xc1cE\x9a\xb4`\x8c\x1a\xab\x0e5\xf8\xec\u0395\xe0pzA\xfeu\xe9\x19\x10\x00=&\"\x85e\x94\xa
b\xef0\xf3&\x8d\x1ce\x8f\xfb#\x11\x1aU\x8f\x9d\xec\x16\x86m\x81\xc3\x0e\xaao\x1d\xa7\u0727jS\
xc4\xcc#\x01\xdf\x15n\xad\xc1\x1e[\x8e\x8a\xe2\xf0\x86\x04\vZ\xdf\u055a@Q\xb2\f\x05\x02\x93\x
80V@\xdfZ\xe4%\xa3)\x81\x86\x00\xa4\xd7\xe4[c`\xb5\x01\xeb\xb0X\xa0\x1a|ZF^\x18\xf4s\x03S2
\x1a\x06P\xa1\xb3\xf0|Q\"\x87\x93\x93\xb7i\x11\xf1\x9d\xaf\"\x9d\xd3l\xec
h\x12B@*\t[\x0e\xa0\a]\a\xcf\x19\xed\xf1{a\xa5\xca\xe4`\xd0??>\xde\xf9\x1b\xbf\x1b\xe1\xf5b\xfe\
x12;(j\xa2\xe3\xf5d\x9f}\xf6\x19ZYY\x11\v\v\xe7\xbdO\xca\x00B\b\x8c\x8f\x8fcbb\x02\x13\x13\xe3\
x18\x1f\x1f\xb7;v\\l.\xf4\xe7\x96\n\x8d\\\xf1\x930\u05a2=\xb5\t\x9d\xa9Y\u75e15\xa8\xcb\x10\xc2\
u008c+\x80$\x94\x94h*\x89\x8er\xe2\x96\xdc\x00\xb9q\x8d\xbca\x9f#i\u06337\xb8\xea\xf4\xc2\xe
0\x93it\x15O@\xea\xa0t\x8c\xc6Tr\xa8\xa0yqQ\xb0\x9c&\x06\x1a\x03\x87e\xe7\x1d\xe1|Q<\x9f\x
8f}g&\x92\x87K\xad\x188\xed\xf6\x87\x02\x96\x85f\xa8e\x03\xd5s]k\xb0\t\x10\x9aGx\x82\x8f\xf6_I
U\x9b\xe4\xbbHa\x1cvm\x15A0\xd0\xe8:\xb8\xced\xfe\xb9\xda\u054b\x01\x0f}\x9f\n+\x13K\x97:\x
c7|8\b\x9a\x87\"CAn\xd1\xed\xdb\xc8\x1b'c\xe12\xe5\x9c\u0692|\x00\xb2\xb5\x16\xec\xbd\u01a3
B3\t\x99p]9;u\xa7\u05700\xb0d\xbdI\x8e/\xaem\xf2\"+\xa0o\x19\x05\x80\x8c-
\x945\x10\x9c\xb8$\xa6P\x8e\x05\xf2\u0720;0\xb0\xa5Ef\xfc0\x99\xa8\x06\xd5\t\x1b:\xf2J5\x15\xf
8\xe8Lu\xfc\x8c\u00c4]H\a\xcb\x18\x8d\xb2\xb7\x04\u05b9\xeb\xee=\xc3\x06\x04\u03b2\x06\xc6\
xc6\xc6\xe6o\xbd\xf9\xc6S\x00p\u026e\x8b\xd6;\xf3\xff\x15\x8e]\xbb.\xe5:6\xf1\xd2<\x94R$j\u05a

0.H\xc2\xe8\x12\u04db/\xc6\xcc\xd6Ka\xf2\xbe+D\x85\x81Xf(\t\xa0\xcdh\x801\x91\tlk\x13\xba\x86p\xaa\xcfCb\x14@\xf6-
dnk\u04514\xf4\x1dV\xf7\u02a8\u0665V\x81\x91\x80\xf1\xe1\xcfI@M\x10\xda
@\x1d`'.bF.\b\xdc\xf0\xde,\xb6j\xbd9\x8apFy\xc2r\xec\xc8\x1d\xf7\x9d\xa1z\x06\xaak
\x8a$W\xd2\xfa\x0e\x9b0zQJ\x8a:y47\x9a\x81y0\\\xa4\x18\xbb\xa7j\x8a\x9e\x81\x96\x84\xa2\xe3\x02\xaf\x89\xebU\x9c\xd6\xda\x04\xd0j\u007f\x95\xd5\u007f\xc3\xf5\u074f\u01cdEi\xd1X\xd1P\x03\x03\xd2\xd6\xd1\x04=\x93\x04\x91.h\x00\x1f\xda\xc06\xfcn(\b\xc3\xe3\xe76t\xec\xde\x13\x9c\x03\xe3F\xc03b\x04\xd0\x12Q,Uh\x8b\xbc\xd0
\xb6h\b@\t\x06\xac\x89|sk\x19EiQ\xe4\xce\xeeZ1A\xf9\xd7\xce\x18\x87\x8b+\x12n\u0776\x9e\xba\b\xae\r\xf6\u067f\xa7B\b\xb0p\xb9\xa0L\x02\xf0\xcai\xa9\x14L\x99\xa3w\ue933\x15\xb0\x1a\u0194N\xf0\u0116'g&09>\xf6\xdc\xc6\u0249\xc7\x01\xe0\xdaW]\xb5\u0799\xaf\x1f\x17\xcca\a\x83\x81\xa6\xe4\xc2v_\x0ev\x99\u06b4\x03\xe3Ss\xd0y\x0e\xc1\x16L\x04QZ\xd0J\xe9\xdc\x0e%\x81\xacA\xc9\n\xda\xc2\xcb\xdf\x13\xcf\x13\xed\x04\x1c\x94\xd0\xdb\xc8>_\xe1\x1b\xb6{\xad\xb7\xa1\x01g\x0f\t\xf1\x9cH\x1c\xa9\xb6\x89v\x9445`@\x18\x94\r\x11;\xfapq\xa7\xdd\xfe\xa8\x90y\xf6\x95\x91\xc8q\xdaU\xdf@\x1a\xae\u01fc\x99\xb5h\u018c\x11%tu\xdf\x1e-
\xbd9\x0e\x14\t\x0e\xda\u023a.R\xaf\x18\x97\xe0\u0107\x9cj\x03\xe7:\xee\xcf<bm\x1c2\xf5\x8a\xc1\x0e^\xcd\n#\x1c\xcdr\xa9\x84\\\u0480\xb1\xce\xdaW\x9bD>o\xaa\x82\x1e,i\x13\xf1N\x98'T\x01\x13\x15\x8e\x0e\xb6\x91Ji}Q\xb6l!\xfa\x16\x86\x85\x8b\x04\xd4\x1e\x1b7\xae\x1b\u05de\xad\x12\xe4\xf9\x96\x19\xd68\xe6\n3#cB\xe6\xc5d\xde\xe38J\xf2\x03\xccRE\xcbzk\x83\x10q\xe4\a\v\x04\xf2Fo\xce$\x8e\x84\x80P\r\x94+\xf3X9y\b\xd6j\u8c80\u0445\u007f\xad\xd8n\x9e\x9d\xc5\xd6-
sG\x88h\x05\x00v]|\xc9z1_?~\xbc\xc7\x03\x0f|\x1f\xaf~\xf5u\xc8\xf3^\xb3(\x8a\rU\u01a8\xb7r\xb2\x16RJL\xcelA\xa62\x94\xdd~\xf4\x90\xb5\x02@nA+\x06F\ttK\v\xddw>&\xa5\xe7f\a\xbf)\x99[\x88\xc2\"u\x06\xac\xb7\x94#@s~\xbei\xf6\xc3_Q\x00\x00
\x00IDAT\xa8\xef\u01bds\x96\x15nX\u82b7\xdb-
\xb3\xad{9e9C.\x1b\xe4D\xce\xcf%\xe9\xc4y\x04lQ\xc1-
\xde\u042a\xe4\xaa#\xb7U[\x1d\xed\x06\xaa\xdc\xf75\xbcP\xd3\u0166\x8e\x9bG7\xc6Z:}\x80_\x18\x8d\xae\x06\xc0(\xc6\xd5\xd0b\x94@7#\x1f\xaf\x1a$\xc6\xf5\xd1X\x87q\x97\xdaK\xdb\x19$\x01+\x04Ta!\u03d7@\xee=\xe0\r{\x01\x8f\x89\x1e(\xd1\x0f,\xfc\x8c\xab\xc2\x1e\x87\u00fe\x98W\xe1\u02ee#\x8f\x03\bN\xa8\xa9\xa5\x81\xb0\x06Zy\v\x8a\xa8nr\xf7\x19\xbcj\x8cGA\x02\xe2\xa2\x18P\x815\xe4\x1f[\x10A\x06;\x84\x04o\t\"(x\xbfu\xeb\x19.\xcc\x16\xde\u0165*\xee\xc2\x05\x90\xaf\x9c:\x8c\xe5\x13\x87\xc0\u01a2\xcc\xfb0F{\xc1
\x1a\xadV\x13\x97_~\xf9\xa3\x17\xfcn{\xbd\u013d|\x8e\x83\a\x0f\x11\x00\xfe\xcaW\xbe2\x93e\xd9\xcd5?\x17r\x17Tsl\x12\x13\x1b\xb6T\xb8\xa5\xb5`\xeb\xd3\u7640\x9e\x86Q\x80!\x89\xc2\xe3\x00%\a/oDY\xb50\x88\xb1[\xc18k\xb4x\xfe\xf9\x8a\xfcj\xfbB\x0e\xf0\x8bq\x8c\x10p\xc5(a\xf2\xaaD?DT\x03\v@c0\xa1`\x9bT\xc1\x11C\x0f\x1f\x9d\xbcC!\xb7N\xce.\xca\xcad*.(5N>\xaf\x02Uj\xb0C\xba\xe3H\"\xf0j\u05b5\f\x10\v\xe7\xaaH\xd5.&\xeb[\x80\r\u028e\xf0\xe6V\xd5\xcea\xd8J\x97B\x18r\xf4\xff\xae|Gl\xaea\xcb2B# \x02\v\x82*-
\xb2%\r10\x91\xe7M!<\u067f\xef\xa9t\x9e\x83\x978\xd8u\xec6\xc9w\xa2Du\xe9\xcdZ\xc2\x10;\x1ac%\xd9~\u04baA\xb2\t\x1cO;\x94\xfb\x17T\xa7\xfe\xf5\x94\xccP\xda\x0f\x9b\x03\x87\u072b9\x11\xe9\x96>\xc0\x82R\x9f\x1a\xe1\ny:\xbf\x00E\xd5.\bP\xcd&t\xd1\u00d9\xc7\xefE\xb1\xb2\x80|\xd0E\x91w\xdds\x90\x02\xe3\x9d1\x8c\x8d\x8d\x1d\xb9\xed\xb67\xfd\xc5z1_?.\x98\xe3\xe7~\xee}\f\x00\x87\x0f\x1f\xbd\xe2\xe8\u0463\x1d\xf6pBhj\x98\r:\x13\x1b19\xb3\x13\xba\f9\x8c\x1c\xa5\xe2L\

x02\xa4-
xE\xc3H\r\x9b5\xc0\x02\u0436\xa2\x83\xc9\u04b1\"R9(%6\x1a\xb4*d\x99F\xa3\x15D#\xb1aw\xf1
Wj\xd1\u062c\x86\x02@\x1c%\xfe\xf0|\xee\x96\xd5\xc8'%L\xab\x12\xca\f\xd3\x0fA\x1c\xe3\xc4D\
xe9\xf0\xfeXP\xfc\x13\xa0\xc8J\\K\xca]\xcbMJA\xf2\u06a8\x80\x12\x06M\xa0\x04\r\xc3\xef\xc2\vx\
x82\x9f;gT\x9d\nP\xa5\xf0\x04I{\xf01\xf1?'x\xca_a\xc0\xdaT\xe6QB@\xb0\x80\xea\x1a\xa8\xae\x8
7W\x9cK\xd8*\u07df\x00\x9b\x04\x13\x1c\x8e0\x8bM\xd01Nr\x908v\xc4Q\x1d\xed\xb9\xe3\xae+\x
ae\x1c\u0365e\xd8\xcc\xdb\xd2&\xbe@\xe4!!cl\f\fQ6\x14*\xaa\x99\x93\xa5a\u0615\n\xb6\x82\\l\xb
0\xe9\x8d\x0e\xa3\xae)\x11$\xc1R@6Z\x00I\x9c~\xec\xbb8\xfb\xf4\xf7\xd1\xef\x9eG>\xe8\xc1\xe
8\x12B\bXcx\xf3\xe69\xba\xfe\xfa\xeb\xe7\xf7\xee\xbd\xea\a\xeb\xc5|\xfd\xb8\xe0\x8e\x13'N\xbc\x
f1\u0631\xe3U\xac[\xa0\x9e\xc1\xa23=\x87\xe9\x8d\x17\xc1\xe6eT\xf5\x81\xc8yZ\x04\xac\xb7`\x8
8\x15\r\u06f60M\xe1\xb0M\u007fO\"\xe7\n+\x8f\x85kh\x1aGk\x10\xcb\u04e4\xddx\xfb:o\xbb\x86\x
bd[$\x9d.\xfb\xce\xd9ue,\xc8\x1b(\x11Ta\x81e
\a`Z\"v\xe8<40\f\xb6\xa8j`A%GC)'\xf6\xf4~25o\xd9a\x80\x85VA8\x15\xf0\x9f<M\x9b\x90g\xc2j\x17
\xe6\n\x89\xf3\xa5\xb0@\xe6\x19:\u0178\xb3\xff\x05[\xb06Il\x9a\xad\xba\xe6\x14\xdb\x0e\x10\x8b
6Q\xfe\x1e\x8a\xac*\f\xb2\x15\x03*=\xbe\x1d}u\xbd\"\u0606\xe2]y42\xd7\x03#\xc2\xebm\x83\xa3\
x0eW\xc3\xd0\u040d\x1b\x138\xe76\xd9\x121\xac\xb7\u0115\x02\xb0\xd2\x17x\xebY'\xd6:\f\u076f
\u007f\x99u\xc5<\x18f\x85\x05L\xacN\xaf\xf6\x1f\t\xbfK\x10\"\xb0\xde!H\xfa\xbf\x0fb!\v\x92m\x90T
8\xfd\xc4}8|\xef\x9dXY8\x8db\xd0s\xaaO\xb70p\xb3\u0664\xb9\xb9\xcdx\xf5\xab\xaf\x8d\u2f75\xf4
(\x17\xc2!\xd6K\xdb\xcb\xe3x\u6667B\x91\xe8\xe4\xf9\xe0CeY&\xb6\xb9\xae\x1b\"\x12\x98\x9e\x
b9\b\xe3\x93[a\xb5\xae\xb6\xd2\xecq\x14\xed}\xd0\r\x83rvI;%bP\x01iv\x1d\xed\xf0\xb6\x19\x89D\
xbfFf\xa1\xb5\x91\x96\xe1\x82I\xb5\xf8\x9f\xb8P\x84\xc7J%Zd\x87\xa8\x8d\u059dWs\xd9@\xe66
4\xed#G\x97\"\xb7\x8eO\xceU\xc7\u01e9K\x17\x8f\xfa\xab\xe1\xd3\x1fAQ\xe4\xd1{\x11\xb2~\xa0\
xca\u4564\xc1\x03\xc5?\xb0a\u023eFv>\aus\xe7\x17b\n\xa7\x03\xd0\x05\u0614\x0e\x13\x0f\xe6
U\xb6\x8aV\xe3\x98.\x15\n\xbc\x85\xd0\x16\u064a\x86\xccMe`eS\x88\xc3e\xe0\u054a3\xb8\u05a
9W\xde\xf8\xec\x98\x1f\xfe1l\x90\xe6\x1b\xeb\x04?6\xc9\xe9\xf4\xefQTezz'\xb4\x1brF\u007f\xfd$\
xbfSZg\x95\x00\xcb0\x9e\xa9B\x00$\xeaA\xe0\x11\xff\xf6\xda\t!\xa4\x1f\xde\x10\x84\x94\xd1\x04(
\xec\x84T\xbb\r\xa1$\xce>y?\x9e\xfb\u0397\xb0|\xe60\xcab\xe0\x14\x9f@p9\xa5\xd9\xd9Y\\s\xcd
5O\xde~\xfb\a\xff\xaf\xf0h\x17j!_/\xe6/\xa3\xe3\xd2K/\a\x00|\xf3\x9b\u007f>\xf7\xcc3\a7\x17E\xee
Y\x01\x0e^\xb0\xc6@eMl\x98\xdb\x05!\x9b\xce\x1b\xa3\xa6\xbeC\x84]\x9c5\x1eCu-
\u050aF\xd8\xc1\xbb\xc1\xe7\x90\xe9v(\xe4r\xa8\xa3]\xa5\xe3\x1f\xa2f\xd4\xc484:\x95\xc7:>\xbb
H\x1e/\xa6\xf2$\x18s\x18\x94\xca\u0722\xb9d\x90\rx5:O\xa1+w\x1dk\x8a\x92\b[Q\t\xa96~\x1cm
S\xa3v\x0fS\xea\xb9>\xf4%\xeb-\x01<\x1f:\xf0\xf1\x85\xb6@Q\xba\\\u0322\x80X\x19
;\u05c3\xe8\x0e\xbc\x87\x88\xb3\x8e\xb5Z;K\xc7\x00\x97p*\x87\xaf0hf\xebT\xac]\r\xd9\xf7\xf0\x8a
\xef\xe6\x899\xc1\xc28\xc2'q\u01d6Z\xce&\x9dz\xbdsw\u007fk\xb5\xf1\xbeE\x1c\xb1\xed\xcaL\x8
dc\xda\x12\xc0
c!Jw\xbe\xe4\x87\xc1\xe1\xe5\x91\f(\x1df\x16\x15\xfbH0U\x9e+)\xa7\xde\xffCx\x97S\xe7y
\xdc\xe03(B\x05!k\x8d\x81m\x89\xe3?\xf8\x06\x0e}\xeb\vX>y\be\xdew\x8aO\x0e\xc6q\x12\xcdf\x1
3{\xf7^\x81w\xbe\xf3\x1d\x9f
\xa2\x02\x00\xbe\xf8\xc5/\xac\xc3,\xeb\xc7\x05\xb4z\v\xf9\xe6s\xe7\xe6\x91\xe7E\xb4\xbc\r\x95\x
85\x84\x84h4\x87\xe2\xba*\xa5%\x05\x01L\xa0\x8d0\xa0z\x06\x99\x00tK@\xe5\\\x85\x19\u0512\
u042b\xa1\xd5jT\x85\x86\x18}k\tq\xea\xaaE$\x14\xf4X\xb8E:\u0324!%\xb7\x83\\dn\u0440\x06\x93
\x84nz\x8d\xa0\u007f\rT\xe1\x15\xab\x11[q\xf0\x920Im\x16XM\x00\x1f\xb9\xb5\xa8o\xffk\u03c6\x9

1.3\xee|\r\xbb\x01.\x12Z\xa0\xe7\xbd[o\n%\xb4E\xa6\t\xb6C.B\xcfw\xf6\x8c49\x89+\xd1&\u06da\x8dm\x96[\xa8\xdcQF\xbd\x02?\x89\xacOv\"\xf1g\xc904\xe0\xf3\xb1\x90\aY|\b\u00a6\u02b66\fJ-j\xf0J\x94\xeb\xa3R_*r\x81\x1e\x9a\xaa\xe1\xabd\x82\nj\xe1\xd4R:\b\x8a\x82\xc68}\x9e\xd5\b\xb6\n\a\xf7\x8b\x8c\x90\nY\xd6\x06I\x81\xe5s\xc7q\xe4\x89{p\xf2\xc9{Q\xf6\xbb1\f\u00c5\x9a\x13\xa4\x92`\u02f8\xea\xaa}x\xdd\xeb^\xf7\x1f\xdf\xfd\xee\xf7|\x15\x00>\xfa\u044f\xd2\xfb\xdew;_\xd0\xd7\xf6zy{y\x1dKKKse\xa9\xfdE\x97\x94
\"Xk0\xc8W\xc0#:M\x1f\xc1\xe8qj\xcf\x06\xf0[\xe5F\u03c5\xfe\x8a\xe8Y\xee\xf1KOidO\xfe%\x1e\xeaVGu\xdb\xc35\x91\xab\x01e\xed\a4\fkT\nS\b\xaf\xe7\xf6\u007f\xc7IJ\x0e\x19\x86\x1cX4\x96-T\x81\xc8[\x17\x9a!\xbb\x9e\xc1\x82j\xb8'-
Cr\xe2OO\xcf\u007f\xb2\\\u06d2\x8c(\xf3\xf1iT\u07be\u0387D\x83\x8b\x01l\xeeS\xeaKO',5\x10%\xf4\x16\"7h,\x97\xceo\xdc\x06/\x14\x93\xb0M\xc2\xe3\xa4\xee\x86\x16j\xa0\xa1r\xebw\x19\x1c\xe1\xa7t\xf7\xe5\xfcTL\x02\xa7\xd8X<\x81\x14\x8e\xe1\u029b\xc5T\xd0\xca0\x9eN\x1e\u06a9Ac\xf0\xec&[9-
\x8a\x84E#\xe0\n\xb9\xb4\x9ez\xe8\xc3]\x04Q\xfc\x8aiP\\}v)\x11\x84!\x0e\xb9%\xb2\xac\x05\x10ci\xe1\b\x0e\xfd\xcd\xd7\xf1\xc87?\x8b#\u007f\xf3M\xe8\xbc\x0f\xa12\x17\u007fH\x0e\x8e\x91J\xc1\x1a\x8b\u077bw\xe3\xcdo~\xd3\xe3\xff\xe2_\xfc\xef\xff\f\x00~\xe6g~\x9a\xde\xf2\x96\xdb\xf8B\xbf\xb6\xd7;\xf3\x97]g^\x850\xd7\n\x8d\x10\xb0\xb6\u0120\xb7\xe8\xd3}j\x16\x84C\x05\x98\x12\xcbU_\xd0KS\xf3\xbd\x88\x05\vU\xe8A\xed\xf1hDXD\x8d\x88\x9d\fCiD\xe7;\xb2)\xa6\x8a\xd1\x12\\\xbah(\x17\xde\xe3\xdf2\xb7h,i\xe4S.(Z\r,dak\xc3Z\xb2n\x0e\x00\xfc(Y\x04#h\x97C;\x8d
H\x8aV\xae\xa1\u02e5*\x12-
(&#VM\x95\x0fwX\x91d\xc1hXF9&\xa0\x9bb\u0235r\xa8#f@\xe5\xce\xf0L\x1a$\x90T:\u04f0C\xe9=~Qb\x8eC\xc7\b\xa9p\n{P\xb4\x92\x8d\xae\x87\xa8\xbcpl(\xe4\xfe\xef*Q\x16\x92\xa1)A\xb1c\x9e\x18\xe1(\x88\x8a\xadw\x00\xa0\xca?\xc5&Cu\xaa\xad\xe4N\xd1\t\xf6\xbeA~\xc1\x90\x19\u02a2\x8f3G\x1f\xc6\xfc\u0267\xb04\u007f\x04\xcbg\x8e\xc0Z\x03\xa9\x1a\xee:\x00\xc1J\x05\x995`m\u0266,h\xc7\xce\x1d\xb8\xf1\xc6\x1b\x9e\xfc\x95_\xf9\x95\x9f%\xa2%\x00\xb8\xe5\x96[\xf8\xa7~\ua9f1^\xcc\u05cf\v\xeah\xb7\xdbOeY\x06)E\x15\x1b\xe7\xc4\x11`mP\xf4V\x92Z\x1a\xe8p\x14)^\x84\xaaH\x06\x95\x0e\x99\xca\xe6\xd6J\xbf\xe5\x0eb\x11/\xf4\x11\xb6Nba^\vv\xa6\xd1?\xe7\xa1\xda\xe8\u04cdyDkO\u05a9P\x8d\f9\x90ud\xc4\xd5s\u03eb\x16\f\xd3\x10\x90=\xafX\xb5\xc9I\xfa\xe7\x14\xec|\xe3\x02E\xc3\u0388\x8cQ\x88z=\xc0\xc4\x15<\x1b\x14\x96\xa8\xa0\v\x88\n\x1a\x89\u007f\xe8\x17\xb6\xb8\x96Q=\xf8Zj\x06y\xb5\xa8n\x8a\xdaZRQ\x16\x1927\xce\u007f\u0778!v\xf5<\xb8\xce\xebDU`+\xf6\x89\x8d\xafU\u0569W\"*\x8e\v\xa8\xf5b\x1fNT\xa9\xd503x\xa8\xb07\xacw\v\x96\xcb+\r\xcfI\xb2\xc3\xc33\xdfp\xb0\x17\x1a\xc1CI\x91zjy\xd5\xce\xd1c_\x11Z#!Q\xe6=\x1c;\xf0=\x1c;p/\xfa+\xf3\xaeOWM\xa8\x84\x11\xc3\u0110RA\xaa\x06\x0f\xfa+\u0632u+\xae\xbf\xee\xfa'\u007f\xf1\x17\xff\xe1{w\xef\xbe\xf4q\x004?\u007f\x86gff_\x12\xd7\xf6z1\u007f\x99\x1d\u059a\xefl\u06b4\x11Y\xd6\xf0\xb8y\u0569\x1b[\xa2\xe8\xaf\xc0\x98\x12\x14$\xd0q\xe7JUWe\xd9\x19MyN\xb7cdT]\x19+D\xae4\xfb!\x1fE\xee\xe2\v\xb4\xb8<\xe4}\x9b0Ix\xa8\u00a7!\fq\xdc\x19\xdc\x15\xb5ss4\xa2:\xe7\xaa\xe5\xe6\xb8-
\u03fa\x16Y\x8f\xeb\x186\rm\x0eR\x1d=\x85x\xbbZ\xd4r\r\xcbO\xb2#+\xfc\x98\x11=Ol\xe899Y\x1b\x12ug\x8a\xabGn\xb9q\x11u\x10\x15\xebD2\xd0\xec\x1a\b\xcb([n\xb7D\\A\"r`\x90\r\x8c{\xaflE\x19\r\x90S\x85.\xa36\u8904\xc1b\xb9\xf2\x19\xb7\xde3%\r\xa7H$\xa2\xfe\xed\xe3jw\x10\x96\x8a`\x9aOT\xb1g\xa8Z\xf6\xd8r\xe5;\xceT\xc9\xf1}\xe1%\"?
v\xb4\u01b8\x03\xe1\n\xdc\x12\x14\x18Y\x0e_[8u\x00\u01df\xb9\x1fy\u007f\t\xb2\xd1Nf#\xfe~\x02

\x14D\x84R[\xda4;\x87\xd7\xdet\xe3\x93\xff\xf4\x9f\xfc\xc6{o\xbc\xe9\x96\xc7\x01Px\xec\x97\u032e{\xbd\xbc\xbd<\x8e\xa7\x9f~\x02\x00\U00016dfc}qvv\xf3q)e\x92\x14C\xb1\x9b\uebdcG\xde_\xae\"w\x12\x8b\xbd*\u6362\xbc^\x98P$\xd8[\xbc\xfad\x1a\u02f5b\x94\x0eBG\x17\xf1\x14\u007f\x19\xfe9\x8d\xbei\x8aj\xac\xc6x@\x06\x90E\xa5\x10\r\xe7\xe5,\x00\xdc\xffe\xc9P\xb9\xb7\u007fM\xf3\xe2\x14\xd5t\xfeU\x97]\x89cj\xcf\xc6\x17\x13k\n\x18=\x80\xd19\xb4\xcea\xb4\x0f/6\x05\xac)amj\x1f[%\xe7 \xc2\xd1\\1R\x12<\x1b\x9e\xe1\"K\x860\xde\xe1P[\xc8\u0720\xb1T\xa2\xd1\xd5 cb\xa7-
\n\x83\xacg\x1c[\xc4\xdfV$y\x9a\x96\x13\u02a2\xef\xc2\x19)\xab%\x04){\xcfr\x9b\xc07IW\xcf\xdes%(b\x839\x96+\x84\"R_\xab]C\xaa\u016c\x16\u02cc\x05T\xf0h`\xef\xcb\x12>k\xe1M\xf1l\x15\x11\xfe/\x84\xf3V\xa1\xa4+\x97\x12\xba\xe8\xe1\xec\xd1G\xd1[\x99w\x8d\x8a\xad\xf2=\xd97\v\x82\x84\v\xea\xd6\x1a\x17\xed\u0709\xf7\xbd\xef\xfd+\xff\xe1?\xfc\xeeo\xf9B\x8e;\xef\xbc\x13ke\x12\xac\x17\xf3\xf5\xe3\xc7z\x84\b=\"Z9}\xfa\xf4\x97\x01\x80|vg\xca\xd3-
\a]\xe4\u0765\xea\x02\x1a\u00a9E\x9a>\x13\vd\xbd\x8c\x86\xa81w-
\xd2\b~\xf9\x1a\x85\x9c\u05c0\xa1\x87\xb0sZ\x033\xafq\u0343J0$\x01\x05\x9es\u02b6\xb1\x15O=\xdc\x16\xd6eDj\xe9%4f\x88#\x9e\xbc$A1\xca
X\x1f\x00lt\xe1\xa4\xf5)4\x118\xd8<\xe42\x98b\xf95\nPb`\x1537\xabs$\xed\xa9\x8b\xda\x15[*-
\xb2e\x8d\xac\xeby\xe6\xda\x19\x84\x91\xf6\xd2zckB.$E\x9c\xb9\x12\x04\x85\x02n\xbc\xd0'\x0e@=\xc6]\xa9wiUh\x84e\xaei\xc3\u023b\x18V\x8aYJ0o\x11?\x1b\x10\x02\x12\x14\x13\x80d0|K\x06\u035c\xe0\xe5B\b\xa8F\x1b2kA\b\xe9<\xf9\x93\x0e^\xaa\f\xc5`\x19K\vG=MQ\xc4\xddN\x1c\xce\xfa]\x811\x06Zk\xbc\u136f\xc7o\xfe\xe6o\x1c\u0771}\xfb\xb7\u00a3\xbe\xf3\x9d\xef\xe4\x97\xda5\xbe^\xcc_\x86\u01ee\u077bt\xa6T\x8d\xbf\x1b\x14r\xbap\xd1Y\xa8\rI\xb9\x06\xaf\n\x06\x84\xa5\xa1`\x06\xaa\x15R2\x88E
\xd2\xe5\x80\xe7/\xecT\x83\x9e\xd7f\xba\xa4\xb5/\xfa\x9b\xd4\rvca\t]\xb9\xa5\u06a2\x11\n{0\x80\n\t4\xc4\f\xa3\x00\xad\x9c`\x87-
\xaf\xe9$\x93\x9e\x905\xa5\xe7*\xdb\x1a%p\xd4*U\x1f\x89\xa6\ue0e8\u007f\x9f\xf2\xb3\xfdW\xe0\x8f\xbb<K\x1b\xbfH[g\u05fbRB\xf5J\x88\xdcq\xcf\xc3b@\xe0$\xe8\xc1\xc6\u0162V\x90\xa3}m\x85\x91\x10\x8d\xe0\xd4sE[\xa4\xa1.\xbbF\u4a7d\x06\x14\xf38\x1d\xc5\xd3\xf9\xcd;\xe4\xcd\xdb,\xf8\xe7\xc5\t\x03I\xf8BO\x00\xa4g\x9f,/\x1c\xc3\xca\xe2\t\xb7\x03!\u007f\vA\xb1S\x0f\x85;\x18j\x11\x11d\x10\x13\x81\xfc{\xca.\x88\x9c\x19\xa7N\x9e\xc4\xe9\u04e7\xce\x11\xd1\"\x00\xdc\xfd\xe7w\xbf$\xaf\xebu\xcc\xfcex\xccl\xdc(\x85\x14\xae
HT\xa0\x817\xe9\xd7e^\xebr\xabb\xc6\xd5\x169\xe5w\x0f\tc\xaa\xe1\xa7/W\"\x91\xe1'^0\xa3\x8a4\xa5\x83<Z\xa3cO\xb1t\x1e\u075eW1g\x1c\xe3\xce(\xca\xed+\xa0d\xb8\xdb\x176\x14\x15D\xbb\x80a\xae;'-
\xa8\x83V\xb4Sd\x06\xe5$\x0f+A\xa9f\x80\x85\x11i7\xa9\x1dT\xcc\xe6Ko\x1b\xbf\xb5U\x97\x19^\xaf\xe0\xf3^22\xe3\x1d\f\x93\x1d\x87+\xd4\t[\xd3\xff\xb7\xe6G\xce\t\f\xc1\x0e\xd7G\x12\xe5VS\x8a&&\xeaL\x15N\x9e\xf2\x86\\\x13/\"\xf6\x1en\v\x0f\xb7\xc5b\x9d2k\x02\\\x15\xfc\xc7\xc9Q\x14\x89\b\xb2\xd1\x04\xdb\x12\xc7\x0e|\x0f\u01df\u074f\xd6\xd8\x14.\xbe\xe2\xf5\x98\x9e\xbd\x04\xe4\\\xdd\\\u0291)16\xb1\x11[.~\x15\x9e{\xe2\x1e\xe8\xb2\xef\xe4\xfc\xc2\x17\xfd\x00\u01d0\v\xc6&b|\xef\xaf\xef\xc1\xefg<\xcd\xcc3D4\xff\u05b7\xbc\xf5%y]\xafw\xe6/\x93\xe3\xfe\xfb\xef\x8d\xdf\xcf\xcf\xcf\xeb\xb2\xd4U\xed#\x8a[h\xaa\r\x1c\xab|E\x1eQT\xe3P12\x18\xeaTF\xc1\xf5\"\xbe\xa6\xce&)\x00\xe9u=\x12nI\x19\x1f\u00eb\b\x11\xb8\x9eP\xba\xeaN\xa8v\xfbp\x17\xf5T\x1b*\u0639\x16\"2\xf7\x86\xa0\x960\xb830\xba\x841\xa5\x0f\xce\xe3j'R{\xd2\u0460v\xcdn\x1d\xa8\v\x8a\xe2\v\x91\xc2\xf2\xb18'L\x98p[\x1f\x80,4

KD\xa1\x13\x12(\x87\x13;,\x8e9\xa9\x9e\x85\x14\xbab\u02f0\u01b1n\xac1\xee{c\xc0~\x97\xe2d\xf7u\x1ac\xf0\u008f\xe2\x1e\xcf\a\x8f\x9e\xf3\xfe\u70bcca:\xe6\xb0u\x8b`\"\x02K\t\b\x01\x05\x01)\x15\xb2\xe6\x18t\xd1\xc3\xc1G\xbf\x81\x03\x0f\u007f\r\vg\x0eb\xe1\xf4!\xac,\x9d\x02I\a\xb3\xd8\xe8\xa9n\x905Z\xd8u\xd5\xebq\u054d\xef\xc7\xc5W\xbe\x0e\xd3s\xbb!\xb3\xa6\x87\xc0l\xdcG\x18k`u\x8eS'\x8fb\xff\xfe\xfb\xf7\xfd\xfb\u007f\xff;\xbf\xf0R\xbe\xc6\xd7;\xf3\x97\xc9\xf1\xb9\xcf}.~\u007f\xe2\xc4\xc9+\x8b\xa2p\x17\x90\xa8\x04@\xbe\x1f\x8a\x98fZ\xfdhT~dr\x1b&W\xb8\xb9\xc6Cv,\x05\xf6\xdd\x15q\x02\x81\x8c\xf07\xe7\xc4X\x8bW7\u0135`\x86\x91\xb5\xbe\x96\xe2\\7\xa6eJ`\vB\x8d\xeaG\x89\x1a\x95\x85\x8f\x1f\xd3\x1e/\x0f\xf7FTu\xdb$\xc0!\x8d\xc6\x18\xdf\xf5s\u0176\xa1d\x87\x92\x9eR-
]\x82\x86^N\xaa\xbd0\x9c\xa4,E>w\xb4\xa2Mp\xf7\b\xb3se\xcb\uba60qTKCV\xbdT1X\xd2\xf3\x8c\xbc\xf7\xf8\x8es\x92\x9b\u0249i\xa5\x0f\xd3\x18\x9ac\x04\xdex\xfc>\xb0A\x98*o\x14f\xefxY\x050\x13\x10#\xfe,\x18\x86\x80\xa6\x90h\x90\x04\f\xe3\xfc\xd9g\xf1\xdc\xe3\u007f\x89SG\x1f\x8a\x9d\xf6\xd4\xccE\xd80{i\xf2\xdcS\x96\x92E\xa39\x8e\xed{\xae\xc7\xdc\xc5\u05e0,{XY>\x8d#\x8f\u0743S\x87\x1ev]:\t\x94\xf9\x00\xf9\xa0\v\x80\xb1\xb0\xb0\x80o}\xeb\xdb\xefd\xe6\xdf'\xa2\xe2\x91G\x1e\xc2\xd5W\xbfb\xbd\x98\xaf\x1f\x17\xd6\xf1\xd8c\x0fc\u07fek\x00\x00\xf7\xdc\xf3\xad7\xfc\xd6o\xfd\xf37\xe5y\xee/t\x91\\\xb6\f\xa92\b\u0568\xbaIN\xbd\xbb+\xd3\xff8\x90d\x97\\\u00de_\xee\xac\aQ\xf3\x05!\xeb
xJ\xf0\xea\xb5\xfc\u0369V\xd8\xeb\xf8\xf7\xea\f\xcc\xd5\xeeU4\xac\xbe\x8c\x9cw\xacR\xfe\x04?tG\xa3s\x85\x9c\x05\"/\x9d\x86r\x90+*\xa4\x81\xb1%\xac\xd1\xd5\xd9\xc7\x13\xe6zd\x1b\xf1*J\xe5\xf0s\xae\xadp\x91\xceX\x9d;\xa7\u06e4\x14~\x11\xe9\x16\xdb\xdf6P3\xadO\xeb\x11\x04\x96\u038dP\xd8*$48\x15F\xffr\x0e\x83lJ\xe8\x96\xee\u0770\x81R\xe8?\fb\x84\xb7\x0e{\xf6\ty\a\xcd`\xaf\\\xadU\t/?\xf8\xa5\x84\xb5-
\fO\t\x10J\xa23\xd1F\xc3\x02\xfd3\xa7q\xec\xc0~\x1c{\xe6~\xac,\x1e\a[\x83\xe6\xd84\xb6\xef\xbe\x0e\x17_\xf9\x064\xc76@\x1b\x97\b\x94f\xd9\x02\"F\xd7\t\x99\xa1\xdd\u0684\x89\xd9\xed\x18\x9f\x9aA1\xe8b\xfe\xf8\x01W\u030b>\xac.
\x84\xc0\xe2\xe2\x12\u039c9u\xe5\x9dw~\xf5\x16\x00\u007f\xf9\x8do|c\xcd}\xe4z1_?~lG(\xe4\x00p\xf7\xddw\u007f\xfc\u0631cY^\x14P\xaa\xe1\x99\x03\x14/\xeaFk\x1c\xad\xb1\xa9*\xf1<\u059dJ\x91I$\x12\xf1\x90\xff\x9d\xf2L\x84P`5\x83J\vmC\xf8\xf2*,au!g\xd4\xc53k^M<\xac\xea\xc7\xda\x0e\x8c\xf5\x01f*:r)>\x04\xf6\x90\x04\xab*/\x94=\x16\r\x93\fm}\xf2<\x9b\u0485\x1a{\xe1I\xec\xa4y\b\u02e7zi\\\xf3\xd9$\x90IM\x12\x13\xc9\xdau\xef\xf6T\x94D\x02qH[\x87\xa3\xb8\xf6\u007f\x8a\xef\x1dy=PJ\xd7\xf34D\xaf8\r\x81\x0f\x15\x86\x9e<\xbf\x04.\xa2\xaa\xf7O\x05\xa7q\x19\xa0!\xc0)\x8a\u0390\x84H\xc0\xc7\xc4\tBc\xac\r\xd9\x16\xc8\xcb\xf38v\xe01\x1c~\xe8\x1e,\xce\x1f\x81.\xfbPY\x1b\x1b\xb6]\x89\x8b\xf7\xbe\x1e\xb3\xdb\xf7A\xc8\f\xba\x1cT\x8c\"F\xa2\xf8\r\x90\xa1\x1bp\x9ab\x00A@gr36\xcc\xed\u00b9\x13\xcf\xc4YG`\xc1\x94e\xa9\x1b\x8d\xe6\x96#G\x8e\xbe\x1d\xc0_\xf6\xfb\xfd\xf5b\xbe~\\X\u01d93'0;\xebl;\u007f\xf0\x83\xfdo\xff\xc4'\xfe\xf9O\x9e=;\x0f!$\x93\x10DB\u0536\xf3\xad\xce\x06\xb4;\x1bk\x8c\x860\x8f\v[k\xf77\xbe\x83\x16\x04\x92@&\tR\x917\xa9fP\u18fd4\xfb\x82>T\u022b\xfd\xfbPGMC\x9d\xf0j\xf8\xbb\x12\x86\xd7d\x93@m\x81I1\xee\u0564\x9bXtd\xf5\x05fp\xe1=kb\xec}T\xb3\xb8A\xa7\xd5\x15?;Q\xb3\xd6
\x13\xa2\xd50\u03ea\x85\x86*\xc9<W\x81\xc8\xe9\xb3\u4d10'\xf8yjW\xe3\xd6\\N|\xd0\xfd+\x14\xd7%\x02Y[\xbd_\xc1\x13%\xb0X\xac\xad-
\x85\x9c,0\xd5\x0e\x01\t|bk\x8b\x10\xa7p\xcf(\x9f_\xae\xa3[\xa8\x911\t*k
\x93\r\b\xa1\x91\x0f\xce\xe2\xd4\xe1\x838\xf6\xf4\xf71\u007f\xfc\x19\x14\xfd\x15\x10\x11&\xa6\xb

7a\xdb%\xd7a\xc7e7\xa339\v\xads\xe8r\xe0\xeeEP\x9c\xcfT\xf7\u02c8\xbc\x19\x87\xe1@@\xa2\xb7|\x16K\xf3\xc7\\T\x9c5\xc3\xef\x8a-
\xcb\x12\xa7O\x9f\x19\x03\x80s\xe7\x16\xd61\xf3\xf5\xe3\xc2:B!\a\x80/}\xe9\xcb\x1f=x\xf0P\xa3\xd7\xebCe\x19\xa5!\u0396\x9d\x05\xee\xe4\xc6\x1dh\xb4\xc6at\x19\u0558iWK!7\x11\x95\x9b\xa2R\x84L\x02\x94\t\xf0\xa4t\xe1\x15K\x1al}\xea<\x11J\xc3q\x98:*h\xad2\xac\xad\xeb*k\x1du\xf0\xf7`\xc2\xc8\x14\b\xc2PU\x1dz\x9c\xa1\xa2.\bPd\x80\x16\x81\xdb\x04\xd1\xf79\x98p\xae\xb2a\xb1r\u007f\xc70TQ\xfa\xa2w\x8d]#\xab\x82Fd\xd4\xf1P\xe7L\\\xa3\b\xa6\xb84\xa7.\x88\x9c.\x1a\xd5+\x13\x99$\x82`\x85w\x17\xf41w!e\x87\u0233\x8a\xac\xe3\x82S\xc26J\xef8\xfa\xb9\x87$\xb7\x14\xf2\xaa1U\xfcp\u04cf|\xeb)\u054888\xfb@\u620bK\xbf\x9b#\x01H\t\x91I\x186(V\xcecq\xe1)\xac\x9c{\x16\xa7\x8f=\x8e\x85S\xcfA\xe7}0[t&7an\xd75\u0636\xeb:L\xcf\xee\x01\x84DY\xf6\u076e(\xb2\x96\xc4\xd0\\b\xf5\x9a\"d\x06\x06\xe3\xf8\x81\aq\xee\xe4!\u05cc\xb0M\x82\xcc}<\x9d\x94\x90R\xf8\xef\xc5z1_?.\xac\xe3\xfb\u07ff\x0f\xd7]w\x03\x0e\x1cx\xf2\xda_\xfe\u53fc\xfa\u0529S\x90R0\x91\"\x17f\xcb\u0582\xb5Fkr\x16\x9bw^\u5de7v\x15\xb2\x11\xe8fQ\xf2.\xdc\x00UJ\x82\x10\x04j\x10\xb8-
\x80\x16\x9c\xea\x10\xec.6\xed\xca@\x99fa\"\x8d\xfa\xa9\xa3\xe1\\A\xab5\x98\xb8\xda\xc0s\x95(3t~\xb4\xc6\xf04\x14\xbf\U0003d400\x94\x06R\x02\x18\x03l\a
\xc9`\xcd\xe0\x9c<\x06LU\x19\xe3\xba#\xa2\xc3\u05a9V\xc4R\f\xbf\x86\xb1\xc7\xf3I\x02\xd6<\x0f\x9a\x92\xbf\x8b\u007fS\x1b\x84\xa6\x88QB\x89\x1c\xde\xd3\b\x00\x8a\xa2\xd25.\x8b>)\xca\xd6\xc47\x1c9\xe2\xd5\x1e
\xb1\xb3M\x9eD\x05\xb3\xd8U0\v\x00_\u0607\x86\xcc\xe4\x8al\xe0}\x93T`\xab\x91\xf7\x97\xd0\xeb\xcec0X\xc4`\xb0\x84\xc1\xca9\xf4\x97Nc\xf9\xec\xb3\xe8.\x9e\x811\x06D\x02\xad\xce\x04f\xf7\\\x8d\x8b\xf7\u0744\xd9\xd9\xcb!\xd1F\xbf\u06c3\u0579\xf3\x9d\x17\"IY\rV\xb7\x15k&\xce\x1a\xbc\nU\xca\f+\x8b\xa7p\xf4\xc0}\xd0\xc5\x00\xaa\xd1r\xf1yBF\x16\x8e\x14\xa4\xb2,\u00e6M\x9bN\x03\xc0\xf4\xf44\xaf\x17\xf3\xf5\xe3\x82:\xae\xbb\xee\x06\x00\xc0\x1f\xfd\xd1g\u007f\xe6\xf4\xe9\xd3[\xbb++Ve\rA\xe1B\v\xc3(v\x91q[.\xfe\x89(|\xa9\xa9@\x13\xa3\xadt\xa8%\x04A\t\xf7\u007fd\xe4\u0094\x05\x01S2F\xac3\x18\x8a\xad\xcb\v5\x95!\xd3\u8ac5V\x99$2\xd7\xfb\xf6\x80\xe9bx\x00J\x18\n\x19N;K\xf7#K\x00\tF&\x18\x92\x18\xdc\x04\xb8I\xee\xfb\x16\xc0\xe3\xe4\u0515\xc6\x0fD}A\xaf\x89z\xbc\x83!%])s\xe54\x99\x92\x13+\xb7\xc6\x04\x13\u786e\x9b\xea\xdd\xf1\b\x14\xaaF+$$\xb3\tA\x90\u0085X\vE\xdew^\xban\u0738\x9d\x86\r\xa6VBx\xa5'\xd5\x17\x87\U0003ace3\x1e\xa6\xf1l\x95\x1a\xb5\x1e$\x1d\x19$\u0273$\x92\x90B\xc5\xfd\x85\xd1\x03,\x9d=\x8c\x85\xb3\x87\xb08\u007f\x18\xfd\x95s(\x06\xcb(\xf2\x15\x94\x83\x15\x98r\xe0\f\xb7T\x06\xa9\x9a\xe8LOc\xf3%\xfb\xb0u\xefO`\xe3\xae=\x18\x9b\xd8\b\xbd8@\xb1\xd0s\u063a\x10u\x1dAt`LX5\xbe\x90W#\x1e\xb7\x8b<s\xf41t\x97\u0382\x84\x84\x14\x02\xa0\u0327\x12\xb9\x1b6[m\xa1\x94Z\xb8\xfa\xea\xab\xef\a\x80\xad[\xb7\xadw\xe6\xeb\u01c5s\xfc\xf5_\xdf#n\xb9\xe5u\x96\x99w\xfd\xfc\xcf\u007f\xe8\xa7O\x9e<\t\xa5\x94!!D4\xd8b\x97\xec.T\x86\xb9\x8b_\x89\x89\x8d\xdb\\|\x96\xf5\x85\x83\xd2B\x85j`\x1a\xb6\xe5\xd2s\x95\xa5\x005\x85\xb7je\xa0)\x80I\x99\x10\xb5\x9d#\x1e\xb3\x830\xb8\x16\xceP\xa1\x9d\xab\x80\xedT9Z\xc3\xdc\xebc\u0151~H~\xa2\x1ay\xe1\xfe\ue534\x90\x12
\x05`\x8c\x00U\xc1%\xd4\x06`\bb\x89A\x85\xb3f\x85\x00\xac\xf5FV\x94\xac0\x89\xf5\xe3*\x1b\x8f!\x8b\x82\u0618\xdb\xcab\x96\xbc\x83`2MD]\xd2?\xf4|\x89jCG\x10AH\x82P\x80Pn!\xa5\xb6\x04\x8dI\xa0g\xc0]\xed^y\xe9\x829X\x00\xd0\x02\xd6$+Qub\xf5\xedP\f\xb3N\a\xae\\-
\\\xe1y\v\xd7\xf9\x86\u075c\xd1\x05\x06\xdds8}\xecQ\x9c9\xf6\bV\u039f@\u007fe\x1ey\u007f\t\xd6\x1aW\xf0U\x13*k\xa21\xbe\x01\u0371\x0e\xa6\xb7\\\x84\u037b\xf6b\xd3\xce\xcb0\xb6a3\x1a\xe

d\x0e\xac1(z]\xf7\x1e5\x04l\xe1\xc3:l}\x98\xbb:\xc9))\xf4\xfe6\xba,0\u007f\xf2\x80\xeb\u02b3\xa6S
\x89Z\x1d\xbdc\xb4\xd6\u0632e\x0e\x13\x13\x13\xfbo\xb9\xe5uw\x01\xc0\xa5\x97\xee^\xef\xcc\u
05cf\v\xe3`f|\xe8C\x1f
\x00\xf8W\xff\xea_\xfe\xe4\x93O>\xb9\xf7\u0739\x054\x1aM\x05\x12\xae\n\x03Q\xbe\u0759\u070
c=\xaf|;\xa4j\xa2,\xfa5\x17\xabp\u0344n>:\x0f\n\x17\xc5\x05\"\xa0\xe1\xbejT\xb4\t\xe5\xfc7\x16\xb
5\xdf\x0232\xaf.\x8c\xb8\xf7*`\xbc\x8e=\xd7\vI\xa0\xb2\r\x03\x19\xcf\x17\x0e]\xdd5\x03P\xc4P~\xe
0\xc9m\x80[\x89\"2\xc0\x1b\x1dw\xfe\xbc\b\xd8\x02\x10R\xb8\x8a\ue855\xe8\xefB\xe9yP\x05]$\x
bcm\xa4\x8cE\xae\xf4\xa7\x18\x82\xd6\u04c4{\f\xdd\xf5j\xa6\x8a{>R\x02J\x12\xa0\xdc\xc2$[\x12b
B\x82\x1b.\x9c\xc32\x83z\xda{\x98\x93\xe3\xfb\x03\xd0\xcc~qJ^_\x1b\x98\xf2\xc2u\xc1CR\\w[\x0f
n\xb1\x05\t\xef\x03n\f\x06\xbdE\xe4\xfd%t\x97N\xe2\u0729\xa7\xb1p\xea),\x9d;\x86\xbc\xbf\x04!$\
x9a\xedILo\xbe\x04Yk\x02\x8d\xf68:\u04db0>\xbb\x19\xe3s[0\xb1i\x0e\xad\x89\rh\xb4\u0690B\xc
1\f\n\xe8\xfe\xc0\xc1t\x1e\u02b3\u02a9\x88a*\xb7E\x11\xb8\xae\t\xf54\x8d\x90\x8b\x88\x94\x90(\
xfb]\fV\x16\x9c\xddB\xd6\xf0\xe2(\x1bEf\u030c\x99\x99Mx\xfd\xeb_\xf7\x9d/\u007f\xf9\xcf\x00\x00
\xb7\xde\xfa\x86\xf5\xce|\xfd\xb80\x8e\xaf~\xf5\xcf\xc4\xe7>\xf7\x05\xc3\xcc3\x1f\xf8\xc0\xed?\xfb
\xdcs\x87!\xa5t\xad\x91\xc7\n\x9dO\x87\x01\x83\xb1\xf3\xf2\xd7b\xeb\xaeWE'\xbf:\xd4\xe1\xbaRA
\xd2\x0fL\xfd\xc6\xdb\xc3+\x9cB,i?#\\A\x87\xf7\u0486e\b\x16\xc8\x04P\x94\xb6\x8a.C\xbd\v\xac\x
b3X\x86\v\xf6\xa8\xf8\x9e\x14?\xe6\u06b6\xbb\x86\xf9\x13 \x84\xf7\x0fo\x00\u0726J\xab\x13
\x8fP\x14\xc6\x00a\b\xb4L\b\xc4\x1e\x04\x9ae-\x0e\x82W[\x9b\xd30F>\xb4
\xd10\xb0N5f\b\rC\xed\xb12U\xab\x9f
89z\x80X\x1a\x02\xd4\x11\xee}\xb0\x8efI\x1d\xe9\x1ds\r\xc0\x06V\xfb\x9cSE\xd0\xc6\x11:\"\x84\"
$B\xb5w6\xb4aA\xb2\xa8\xf3\x87\b\xa4$L\x99cq\xfe9,\x9e;\x823\xc7\x1e\xc1\xe2\xd9gQ\f\x96\xa
1\x8b>\x88\b\xad\xb1\r\x98\x9e\u074d\x89\r\xdb19s\x11&gw\xa1=\xb5\t\xcd\xf16\xb2\x8e\x82\x1c\
xcf \x9a\xca-
\x1a\xd6\xc2\xda\x12\xe5\xa0\v\u03ad\xe3\xbb3`u\x12&\xad\x84O|\xaaXU\x11+G\x02\xbf\x90\x83
\xc6\u020b\x94(\u0495\xfc\x1cDkg\xfef\xc3\x02l\xd1\xe9\x8ca\u02d6-
\xa7?\xf2\x91_\xb9\xeb\x1f\xff\u33fdd\xaf\xf9\xf5b\xfe\xbf\xe8\xf1\xd0C\x0f)\x00\xff\x1f{o\x1ak\xd9
u\x9d\x89}k\xed}\u03bdo\xac\xc7*\xb2XU,\x92\")Q\xd4\u040eE[\x92GIv
\xc5v\xcb\xee\x8e\xed\xb4\xe1\xb8\xe18v\x8c\xfe\xd5\b\x108H#\x06\xd2H\xfe\x04n\xa3\xff\x04H\x
1b\xe8\xd8\b:A#\xed\x8ch\xb8\xd1vK\xeeA\xb2-
Y\x94<\xc9\x16\xe7\xa9X\xa48W\xd5\x1b\xefp\xce\xde+?\xd6\x1e\xcf=\xaf\xa8?,J\uaec1G\xbeW\
xf5\xea\xbd\xfb\xce;g\xed\xb5\xbf\xf5\r\xcb\xdf\xfc\xcd\u007f\xfc\xd1\x1b7n|\xea\u06b5kh\xdb\t\xe
b\x8d\u0369\xb8\x88w\xd8\u07bb\x80\xf7~\xf7\xdfD;\xdd\xc1r~\x98\vbY\b\x99\xf5\x81\x8f\x10\v)F\
xab\xc5\x04\x90\x86\xebf5V\"\x03\u040e\x85\xf49\x1f\xd4B\xedG\xbb\x90\xbc^\x0e\xcfV(\x84i\xf0\
x9a\a\x82\xe5\xc00\xe7\xc1Q\xfd\xba\x87\xfe'\x00\x98\x04lD\xef\xfa\xa9n@\xa7A#D\x00m\t\f\x04r
\f\xf4\xc1\x00\x8a\xca\xd3\x01\xc9\n\xf7;\xa6\x1d\t\x95HF1<-
R\x84ra\xafs8+\x9c\xbc\n\x80\xce\xc1\xc4\xd62\xd8\xe8\x06\u028d\xc2+\x98h\x91J\xa7\x83P\xd0
\x01\xc5\xcf)\xf8\x92\x13ge\xae\x16\xbe\x12_\x96\x14\f\xa1\xd7Y\v\xa5\xf8
\x02b\x1df\xbev\xf5/\xf0\xf8\x9f\xfc\xbf8\xb8\xfeu\xb8~\x01\x88\xc7ts\x0f\xb7\xdf\xf5~\xdc~\xe1!\u
c77f\x0f\xbbg\xef\xc6\xc6\xf69e\xdb@@-\xd0l2\xcc\x16\x03\r
}\xa7\x18}\xe2\x9dS\x86\x92\xa2A\x1a\x91\xdaB0\x81\xac(\x05\xb1\x18\xb2f\xedZ\x91\x10U\xda
3a\xba\xb1\x8d\xe9\xe6^\xf0\xac\xe9\xe1\\p\xe7'\x86x`\xef\xcc\x19\xdc}\xf7\xe5\xafN\xa7[\x8f\xe8I
\xf6\u007f\xc0\xaf\xfe\xea\u007f\xb7.\xe6\xeb\xf5\u03af\xaf|\xe5\x11|\xf7w\u007fd\t\x00_\xfe\xf2W

>\xf1\xf4\x99\xea\x13\xc8\x00\x00
\x00IDAT3\u03c0\x98\x05\x14x\xe5D\xea\x9c\xe7\x1d\x98\x19\xef\xfb\xf0O\xe2\xe2}\x0f\xa3\xeffy
\x00V\x15r\xed\xdc\xc8T\x92C\x18\x0e\u0130\x86A\rUP+S\xd1\xcd6\x04:cUH\x14\xe4\xe2\x16\x0
4\x0f\xc0\xf5\xa8\a\xa2\xc3\xc6{4_ndh\x1a\xe5\xf9\x05\x8e-
\xf5\xe1B)\u0261\x90\u01ee|\x85\xdaX82\x1a\x06h[1\x12\xdaw\xe8\x8d\xc2\x11zB/B\x1a\u00a0\x
14\xa1\b\r\x1b\xf6\x8c\x9bS\xfawCX%\xe7\xaaJ\xadlM\x15?\x98\u007f\u015f\x83\x94\x92H\xb1\x
90o\xf0hl\xaaX\x00\x9b\f8\x06y\x17NC\xf1D\xe5\xd5\xe1WrPC\xe5\x99P\x9c\x1c\x88c\xf8\a\xc1\v
\xe0\x9c\x83\xb1\x13\xdcv\xc7}\xd8\u0739\x03\x1b[g\xb1w\xfb}\xb8\xfd\xd2C\xd8\u07bb\xa0\x9f\x
e7z8qp\xbd\a,\xd0Z\xa3\x8c'\x13\xbdd2K\x88\r\xeb}\xd4\v\xa4\xf3\x05\xaf\x9e5\xb8\"\xa41I\f:\x19\
xccLjQ\x1b\u00acD?\xb6\xed\x14g\xef\xbc\x1f/>\xf3\b\xfa~\xa1\x0e\x8c\xc6(\x13\xcb\x1ax\x11\x1
8c>\x1f\xbf\xde\x0f\xfd\xd0\x0f\xad;\xf3\xf5\xfa\xe6X_\xfc\xe2\x17\t\x80<\xfa\xe8\xd7>\xf8\x0f\xfe\x
c1\xaf\xfd\x9d\x97^|\t\xd64\x94\xbc\x9d\x83
\xa4\xef\xe6\xb8\xe7\xa1\xef\xc7C\x1f\xf9\x8fam\xabX\xf9\x00\xd0
\"\x80M\xa2qA\x02\x9b\u00c4\x87\xc5\x10\xa4e\x90\xd1b\xee\x11\xc2p8\x1c\x00$\xccD\xa7\f9cA
\u0483\xe6\xc5@\x14\x8a\xdf\xc2K\xee\xfa\v||\x05J\x11\x19M\x9a\x1b2\u05c7\xaaO\"e\xaf\xc0@\
xa9\x93v\xc0F\x17\xacl\x1a\x04Q!\u0536\x9ao\x91\xf3\xe8\xbd\xd2\"\xc5\xc5\xf4$\xaa\x10\x15B\x
e9\xa8(\x15B2\xf4a_\xd9\xc0bT\x1c\n\x91\x10\xd5\x1b\x9b\tTP\nb-
LI\x8b5\u05db_4\x10\x13\xaf\xbf\x00\xda4\xba\x19\xf4\xda\xf12\v`\u0507\xdd;$.}}\xa4
T\xd9\x12\x1e\x1aDa\f.\xdc\xfb\x9d\xd8\u067b\b\xb6\r\xa6\x9bg\xd0Nwa\x9b\t\xbc\xeb\xd1-
\xb3\xa0\u01c7\x13L\xd32\u0314AS\xbdW\xe0\xc3=\x02B\xa1A\x03&\x16\xd2\xf5jY\x1c\x94O\x86
\xd4A\xd1#\xde\\a\x04\x1cB\xc9K\x83\xaf\x94PT\x84\x8e\x13[\x9c\xbf\xfb\x03h'[X\xce\x0f\xe1\xd9
\x06\x91\x11\x81\x8d\x85\xf7\x82\x83\x83\x83\x97\xe2u\xfe\xbe\xef\xfb\xc1u1_\xafo\x8e\xf5w\xff\x
ee\u007f)\x00\xf0{\xbf\xf7{\xdf\xfb\xd8c\x8f\xefz/`c\x82%\xa9\x16\n\xd7-
\xb0\xb1s\x0e\x1f\xf8\xfe\x9f\u0159\xdb\xefA7;JY\xa0\x89\xb1\x1d:\x1b\xf5\x9b6\xe9A\xf7&C,b\t\x
98P1\xac\x8bc\xb4\xc21/\u05ba-
\xa3\xb8\xb9\xd3\x02\xce\x00\x1aa,{\x9f\xf5\x9c\xd1#DFb\xd9N\xe9\xcaSP\u0100\xa0Q\xd6H\x06
\x94\xb52!\u0224TK\x16\xa2\xf3\x01\xa1\xa6\x82+\xb6\t\xa6S\x0fqu\xc7\xd59\x82/\xbdfB\x11A\xc
1\xd7N\xe3N\xaa\x1b\xffj\xc7\xc9f39^u\xf8\x93\x87\xc2jX\x8b\x1f\x19\x807\x8c\xce$\xa2\xc7L\x04
g\x06\xf2\u007f\x91\x00-
mi\x87\x8ec\ar\u0280i\xc0\xf0,\xf0NB\xfe\x85$oxfJz\x83d\xe4\x15\xbe^;\xddQ\xa5\xb0\bDz\x888t\
x8b\xe3\xe2\x17NI9k[\x86\x9d2h\x83A\x96C\xe0rqc\x14\u061c\x18\x81\x01\x01\xc7\x0e\xe8B\x18
8\xeb\xafN\x85gT\x18\x8a\xd1\n\x8bI7n^\xb1\xcc\u07fe\xed\x12\xb6\xf7.\xe0\xe8\xc6\u02da\x9c\x
14EV\xe1\xb3\x1ak7\xbe\u055f\xfb\xb5\x05\xee\xb7\xd9\xfa\x97\xff\xf2_\x84\x1a$;W\xae\\\xf9/\x9
ey\xe6Y\xbd\xb9\xd9\x14\x93\u007f\xf5\xa5x\xe0\xe1\x1f\xc1\xbb>\xf0\t\xb8\xe5|0\x95C\xe2iG\u0
447\xb2XD\xbd\xa8\r\xa5\xc1\x1b5\x19{.\xc2\xe5*\xa8\xa1\u00a1\xb7-
h\u01c0\xac&\xd3Y\xab\xf8\xafX\x827j\a\x10\x1d\nW:p\xa9;\xefa\xd7.9]l\xd0\xd5\v\x98Dq\xda-
e\u007f$8Z\xe4-
\x1d8\b\x00M\rh\x87aZ\x85_\x8c\x91\xa0\x9f\na\n\xac4F\x89S\xc9\xf25\x0f\x8d\xbeb\xa8\x03\r\xa
f7F\x15\xf1\xf1\xd3\f\x87!3\x11\xa8a\u040e\x05\xb5\\\\\u3890\x17j\u0304E\xb7\f\u07b1\xa0M\x03\
xb2a\x98\xc9\xfa\xc6Q\xf8\x15m\x84\x933\x1a\xd5v\xbe\x01\xaf\x12\xf1\xe8\x963\xf4}Td\x02d\x0
3\x14\x17\x18NB\x80i\x18\u0366\x85\u0659\xc0n4`\u00ea\x06\xb5\xe1\xcdpbIqp3\xa4\xa9\x05m5
@\xcb\bLX4\x86\xd0X}\x9d\xe9\x0f\x83N\xa2\xcc\xe9L\x14ZP\x8e\xad\x13\x81\xb5-

vo\xbf\f\xa2`\xc2\x15\xec+z\ufc71\xb9\x81\xcbw_\xfe\xd1\xf85\x9ex\xe2\xd1ug\xbe^\xef\xfc\xfa\xb1\x1f\xfb\xb4\x00\xc0o\xfc\xc6\xff\xfc\x1d\x8f>\xfa\xe8G\xe6\xf3\x85\xe2\u0764Gj\x88G\xbf\x9c\xe3\xf6w}\x10\xef\xff\xf8\u007f\x8a\xb6\xdd\xc2\xe2x?\xc9\xfa3\x9dC\x1f\x06f\vf\x9b;\xd8\u062d\x87B -
\xd7\xee}A\x15\xba\x02\x8f\xc4V\xd9\x00\xb4\u06c0;\x81\x1c\xf5\x00\x01\r\x03\xbd#t^s;\x99bp\xef@\x918\xa8v\xa9\v.\xc4D\xa5\xadJzM\xa4\\k\x9a\x12dR@\"e;\xef\a5\xbdpK\x04\xd4\bJ6\f\xa4\xf3`q\xc0\x12\x89\xfb\xdd\u01df\xcd\xc7\xe0\x03_\xbd\xbeX\\\x93;a9\xa0\xa3\u06b7\x86V\xc0{\xfd{f\xe8\xc0\x93\x01\xb2\x04l[`\x83\xf3\xcc`\xe0\u03d2,y9t\xdaqc\xdd\b\xf7\x02\x11p\u0483:\xc0\x04\xa1\x97x\x01;\xc0\xc5{\x05\xc55\xaa\xc2@\x94\x11\x14y\xde\xe5\xe7\x88d\aCb\xc0n2\xecn\x03\u07b6\x01v\xf3\xe9s*\x9c;\xfe\x9c\xf1\xf7\xb8\xc1)@\x9c;\xbdn\r1:\x84\xccQ\xcf\xf5\xb6^\xbc\xbe\xa4\x04-
~\x99l,6\xb7\xcf\xc6<\xbb\xf0m\x19\x86\x8d,\x17Kz\xfe\xf9+\xf7\xc5\xcf~\xef{\u07ff.\xe6\xeb\xf5\u0373n\xdc\xd8\xff\xf9g\x9f}\x0e\xbd\xf3`\x13\x8a\xb1wp\xdd\x12\xa6\x9d\xe0\x81\xef\xf94\xce\xdf\xf3\xd7\xd0]?\xc8~\xe65X\x1e:&[\x17\xbd\xb2\xe0\x9b\u0615S\xf2<bR\x15b\xb6?\x1f\xd0\x15=@-\x01{\x8d\xd2\x15O\x1c\x98\b\xad\x12\xe8\xe0\xbcv\xd1e\x1aN\xe9\x12\xb8\x12\xe1F\x85\xd0\x05\xb9(\u050dd\xe8\xca7\x90\xbar\xfd\x11\xe4\xf4\xa6|$\u308d\x81l:H\xef5W\x14\bC<\xc0\xa5\xc0k\x19\n+Q\x99\x98\xa4bB#\x93WZIgC(\x8a&\xce(X\a\x9e\xbcm\U000b597cAD\xf5\xa3G\xbd\xd1\xe5\u0670\x80\xa6\x06\x1c\n6\xc1\xab3d:-
\x10\xc8\x01\xceyxY}\x99\xb5\x0fM\xf8\x1e\x1c\xac\x94\xa3\x8bd\x80?LK\xb0\x9b\x16\xbci\xc1\x8d\t\nS\x14\x02\xab\xbc\x19\x94\x9e\xfa\xf1g\xe2\x8d\xd0\xe1\x1f\xf7\x9a{\xea\x152\xf7\xac\xdf+\r:#C(\xb8v\x0e\xdc\xdb\xe1\xc5\u00d0A;\xdd\x0e\f\x16\x97\xe6\n\xc6Z\xda?8\xc0\vW\xae\x9cy\xe3\xcdW\x1f\xbe\xfd\u071d\u007f\xba\x86Y\xd6\xeb\x1d_\x9f\xfd\ucfca\x0f4?\xf9\xd4S?x||\x92:l\x1f:\xa2\xbe[\u0dbb\x1f\u0103\xdf\xf3\x13\xc0\xa2\xd3\x1b;\x03\xc4\x05\x19$\x14r6\x15\xd0+\xa1\xa0p\x80Yb\x8a=\r\xa8e\xc3X\xb8\x12;\x87\x00<e\xd0\x19\x85\t@j\t\u0432\xf2\xa7}e#@5\x94]\x92-\xcaa(Q\xc6\u024bM\x89\x10\xe0\x90
j*_S\x15\xeaSvx\xa7\xe4\x93\x121\xc8\x1a\u015e7\xf4\xe9a\x12X\xf2`)\xfc\u06c7P\t\xa12,+\x1f;\x19\xc0\x03\xe5&\x15\x83\x1b\f\xe9|\x81\b\xe0\x96@;\x16hy\x00-
d\x8c\u0757f_e\x88\x04EN6@\x13\x06\xef\xb4\xc0\x94\x03\xe4\x15\u0330\xc2\xef\xd6\x04\bD\x8f\x04\xfa\x96r;cV]\x80\xef\x925\x04\xe7\x8fMc`7,x\xab\x01\xb56)\x87\x13\u0155\xb9\xc6\u0283\xc3c\xfa\xb9)@0\x1b\x06\xb4\u0640\xad\x81\xb1\f6\n\xc7\xc4\xe8\xb9\xe1p<\x8a\x9aT\fW\xd8\xf7\x82`L\x9bN\xa0\x89E\xc3\x06\xcbE\x87\xe3\xe3\xe3\xf3\x9f\xfd\xccg?\xb2\xc6\xcc\xd7\xeb\x9bb}\xf2\x93\x9a]\xf8\x97_\xfd\xb3w]y\xfe\xf9w-
\x96]!\xae\x10\xb8\xbe\x83i'\xb8\xe7\xe1\x1f\u0199\xdb.C\x16\xf3\xccb\x18\x14\x126\x06ll]d\x8abI\x1c\xba\\\xa3\x85\x86\x8b\xbc\x8a\x8cs\x94\x82}AM\xd2\x06h\u04c2\xb6\x8d\xb22\x88\x14\x17\r\"$g\x02k\x86\x8atw\u0501\xd0BE\xcd\x1a\xc3\xd0\xe3\xa7F\x06KS\x1f0\xe4\xe6)n\xa3x\xbd\x16t\x026\t4\tu\x8e\x04\x86\xfc\u02b06a\xe2\x05\x9c\xa0X\xb2\xa4\xacO\x12*\xb0qZ\x11Gq\xf4^\x89FZ\x9bV\a\x89\xf1$P\xfdz\x02C#}\xa5\xe8\xac\x18_{\x1e\xae\x82\x00\xda`\xf0\x96\x05\xb5F\xf1k\u03b3\x106\x14\xe0\x97\xf0\xcb5\x8c\xf4BH\v*\aK\x88\x14\xb9\xcc\f2\x06\xc6\x1a4\x13\v\xb3\xd9\xc0l\xd80|\xcf\u007f\x1fo\x16b\x0e\xf1r\\\x17\xf1P\xe8%Z\x18l\x06\x8b\x82\xf0\x12\xac\r\xc1D\x91\xdaH\xa5\x1dqpk\x1c\xf0\xf9!\x9a\x0f\x1a\xff4\xde*\u030c\xdey\xf4\xde7\xcf?\xff\xc2G\xbf\x95\x9f\xff5\xcc\xf2m\xb8\xfe\xe8\v_\xfc\xfe\x97_~e\xbaXh\x8a\x8a\x84\xecH\xd7/\xb1s\xe12\xee\xfd\xceO\x80\xe6\x0e\u03b9\xcawE\x12\xa7<;\xca\r\x1a\xdd\x02/\x0f\xbc`\x1at\xddT\x97RZ\xa9\x94\x8

9\x12\x016\xa4\x96\xb9\x9d@\x8e<X\x80\u01aa\x19T\x1fC!\xdcX
\xe80`\xb9\x00*\xa8B\x8a\xc0\x10P+\xca+\u720d\x13\"\x1a$\xa5\x0f\x8a\xd4\x0e\xb6+\xdf\x0eH\x91c\xd2x\xc8V\x10=-
\xa0\xb4GR\xb8\xa58\xe8h\xd5q\x92\x82\x84\x13\xb52\xa1S2j\xa1\xab\xff4\xe6f\x86a\xf3\u0504\u036f~q\xa5\x97x4\xc0*\x8d\u0232QV\xe8\xe0#\x0ee\x00\u06b2\x80\x10\xfc\x89\x03z\u0162M\xf4<w:I\ue4f5{\xb4\xfd\x95!\x8b?\xfd<\x86\x01\u06f0Z\vl\xea\u026b\xa2\x8d\x06!R\u029a\xa5l\xa2V\u0307Wn\x19\xda
\x88c\xb0\xf8`\xf5\v8O\x9aGZ\xde\x04!l\x9c\x06\x9e\xc7\"\x1e\xce\xf5\xa9\u0448\xb4Q\x02\x81\r\xcb|>\xa7g\x9ey\xfa\uce98\xaf\xd77\xd5z\u9957\xbe\u007f\xd9\xf5\u4703\xb5MrA\x04\x11v/\xbc\v{g\xee\x06\xf5.p\xbb\aC+\x10\x88,\x88l\xfd0\x84NX\x8a\x937l\u044d\x97\x14\xc1\xdac0\xb3\"D\x8a\xce=\xc0\x11\xad\x01\x9d\x01\xc8\xf5\xc0\xdc\xc10\u0436\x04\xdf\t\x9c\v\xc5\xcc\xc5H\xb3REIE\x1d\x91\xaa\xb0\u01d7\u0361+\x97i\x80YJ\xba\xa3\xc4\x01\xe5i\x8d\xb8r\x9b\x87S\xd1\x14\xceA\x81\x92\xe9(\xe1\xb9`\t\\\xec\x02\xad\xa6\\\xd0\v)j5\xac\xab$\xb0\x85\xf0\x89)\x0e\x94\xa1*\xcf-
\x03\x9a\x98\xf1}\xad\x84\x8f\n\xd7\xc0\xe0\xbb0\x88p\xe3\xfc\xc9\r\x01;z\xba\x92yNZ\x82S\n)\\\xf4r\xa7`\xc7
u\xb5\xcd\\J\x18#\xb0\x96\xc0m(\xe4\x1bF\xf5\b%\x9eE\u0671\xb1\xe2\xe1\x17\x1b\xbfO\x03J\xaa\u007f\x86\xcd\xf0\xfe\u072b\xc8\xc8\n\xd0{8W'\x1f\u9f55sGc\xb7\x1e=g\xe25\x97p\x0fXki\xff\xc6\r\\\xbfv\xfd\x9c\x88\x9c!\xa2\xfd\x17_\xbc\x82\u02d7\xef]\xc3,\xebu\xeb\xd7s\xcf=\x93\xde\u007f\xf5\xd5\u05fe\xcby\xa78y\xf4\xb6\xf0\x0ev2\u0145{\xbf\x03\x1b\xb2\tY\xf6%\x18\x8c\xa8\xf2ccal\x9b\x02(\u02beZ\xe2CK\x81Qa9w\x81\xc3\xe69=\x9a\x94\\\xaeJL=w`\x02l\x18p\xc0\u03c95~nbT\xae\xee,\xc17\x04_\bKd\xb8I\f\x11\x96\x88Q\x13@-
TTc\xb0\x12G\x97\xbc\xba\xc3?\x12\x1a\xd6\xc6\xf1b\x1f\xad\x83\x01(\xa3d\x8b\x14w6\x04k\x04\xcc\xd9\xd8K\xaa\xe8\xf9A\xcb_2:\x06\x17\x90\x89`\x8cZ\xb4\x92e`\xd3\xd4\xe2 \xe44\xfb\f\xe7\xd1\xf1rVk\xf8\x1a\xc4a\xf3\xaa\xbf?\x91\u0489h\xbb\x01O\x03\xb5\xd00\xb810\xad\x01O\x14\xff&\x13\xb2D9JP9\xd3\x14I4\xe8\x83\x19l\rx\x83A\x9b\xac\xf3\x14\x19$TQ~\xbf|\x8b\x94\xc5\xfcu\x91\x86\xa2\x11\u07e6hO0Q\xe1\x11\xd9p3F(\xa8\xbc\xc4\x15\xa1\x8a\xaa\xa0\xf1\xca\xd0\rj\xc6utt\fb\xbe\x0f\xc0\x87\x00\xe0\xf5\xd7\u07e05f\xbe^\xef\u023a\xef\xbe\abW\xb6\xf7\xc2\vW\u03dc\x1c\x9f\x80\xd9$:\x9c\x17\x8fv\xba\x85\xdb\xefx\x00\u0187\x89>\xd5\xdcp\xf5\xe0j@l\u01cbL\xc0\xaf\x99\xa2\xb1\xd6*\xfb\xf0\xadJ!\xd5\xe54Uu\xda2\xcaA\xb7Z\x9ct \xaa/\xc3\xd9\x10\x18]
\u00eb\x1b\x87\xac\xf6\xd6F@\x93\x80\x95\xaf\xee3yc\x10\xacJ\xe8k\u06eb\xb1\v\x11pb@6\t\x98\x86\x82\xde\x00l%3z\xa2\xb3\xdf\n!>^\x02Z\xb9*\x89SN\x01^\x99\x04\u0738\xc9hx\xe5\xea\x1d\x8bc(\xe4\x1c\xb0i-
\x949\xc0\x99\xaa\x01c$4\x86\xdf\xfd4\\\xff\x86\x03\xb4\x13\xe06k\x14S7&\xa8\x81\x8b\x90\x0f\xa1\x84n\x18CZ\x90\x1bV\\\u007fb\xd2\xd03\xfb\xa5P5\xac\x15p\xa0\xbf\x9a\xb4A\x12\x13L\x84\xb2b\xfaR\xe9\xb7b\t\xb4\x11
>\xa3\x9b\xbcg\xe49Da\xc2\xe5\v\x8fx\xf5{\xf1\x15\xbb2\x0f\xe7\x19^\x04'"\xf4\xf5\xaf_em\x88^]\xc3,\xeb\xf5\u03ae\xc7\x1e\xfb\u068f\xccf\xb3\x8bG\x87G`2\x15M\xae\x99la{\xe7\x02\x98\fzY\x1d\xd61\x99@c\xc4j\xe4\x19B\xdeg\x1c~6\x14\x98\rC|\x93*\xc9=Um8*\v\xaf\xe8\xf3\xa1\xd1g\x02\xda5
\xe7A\a\x02t\x82V\b\x8e\x80\xa5W\u0699\x0f\x18xY\u0487q\xc9\t\x8ae\xa8\xa9V\x18|\u0280\x8e<\nU\u0230\u04e7U*\xe4\xe04\x03\x0f\xc0\x00\xb2\x05\x90\xf3\x80\a\x8c\xd5\xef\u0457\xedy\x8

4#\xdc\xc0\x18jU+\x14\nr\xf0'o\t\xbc\xd5@6m\xf2G\xc9\xf8u\x86L\xa8\x18\xb4f\xd8K9\xe6\xc3@'\
xa9\xcfMH\x86/\x9b:\x90\x95\xc3^\xed\n\x90\x19C\u0300'V\x8f\x14\x93\xbd\xcdI\bM\fy0\xca\xc5\x
f7\x1bV9\xf1\xc1\xdd\x11)\x12/\x02!A#\xccT\xec\u0152](\x83\t\"\x93\xfa\xf8\x88\x1f\xd0=\x1b\x06
m\b\xfa\xce\u00c5\xfb2\xfd\xb6$x\xd8H\u212aj\x19\x9a\u16b7NN\xf7\xae\u0386\x18\xf3\xf9\xdc_\
xbf~\xc3\x01\xc0\xfe\xfe\xfe\xba\x98\xaf\xd7;\xbb>\xf3\x99\xcfto\xbc\xf9\x86O\xe6LI\xfc\xc1\x98n\
xecbs\xfb\x1c\x04\x94p\xc9\u071eS\u8fb8\xe0\xe9\x16\x85<(\xfa\x88C\x10\x82\xe5\x04\xfd\xd2J\
xdfM+\x01\xbe\xa9#\xa5<\x1a\x95\xd2\xdeV\xb4H\u040e\x05\xf7\x02\u007f\xd4\x03
L\x04\xf0^\x14r\x8e\xaf!\x16\x04\xa2\x15\xe7\xd9\n/\xb7\x9a\"T>\xd72R\x98\xa5\b!\x93A\a\x1f\xf3\
x96\xb8\xf4&g\x02\xb3\x81\xa7\x10\xec,\xba\xb9\xc9V\x18v\xf6\x801\x1e\xde3\\\fS\xe0\\\xf8U7_\b
\x8a\x90\xf3,\x05\x019\xe0\xe0\x86\xb8\x99;f\n\x1bB\x19\ueb1b'UB\x19*\x87\x9c\x83\xbc\xeb\xf2\
xf7J$U:\x121\x03\xdb\x04\x11\x86\x9c8\xf8N@^\a\u04d0\u0e98X+jUkC\xd2\x14\f\x01S\x03\xbf\x
d3\xc0Y5\u01f2\u0308\xe1@\xbe\b\xf3H\xf4H\xa2\xc2D,w\xe1\xe2)Q
\xc5\xfb\xc2q2of\xae!,\x1a\x86\xef\x04\x86\x00\x98`\xb3\xdc\xfb\x94\xad\x9a\xad\x90\x87\x16\x00
\xf1\u0102\x90\x96\xc5pT\xf8\xd9\u007f\x8b\xaeu1\xff6[\u05ee]\xbb\xec\x9d\xdfr\xce)%,\x0e\xc4\x
d8`\xeb\xccyL&\u06c5\xaa\xa6\x18>E\xc9?h\f3Q8\x81\x00K\x8aU\x92-
\xfcP\xca\u007fB\x92\x1c\x04W\x02%GQ\xc8:\x81\x81&\x04\u0675\xa0Nt
\ua056\t\v\tG\xe4\xb1/]\xd2\x16C\x81#\x16\xd0\x14\xc0d\xe0\x0f.uC\x9e\xde$\xabH#\xc9\xd0\b%\
x9c\x99\xc8\xe7\xa86
][\xefz\xc0\xa9<\x1c\x13\x026\x058Vq\x8b1\x02/\x9c\xaf\x9f\xc4\r\x82s\u0403\x97\xca\x13\x87)@\
x16-
\xc1l\x1a`\xa7Qs\xaa\xb2\xe8\xca`JP2\x8a\n+\x81X\u0529\xa2\xeaQ\xb1\u007fsm\xbb@\xba)\xd1
n\xf8\x1d\xfa\x1e\x0e\xac!\xd7>\xc0\"\xb1\u007f&\x85UR\xfek\xc3\xc0N\x03lZ,\xc2\xe7\x9b
\xed\xd7Z+\xd9Q2\xfe\"\x88rP4\xfb\xc0Z\xa5\xa0\fR2M2{C\xc1Z\n\f\x16\xde`\x18\x11\xd0\"\xdc\x1
b\xbe\xb8\a\x92\xd9X\x0e\xa2\x06e(\x87\x8a\x98\xb9H\x87d\xc3h\x1b\xc5\u4b35\xebb\xbe^\xef\x
ecj\xdb\xf6\xfb\u06b65}\u05e15&\r\x19\xd9\x18\xec\xec]\x80m\xa6)\x99\xbc\xc4Q#\x15QF\xea,R\
xd2{\xe8\x1a\x83\x8fv\xd5\xecq\x8d\a\x8fR\xd7V\xb6\x8a\xf0\xa0E\xf5_(8\xbc\xc1\xf0g\xacv\xb2\v
\x0fk\xd4\u072b\x8b\xec\x12\xa6D\x8e`PEnI\xf4?\xab\x99\x9ed\u01af\x93\x14
\x83T\xddyP\xe6\x83
\xa1S\xa60\xc43\xe5\xcfE\x04\xb6\x16d\x18\xd2;x\xd7\xc3;\xa71t^\x80c\x82a\x817\x02\xe79\f\xf1\
xd43\xc0s\xe0f{\x0fr\xb1#\u0542n\r\xa3i\x18\xbc\xc1\xe03\x16\xbc\xdb\xe8\xbc
\x9ap\x15\xfcI*v\u04c8E3\xd5\x13\t\x891uq\x00(\x12h\xdeC9g\xf1\xab\x9a\x10d\x97!\x0e\xe0\x99\
x03\xbc\x0f8|d\xbaxx\xca\xcc\x1e\xa5MZ\xf0N\v\xb1\xa4\x8c\x17P\x10\xf6Pm\x89<<\xaf\x89@\\\x
1f\xd8'\x94n&a\xc9x\u007fiw+y\xd87\x99\x90n6\u0783:=\xec\xa4\xcfd^\xb1\x1b\x8a\xf1v\xe5\xecE
@\xc9\x17\x06\x02\xf4\xce\xe9&\xed\xfd\xba\x98\xaf\xd7;\xbb\xdex\u336d\xae[\xe6\x1b?\xdd\xd0\
x06\x1b;\xe7\xc0\xa6I7j%V\xa1<8\x1a>\xdc
\xc03\xa5\x00g\xb2\x94\n0\x15\xc1\xc9\x15\xd7;\xcd6\x87A\xc0\xa3\x8ft\xe6\x1b\u01c7u\xdb\xc0\
xf7\x1e\xf0\x1e\xa6#\xb4\x81M\xd1G\xfc4\f\xf1\xca\xf9\xa1\x14\xf8\x88L\x83|\x1f\u046av\xc0,\t\x0
f\xb3:\x05ReEK\x10\x98\xd0\xfe\u01df'\b\xdf\xf5H_\x15\xd40\xf43\f\xea\x1c\xbc\xef![N\xa9}3\r\xb7
p\x88~\xe7\x04\x11\x82k\x00!F\xb3\xa0\xe4?\xa3\xc3F\x86\xb5\f;50\xb75\xa0\xbdF;^\x19\x04\x8a
V]y\x8e\xf5\xe3\xf0;\xf4\xa21\xda\x10\xc0;\x1f\x92|\xbcF\xa5En7\x97G\x15Z\x9ddo\x1a\x18a\x00
K\u0639\x83a\r\x87\x96^\u0099\xc2k\xd7m\x83\xe9\xd7n\x03\x99X\xb0al\x98\xb0\xf9\x15\xc3Wp
6\xf2RA\x1aC\x9c\x87\xeb:\xf4\v\xc0K\x9f|\xed\x01\xa3\xd4W\xf6\xf0\xec\x01'up\x93\xa4\xfd\x1c\x

d2\x12\xfc\x86\x81\xb8\x1eB:\fe\x9fKu\x05c\x19S\xb8K\x04\x8fv\x11\x90\xa8+\xe8b\xb9\xc4k\xaf\xbd\x06\x00\xb8~\xfd\xfa\xba\x98\xaf\xd7;\xbb\xae\\y\xa1\x9b\xcd\xe6
c\xb2\x98\"\x14\x8b\xe9\xf6m0\x8dM\x8eq\ts-
\x1b\xe3T\xa4\x8b\xee\x95\x011*\b\xe1\xe8\xc7\xc2\xf5\\\x8f\xa2\x87t,:e2\x83\fl\f\xa5<\x0e\x0f\a\x91\x94\x9eV\xda\r\xdd\xf9A\x0f\xb6\x04\xdb\x10\xbc\x10\xbc\x93\xdc\xd1sV6\xa6\x8e\xb3!\xd0.@\xad\x9a\u056e\x84^\f\xe4\xf6\f\t\x12\xf8:\x9b\xb3\xe6\x8c\xc8\xf8n\x15?\xd7\x18\x10\x1b\xb0cx\xeb \xe4\xd4\xfeu!\x80c8\xd1\v\xe6\xa1\xd4E/\x00,\xa7\x94$b\x8bfb\xd0N-
\xec^\x03>\u05c2[^y\xad\xd5V\x1bX2D\x92\xe4\xfe\x01\x91Nv0\xe2<|\xef@\xceA\u0125\x81\xacZ\xd3\u02c8\xd50EZ7\xb0c\x000x\xbf\x83_zH/AD$
\x16\x9de\x18V\x13\xad\x9d\x06\bE\xdaZe\xa8\xb0QX\u00c7MS\x88\xf3\u07c7\x1d\x9e\xfb^}\xf0\xbd\x878W\xa8\x889X6\x0f:\xf3\xf8\u07c2\xa7\xefZ\u00b2!\x15=\x05\x02\x8d\x06\x92K\x1a\u019b\u05a0\x99n\xa9E\x05\x96\x85\x1f\f\x87a\xb9\xc5l\xb6\xc0\xd3O=\x05\x00\xb8z\xf5\xc5u1_\xaf[\xbf\x8e\x8en`{{\x0f\x00pxx8Y,\x16\xc9\xfa3\x1d\\\xd9`\xb2\xb5\v2\r|\xe7R\xe1\xaelk%&\xf6He1\xeb\rrWn\xb4=MRy\x1a\xcf\xdb\x04rr\xa4\x10*E\"\xc6R\x85\xa4,\x8e1}\x99\x81]\x8d\x9cC\x0fp\xaf\xa2\x94N$\x85\x1e\x10)\xe4c\x88B\xbe\xa7\xe6\x8e\u04b6W\xc1P\x14\xf3T\x91?\xf5\xb9\x80\x13\b/\xc1\x17\xab\x84\x85\ua23a\b\xbbTXsa\x88E\u0702!\x90\xd6\xebP\xee\u0423\xe9\bB\x06\x10\x81\xf3:\xa8sN\u09aa\x18U\x81\x92\x01O-
\xb0\u05c0no`&\\c\u0660:\xfc\xa2\x00\xaa\xa2\x9a5\xa3\xe2\x94^c\u0083\x9dQ\xab\xda8\x88L\xd6\n\x18\x19\xfceoq\xec\x1a\b\fx\xbf\xd3a${\x15\xeb\x10\xe0\x9d\xa0\ac\xd94h\xd8b\xc2\fc-
L\xdb\xc0X\x930q&5P\xeb%\xe6MPr\x824\xb6\x81\x99\xb4p\xddB\x87\xf2\x92g/Y\xa2
\xd9\xd5R|u\xa3\x91\x10\x9c\b\x16\r\xc1t\x04\xebD!
.|\xe6\xc3\t\xaa\xd9\xdc\xc9\x16\x15\x05\x04\xc3`\x881X,\x97\xf2\xecs\xcfy\x00\xf8\xfa\xcb__\x17\xf3\xf5\xba\xf5\xeb\xf9\xe7_\bl7\xf9\xe0\xdf\xf8\x1b?q\xff\xf1\xf1\xf1@_/0\xb6A\xbb\xbd\xabFQ]_\xc3\u0525'Hx\xd0S\a\xc8HF\xfe\x86\x82\x80\x85)\x15\xb5\xe1\xfc\xb3\xf4\xe5\xa6*2\x1d\xab\xdcu\x1a\xa0\xe8\xd1\x1a\x15E\xe5l\t\x14$\xff\xd4{M\xa0q*\xe36\xa1\x80F\x95$\x99(\x80a\xf5\u0096^;P\x89\x015\x8a\x1b\xfb\x01T\xa2\x98\xb5T0\xcb\xe0\xf2\x9d\xba\x04y\xa0F!\xb0\"\xf2\xb8\xe5L\xa0\xd9\x1dQ\xb2\xa1\xd5z\xaa\f\r\u01c07\x04\x17\xd2vx\xdb\x02\xb75\x10\x9b\xb3<\x93\x87\v\rLe\x86\x1cK\xca\xd2~\x19\xd9Y\x89\x19\x863w;n>\x88CXHE\xc9D\x80\xfe\xb9\x05\xc4z\xf8n\tYv\xf0\x8d\x16Vo\x04\xd6\x03=\x04\vg1\xeb\f6\xc8`\xbbi`Mvt\xf4\x14\xe5\xff\x04\x1b\xae\xbb\xf3\x81\x9c(\xba_\xab\u0372\x81\x90\u02ef!\x89\x89\f\xbcs\xf0\xe22\xfe\x0f\xa0\xbc\xb5,\x01mC\x1a\x9d\xe7\xe3\xf0V\xed\x94)\xa9\xac\x18\xcd\xc66\xd86I,\x86\xf4;\x03\x96\xcb\x1e{{{\x97/]\xba\xf4\x9d\x00\xfe\xe0\xde{\xeeY\x17\xf3\xf5\xba\xf5+\xe2|o\xbc\xf1\xdaE\xe7\xdc^\xd7u\xa1\x13.\x86?\x93\t\x9a3\xbb@k!\xb3y\x91?\x19y{\xb9\x13\u00a0\x9e\xa6\x0e\x86D\x85B\xb6\xb6\xb6\xae\n\xb5\fA\x81S\x99\u06ab5\x92h\u0624\xeb\xc7S\x03\xd9\x16`\xe1\xc0>\x04\xfcz\xa5\xfaq\u8b09\x95\x93M\x93
\xb3gh\xaa\x0e\"\x05/|\x8b`l\xe5\xbd\u2fa9\xb8\x95\xa6\x89\x95\x81\xd7\n\xf912,\xb3\u0425\xa0\xbfQ\x12\xc4\x04\xdcgK\xb9\xe5\x98\x03\xe8\x03\x17=\xc0l\xd6k\x98\x05\x87@l\xb3caZ\u029bd\xe1s#R\xb2\xb43\xa3e%K\xf4-
\xaetR\xe1\x06Q\x90\x1eMdt@\xcca\xb0\xed\xbd@Z\xb5\x11\xe6^\xe2\x88R\xf1i\x02Z\a\xb8%\xe3\x18\x84\x1e\x0e;\x04lNM\xa2%\xe68<\n\xac\xa1B\xd0\xe3\x01O\x04\t\xc2!xWX\xff\x1a0;xb@\\\xb5\xe9\u01efG\xa4\xf3\x14\u00c0\xb7\f\x0f\xa0w\x02\xe9}\xa2gF\x05\xa9m\xa60M[\xfe\x84z\x12\x15\x0f\xd7\xfb\xe5t\xb2\xd1z/\xef\a\x006\xf6-

\xb6\xf1u1_\xaf\xb7a\xc5a\u0375k\u05fa\xf9|\xee\xca\a9\u0791\xa6ia67
\x13\x86\xb7ZX\x84Vs)s6X\x9d\xab\x99\x1a\xee
\xe7\xa6\xc1\x103\xc16\x95eG\xe9\xad1,\x16\xe3E>\u05dfh\xea\x14:\xa8-\x03,-
\xd8uh\u0200z\xafy\x96\xe1\xb5q\xb4\x9emIa\x00\f\x92|JhB\x04\xc6\x10\xc4S\xf0\xad\x11$\"z\xf5
\x1a\x86\x83G\t1zA&?`\x84\xac\xd8\u007fK\x88\xd5\xdb&\xc0yH'\xa9\vN\x9d\xa1Wk\x04\xbfe4\xc
eN\ua409lD%\x95=\x80\x14\x97\x89\x90-\xd2E\xb0\x1a\x9e:\f\xc5NB&_|-
\xca0\x93\x14\xa76\x0f\xc8\xcci\u0232!H\xe7\xe1\xbd\x16cb\r\x880N0\x99{\xcc-
0\xef\x19\xfd\xb1\xc3\xc2\x03\xdbSFk\x92\xa7b\x15=\x17\xe7\xaf\x1e\x80\xf3@\x17a\xbeP\xec!z
>\x122\xc9z7v\xf8\xd5\xcc
\x88\x8d\x98\xf4t\xc5\x1b\f:!,\xe7\xe15\x16n\x8a\xb6\x9d`\xb2\xb9\x8b\xa3k/\x17\xf7g\xc0\u052d\x
a5\u00e3c<\xf9\xd43\xc7\x000\x99L\u05dd\xf9z\xbd\x13\x98\xf9\x11\x00\xe0\u018d\x1b\xe8\xbae
8FF\xa3\xa1\x10\x1c\x11d\xfa\xde\x00\u0796E\x83\x82;\x1eV\x83$R\a(\xb9xq\x0e\x83(a\x1a\x06\
x06f\xe2\xf5\xb7\xa8+I\xc1/\x1e|>\x86\x1bD\xa8M\x86\x01\xbfc\xe1\x9d\xc0\xec\xf7\xabX\xbd\xf7
@\x17^\x9f%\x15\xb8T\xa7\x05\xa9g\xac\xb1\x9b\x17\x0e\x02*JA\x13\xa9\xd8\r\xbcu\x89sDY\xe
93\xb3\"\xfb/\xc25\x88\xf5\xf8/\v\x01\x0e]\x1d\x8c\x03\xa8\xc8\b\xac\u0664!\xb7BLm\xcd{Z{(\xc5\x
e1\u01e10L\x93\x02\xf2\x1a\xfb\x9d\x8e\xbc\x17\xb7W/\x929\xf7\x80\xb2WN\x1c\xd0I*\xe2\xce\x
e7\x0e96\xf5\xb2\xf4h\x16j\xbb\xe0\x9c\xe0h\xee\xb0\x14`\xa3!lZ\xed\x9e\xe3\xe5\xe4d\u02dc\xb
3F\xbd\x10\xe6\xbdn\xca\rS\xa2\xcc*\x13\x86\x835E\xf9\xeb\x94<\a\x15\x9f\u007f7.$\x19\u016f\x
11\x93\x87\xe0a\x9a\t\xa6\xbb\xe7B7\xae?K\x14\xd5\x1acpxt\x8cW^~e\x02\x00\xff\xf5\xaf\xfc\x8a
H\x05\xfb\xad\x8b\xf9z\u0742u\xe3\u018d\x00\xb3\xbc\x81\xbe\xef\a\x1e\xe4\x91\u007fl\x82]\x88\
x87\x04Iv|\x9e\xcd\b\x9e]i;$\x91\v\x92\x89T\x8d>\x04\x18\xa0d\xa9\xe0\x14\x0f\xa9\x9a\x8e\x81\
x9aK!\xa3]z\xb2@\xb7\xcap\x91N\xc0\xfb.\x83\vqo\x12\x01\x16\x02,=\xc4\x0e\x1aT\xa9Y\x119\x9
c(gE\n\x17\x8a\u039c\u021c\xb22A\\Y\x11\x94Cd\x91\xb2\u04e3\xcc4Ah=\xbd^w\n|\xca\xc4$\x89
\xb5v)\xf0\x8d\xa07\xd9>\x9c\a\x9e\xe5\xb2\n\x85\x87\xce\x14\xd5\u0303\x8b\x80\x90\xe8\xdcX6
\xe7\xe9\xc7K\u009d\xdc\xf5KTkF\xab\x85\x99\x83\x9f;\xb8\xa5\x87[xx78\xa8\x84\r\x9f\x1d\xd0\x
cc=\\C\xe8'\f\xf1\x82e\xef\xd1y\u00bc\x03\xb6Z\xc6\u0532b\xe4\xd1}\x92\xf2\u0463i4lZzI\x14E\x1
5\x101`\f\xe0\xbc\xe2\xe1\xc80\x95Gq\xd0\x11\x00\xbd\xc0\x1d\xab\x8b\xa2\x04\xee\"\x89\x86\x8
5;q\xa0\xa6\xc5\xf6\x99\v
29\x9d(l\\\xd6\x1a\xbe~\xfd\x06\xb6vv\xbeOD\xdeMDO\u007f\xf9+\x8f\xf0\xea\x91m]\xcc\xd7\xeb
m\\\xb9[L\xa8o\x91n\x13\xdd\x00\x95\xa2\xe6{\x0f\x81WA\x87\xb6Ia8\x88\x8a\xafM%\xf6-
\xd0\a\x89\x18Q\u0418fp4,\u0265:h\x00<\x14\xb4\xbe*$X\xc6Fv#\xcb+I!\x9b\x06|\xe4\x82JD*\xdb
[\x9a{\xe0\x10\xc0n\xe9\x94\x18\x8a,\xe5\"Z\xee%\xe9\x8c\x10(\x10\xf92J\u0751\x97@6\u0556_\
xa5\xf5oe*%\x80\x9cx\xd0\xccg\tyX\xbd\xa8\xb5\xac8\x0f:\xea\x01\x03\xb8M\x13\xca\aeXb\x18Y\
x11\v\xb8\x94\x87\x88\xdce\nj\x1dV\x91\aR\xed\xa8\x12:\xd3\xf85bW\x1eA\x1d\xd7\v\u0731\x83\
x9bi!G\x9f3Y%\xf8\xbe\xe4\x03\x00\xc1\xf4\x82f\xee\xe1C87y\xc5\u0717\x0e\xe8\xe7\x1e\xf3F\xb
0i\x19\x1b\x96\xd4y\xb3\xb8\x8e\xd1q\u0449(\x1f\x1e\x02\x1f\xa5\xf6.\x18|1A\x1cU\xa7\b\xef\x83\
xef\xcaR
'\x0en\xe1\xd5\x0e\x80\xc2\rCq\x98\xeba\xed\x04[\xb7]\x00\xb3\x85x\x970\xf7\xc0\u28ae\xefq\xe
d\xda\xf5\xb3\u007f\xfe\xe7_9\a\xe0\xe9?\xfd\xd3?\xfb\x96rN\\\x17\xf3o\x83\xb5\xb5\xb5\x05\x00\
xd8\xdd=\x83\xa6\xb1\x15\x94\x91Q\b\a\x17Zg\xf1Y\x99W\x9e\xe3%\f\v1\x92\xb5\u0245\x97\x85\
xbeIb\x89I\xe1%=\x90\x0f\u055e
+\x8aCY\x1dv\x9e\xd2\xcc{\xc9](H\xbbt\xf4\x12\xb3\x12\x10\xea\x06\x8c\x03\xe8\u062b\xf9\xd56\

xe7\xe1\x1b\xe5\xee\xb9T|Fiz\xacR\x14v\xa7\x98Z_\xbd&\x1a\xf7\xa1\xc90\x05\xad\x82D\x9d\x0
0\xc7\x0e\xb2\x10\x90SJ^\xec(\x97\x02\xf4\xe1\x1d\xea\xf5\xf3=\x11x\xcahL\xceT\xa5\xe2\u0691
T\a\xaej\x10-
8\xd5Q\xb7\x86\xbe\x90F\x11\t\x1f\xf7\x92\xdf\aT\x9c5?v\xf0G\x0eXx\xf5l/TW\xb9\xb3\x0f\xc6`a\u
00f0K\x81\x9f{,6\xb5;\xe7\u008fg\xd6\t:\xe7\xd0y\xc6f\xc3hM\x11\x10R^\xe0b\xa0\x1c\xd9A>X\xf8
:\x0f\xf4\xbd\xa0\xef\xbd\xd2\x1d\xbdW\xe8m.\xa0.\x04\x0ez\xa9g\x0f\xa44\"\xdbN\xb0}\xf6\"\xda\
xcdm\u030fn\f\u0384D]\xef\xd0\xf7\xee]\x8f|\xf9O\x1e\x04\xf0\xa5\x97_~e]\xcc\xd7\xeb\u05ae\xf3\
xe7\xcf\x03\x00.^\xbc\xd0L\xa7\x1b\xa6\n\a\x0e\x0f]\xdf-
\xe1l\x0f\xd9d\u040c\x13\u034b|\xf4\xba\x88\xe7\xe7,d)\x8b'\xb3\x0e\xfe\xa4\xc8\n\x95\xa0^\fv\x8
6`\x89r\xf2\xec\u0597\n\x10\x8d\x90\xd1O\xa9\xdc2\x98\xd9\xc5\xe3\xbf\x10@N\v\xa4\x84\x82.\x
bd\x84\x87\x1a\x80\u04ce\u03c8\xc0\x1c(\u03dc6\xa3\xf2\xb2\xe8\x9c\aa\x10R\x85\x86Rm\x8aM
\xd9\xe50y\x84\xa0\x8c\xb2\xab\xc5P\x05D\xaf\x1d\xe6Qx[\xc6\xe0\n\x86\x17\xe5];\x0e~[>lT\xc7\
x0e\x1d\x01\x1e\x16\xdc0,\x11\x1a\x124&\xe4rV\xae\x88X\x11\x14\xc5\xea\xee\xb3\xcd\x14\xa8\
xc4\u05fddc4\t\xb1|\x11s\x0f\x17\u018b`\xe9\x05\xb3\x85G\u007f\xe4`\x97\x02\xeb\xea\xefYN;d
@Ae\xa7\xddy\xdf\x10\x1c\xa9\x1f\v\xdb\xfc\xba\x9d\a\x0e\x17\x1eK'\xd8j\x18\x9b\r%\xc36?82\xa
9s\x80\xa0\xf3@\u702e\x03\xba\xa5G\xb7\xe8\xe1\xfa\x108\xed\x05f\xee@}T\ud19d\xaa\xa0\u06
2a\xca\x13\x90\xdeac\xeb,v\xce]\xc6\xec\xe0\x9a\n\x95(:4\x12\x11\x91;8<4_\xf9\u029f\xee\x01\xc
0\x9b\u05ee\U0007a62f\xd7-
]\x97/\xdf\x05\x00\xb8\xff\xfe\xf7\xbch\x8c\xb9\xd64\xed\x9d\xfd\xa2\xcb\xcf\x05\x11\xdcb\x8e\xc5
\xec\x00n\n\xc8\x06CN\xfaB\xf4\xa2\xc5\\\xbcW\xe2o\xe9~H\xc8\x11k\x94'\x9a\x12\n\xba\x17\xc0
\x05\u05fb&(\xf6R\\[=\u007f\xac\xbaX*\xf1p\xac\u01b6y\x19\xf4\xbf\xf1k\xb9\x80\x89\xfb\xe8\xa1\
x12\xbbv\xfd;\uf05e\xb5\xa80<\x98\x18<\r\xa1\xc8Bu\xb0\x05Q\x96J\xa6\x02?\n\xe9Ws\b*\xe2\u0
688\xea\xd7.\xe9\xe8\x0f\xc8\u0703\x0f\x1cp\x02\xf8>\x9ayi\x01\xf1\x84\",B\x92\xa8\xca\xce<\x1
6\x8d\xc7b\x9b\xd0A\xb0\x00`z\x825:\x18l\r\x05\u0719j%\xfe\x00W\x8f\xa9N\\0c\xbcD\a\u00c8YQ
\xc2\xfe\xbd\bz'\x989\xc1\xac\xd7\xeez2w0\xbdO\x1d9\x95\x1e\xe0X\xfd\xfdD\x1c\x9a;\xa0\x9dy,\
f\xa9\xf0H\xa8\"\u0590\x00\x8b^\xd09\x8f\xce\x01\x1bF\x1d\x19\xf1\xb2~\x8c\x00\x00
\x00IDAT\xbd\u03ff\xf7\xde\x01\xb3\xcea\xd1y\xb8^\xe05\xd5\x0e\xce\xeb\xc9\u0487\x19\x89]z\x
d8pj\x90h#_\xdcx\n\a\xe9\x89\xcb-
\xe6\x98l\x9e\xc1\xd9K\x0f\xe2\xb5\xe7\xfe\"5/\xf1\x04\xd6\u0616\xbf\xfe\xf5W\xf0\xe0\xbb\xef{X
D\x1a\"Z~+\u0541u8\u0177\xc1z\xdf\xfb\x1e\n\xb38z\xcc;\xf7\xfc\xf6\xf6\xb6\xfa\xaf\x14\x86L\xbe
_`v\xed58\xe9\xd0oY\xb8\t\x87pa\xed\xd0<\xbc\xfac@\xbd\xc3\xd3s\x1bR\\$\xfc9r\xc0L\u0228,:\x
ea!\xfb\rU\x06\xc0\xcaG\xd5\xe7\u0240\x12\x87S\x86~]\xe8r\x9d\xc0\xf7\x02\xd7+\x03$B\xdc\xec
\xf5\xe3\xae\a\x163\xe0\xe4\x06p4\x03fN\xd0I\x0e7F\xe2<\x97\x85\x9aV\xe97$\xab\xaf\xbcP\xe6
D\xf6\x8b\x17\x81s\x0e]\xef\xd1y\x8f\xbe\xf7\xf0\xc7\x02wBp\x1d\xc1{J\x9f\x1b_'\xc5\xea\x1a\xac
\x03\xd8\x01\xdc\t\xda\x13\a\xd3\xe7!d\xe7\x05\xb3Np\xbc\xf48Zz\x9ct\x82E/\x8a/\x17\xb8\xb9H9
\x89(\nZP\xcc\xc6k\x9c\xdf\x04\xcey,\x9d\xc7q'\xd8_x\x1cu\x82\xce\t\xcc\xc2\xc3\xf4\xa2F`R\xc8\
xe7e\x04\f\v\x03W.\x14\xbev\xe9\xd1\u0303\xaf\x8a[\xa5\xd3pPv\x1eu\x0e7\xe6\x0e\a\v\x87\xe3\
xa5\xc7I\xe7q\xb0\xd4?;Z\n\x16N}\u1f68\x8a\xd9Z\x03n4$\xda:\x82\xedC\x01\x17\x02\x87\x1f\x8c
D\xe1\x1d\x16$=\x82x\a\xe9{X3\xc5\u079d\xf7c\xb2\xb9\v\xef\\:\xbfH\xd8\xd0Ofs\x1c\x9f\xcc>p\xf
d\u06ab\x97\xbf\xd5\xea\xc0\xba\x98\u007f\x1b\xac\xe9t;\xbd\xff\xc0\xbb\x1f\x90\xdd\xdd\x1d\xb8
\xbe\x97\x12\xb7p}\x8f\xe37^\x86_.\x81\xb6A\xbfi\xe0Z\n\x1d\"\x82\x90\xc3\xd5>\x18a\xe8T\xce2
)\x14p\xa6p\xf4g\no\x99;\x1c\xe1\x199e\x92I\x83\x02\xee\a\x03\xd8!\xf7\xbdt\x00t\v\x8f~\xe9\u04

47\xff\xfb\x98\xff(\x9a\u00e9\xff\x97\xf0
\x03\x98\t\xba\x03\xe0\xc6\xc2\xe0ZG8\xea\t\v\x1f%\xf4T\xc0%\xa5
^*+\xf0\xe8\xab\x18i\x9a\xf1T\x12\xdd\b\x9d\xf3\xe8{\x87\xcei\x81u
\xf4\vF7c\xb8\x9eV\x87\x8f\xe1\xd8\x1fO\ni\x18\x1d^\xbbY\n\x9a\x93\x9c\xd1Z\xb2F\x97N\x8b\xfa\xc1B\xdfN:\x85,\xa2\x16)\xbfI*R^j\xa6J,\xee\xcb^p\xb8\xf0\xb81\xf78X8,\\\xa0\xeby\x80;\x0f^\x86k\xeasW\x9eSz\xa4\x1e\x8bH\xa9\xc2\xd7\xe1g3\xf3\xb0s\u5967\xc3P\xf1\u03e3\x1dD\xe7\x04'K\x8f\xa3\xa5\xd3\xcdj\xa9\x1bX6\x0fCrP\x8c7\x1a;\xc0v!{\xd5\xd7\xe0\xbf\xf4\x1e\xe2\xf4M/\x8c$\x9bg\xdf-
\xb1{\xee2\xce\xdcy?|\u07e5P\xedxk.\x97=\x98\u0307\u007f\xef3\xff\xfa\x01\x00x\xf4\u047f\xfa\x96\xc1\xcd\xd70\u02f7\u067a\xfb\ueedfi\u0513\x99\x04\xd9\xe0\xd9\xf7=n\xbc\xf8,\xdcb\x06\xbb\xb5\x8b\xaea\xf4\xdb\x04C\x0e4\xcfT1\a\a\x8e\x14<\x92\x14zL\x05K,\x9d\x95\ubf00\x02\xad\xa0\xf4\xac3j\xd6\u02f0\x1b\xc7\x00F\xa9\x1a\xdfb8\xd7;\xc1\xe2\xc4A\x0e:
\xf0\x9e9\xc8\u096a\x109L\xc2\a\x8c\xdd\xce<z\u00d8o2\x16\f4\x0e\u0630\xc0\x86\x01\x1a\n~\x88\x15\xb6\xe2Wp\xe8\x01\x05^\x8b\xa4\x0f\xd8wP\x18\n\x18=\x18]G\xc0\xb1\xa0Y\xa8\xa4|\x84\x89\x9e\xe6\rTX\xf8\xa6k\xdd{4\u01ea\n\xed6x\xc5\xd4\xd0\t\xe0D\xb1\xe4\xb9\x13\x85_\x98\xd0\x18Bc\x82M1Q\xf6='d\xf6\x87\xd3\u03bbs\x82y/\xe8\x00\xb5h@a\u0535p\xb03\xddTHj\u06c3\xd1\xe94\xb279\x87_\x98'\xc0t\x82\xf6$\xb0[lpQ\x1c\x8a\x82\v\xae\xb8/\xf4\x11%\xd5R\xfd\x82(\xc1~\x1cpr\xd3\xf9\x9c\x13\x1d\x8a\u007fd\xd8\b\xa5@\xbc@\x99U>\xbf\xef\x97\xd8\u07bb\x80\xf3w\u007f\x00\xaf_\xf9j`\xf4P\xe2\x9e\v\x80W_\u007f\x03\u007f\xf5\xb5\xc7.\x02\xc0\xfb\xdf\xffAY\x17\xf3\xf5z\x87\xba\xf4\xe9\x1f:\xe7~\xa1i\xec\x8ew^}\u02a1\xc7\u0323W_\xc2rv\x8c\t\x1b\x104,\x19[\xd0\xe0\x85\xb9\x1e\x93\x1d{\x80\x03\v!\xa3\x00\x11W\xc9L\x16\u0285\x8e(\x17x)\xcdGiP\a\x06\x05{\u0629\x97\x03\xd0\xe4+\xee\xb5\xe8\x1c\xcd\x1c\xfa\xfd\x1e\x93\x13\x8f\xb6\x0f\x1d#ibO5[\x95\xb0\x01\xf9\u8ded\xf8\xf9\xf4\xa4\aL\x83~\x83\x94E\xb2\x04f\x860eB\ub056\x05\x86\x82*r\xe8\xeb\x8e2\x18\"\xc0\x13\xa1\xa0\xc6b\xe2\xc0\xe8`\xb0\x14\x03\xcc<\u06a5\x86YP\x1c\xbe\rmi\u00b55^\a\u007f\x1eY|K\x02P\a4'N\u00ec\x1b\xca\xf4\x93\xc1\xcb\x12\xd1n\xbds\x02\xe34\xf9\xa7\t'%\x13\x8d\xc8\u0086\xd8-
=\xbaN\xf9\xdf.\xea\x97L\x10|E\xcfz/\xb0s\x0f\x13^\xff0R\xaf\u0718\xaa\x01w\xf8\x05s\xfcf\x81'n\x96\x1evF\xe8ZN\xa7<ZqX\xf6yn@R\x8b\x83H\xb7f\x82@X\xc0\x8e`\x96\x00\xbb\x1c\b\u0392\x95\xc0\x948\xec\x94N\x89y\b\x1f\xaf/\xe1\xcc\xf9wac\xfb,f\x87o\x82\x9biz-
\xc6\x18\\}\xf1%|h\xfe\xc1\xef\xf8V\x13\r\xada\x96o\xb3\xf5\xc9O~\xf2\u037d\xbd\xbd\x05\xb3J\u0563k!
X\x1c^\xc7\xc9\xf57\xb2\xc0E\x00g\x19\u0766\x81\x9b\x10\x9c\x01<\xc5\xd2\x12n\x10\x8fp\x94\x95\x14\xdb\x06_\xfb\x84\xa4\x87RV\xc5\xf3\x83:[\x9b\xf6\xa1\u019d3.\xadP\xc0\xb2W(\xe1\xc6\xc2\xe1d\u1065\x87q\xb2\xf2}V\x1bFI\xd49\n\xbc\u0148E\xdb.s\xab;'8X\b\xae\xcf\x04\xd7g\xc0\u0452\xb0\x14N6[\x92\u07b40y\xaf\xc7\xffX<E\x00'\x84%\fN\xa4\xc1\x89X\xb8\x0e\xda\u057a\x9a\xcaW\r'\x91\v\xa7\xe9\x14\x9b6N\xaa
\x10\x88\xc0,\x14n!\x87\xcc\xef/\a\x8e\xc5\xff\xa1\x89uX8\xc1q\xe7q\xb8p8Xx\x1c.\"$\x13
\x8c\x85\x87\"p\x83IF`\xf5p'\xb0s\a\xee\xe54\r\xd7Hg\x9eOF\xc9Z\xc1k\a\xcd^\xd9-
f\xe6\x14;/\u0607\xb9\x1f\xf0\xc9?eh%\x11\xe3\xf0b\u03a7Y\nl\xef\xf3\xcf\x1e\\(\xa3ChY|%\x85r\xf8j\aq\xcb9vo\xbf\a\u06f7]B\xdf-
\xc2\xd0\u06a7(\u0093\x93\x19\xf6\xf7\x0f?\x02\xe0N\x00\xf8\xf2W\x1eY\x17\xf3\xf5\xba\xf5\xeb\xfd\xef\xff\xe0\x17\u039d;\xf7\xc6\xce\xf6\x0e\x9c\xebU\xea\xac\xe7N,g\xc78z\xf5j\x90?\x1bm\xb6

Az\x9c\u07f6\xe8'\xac]\xa2\xb8\xf4`\xc6\xeeRz\xed\x92{\xe5\xbd\x148l\xae\xc2%t24&\x1c\x1bl\n\r\xc2*\x82\xc4z\u953ev}\xeeq\xb8\xf4\xe8|P\xe9{$\fw\x05Q\x1f\xa8;S7\x16pt\b`\x16\x1e\u0371\xd2\xd8\xe2
\x17\xd0\xc2|\xb4\x10\\?\x11\\\x9f\x11\x0e\x97\x8c\x85#\xb8\x98?\xeaU\xe0\xd3y\t\xb8\xb8\x1e\xfb;0N`q\xec-
\x96PFh3\xf30]\xc0\x88]\x06\x94ip@I#WO\xa1\xf0\xe9\t\x82b\a\x1c\nU\u011d\x87\x85|Xai\xf0u%\xc01K'\xe8\xbd$H9\xf9}Sa$\x16\u007fg\"03\x9f\u1555\xcd2\xb3\xf3\x05\xa7\x15z\xc9\x03^\xe8\xef\xca\xf6\x1e\xed\u0301;\xc9\x12\xfd\xf0\x83\x88s\x90\xde%\u06a4\x1f1<\x8b\u0775Y\uad48\xa9D\x14\u007f\xb7T\x06\x87\x87\xfb\xc8\xc7B\x1emh\xf2=\xe2\x9d\xc3\xc6\xf6Y\xec\xdeq\xb7f\xb9\xa6\xcdDW\xef\x1c\x0e\x0e\x0e?\xfa[\xbf\xf5\x9bg\x157\u007f\x9c\xd6\xc5|\xbdn\xd9z\xf2\xc9\xc7\xc2\xf3A\xb3\x87\x1ezh\xbe\xbd\xb3\xad\x0et1a\x86\x80nq\x827\x9f\u007f\x02\xde\xf5\x1ay\x96\xfcE\b\xd22\xba-\x037\t\x93\u007f\u7a87\x99\x9c@\x16\x1e\xae\x0f\x98-
\xb2\xce9u\xe8\xc9f\x16\u0574k\x14B\x89\xb2\xf1\xf0q/\xda\xf1\x1eu\x1e\xd7\xe7\x0e\xfbK\x8f\x85\x93\u031c\xf1\xa2\xec\x8a\u00a8\x8ad\xd0\xfb\v\nA)\x15Gp\xfd\xbb\xd8%\xdaY\x96\x84S\x819/\x98-
=\x0e\x16\xc0\x8d%\xe3\xa0c\x1c\xf7\x8aK/\x1c\xd0{RL\x1c\x063\xb18\xf6\r\x16\xde\x04a\x8b('{\xe1\x146\x90|\xe4\x90\xc1IE\x068\t\xf9\xf0Vp4\xd3|\xd1iwn\x96z\xc2\x12)q\xe6\u054d\x92\x06oLyp\x9dy\xf2\x94<\xc5\x11\xf4\x02B\x00/=\x9a\x99\x039\x19d\xac\xd6g\xad\xfc_\x19?\x19\x85\x91\ny\x81\xf1\xca\u04b1s\x81=v*\u02e7|2\xf4\xce\xe9\xfd&\x1eU<w\xa8\xec\xe9:t\x1ev\xe6\xd4`-
\xc4\x05r\xf8\xbf!\xceXy\x11\x8f\x17\x1d-
\x93\x127u\xec\xfaL\xec]x\x00\xd3\xed\xb3\x10\xd7W?\x87\xf7\x82\x17_zi\xf2\xda\x1bo\xbc\x17\x00\xfe\xb3\x9f\xffyY\x17\xf3\xf5\xbae\xeb\xc1\a\u07d7\xde\xdf\xde\xde\xfa\xacZ\xb4\x1a=>\x86\xc1\xa0[.p\xfd\xf9'\xd0\u03cea\x8c\xa9\ak\x02HCXn\x19\xf4\x8d\xde\xf0\xa9\x93\x11\x1d\xf6a\xee\x80\xc3\x1e\xbe\x93@ P\xec\u060b\xfa\\y_\x02/\xf5\xf0-\xf1@\x12\xcb-
\xc0\x14\x01\xb68\\\n\xae\xcf=\xf6\xe7^y\xceR\xe6F\xaa\xf5*\xf7yhXU3\xd4\xd9qe\x1c^\xb4\"H\x96\xb1\xbd\xa0\x9dy\xd8P\x1c\xd3\xeb\x93\x1c\xf2
\"Xz\xe0\xc43\x8e\xbcv\xde3o0\x13\x83\x13X\x9c\x88\xc5B\f\xfa\x98\xfaN\x00\xf5\x80\x9d9p\xa0\xe1\xb1S\x88a$ \xaf\n4\xa3\x04I
\xb1q\xaa\xd0\v\x00\xa6\xf3\xb0\x01n)\x9fX*\xf1w\u0526\u00e8\x1b\xd5\x02\u007f\x1e\xf9;\xaf\x83\xe2&\f=\xd9\xcb*l5J0\xad7\xd2*z/\x15\xf4\xf0\xf3\x85M\x89\xe7>o*\xde\x01\xbe\x0f\xf7\x1a\x06\xe1\x1b\xf9\x9e\xa1\b\xd5t\x92\xba}v\x8a\x93s\xc4\xca#\x93%\x06G\xa7$\xa2\xb0\x891U7\xa3w\x1d\xce]x\x0f6vnG\xdf-\xf2s\x02\xed\xe2_~\xf95\xbc\xfe\xfa\x9b\xffa|-
\x8f>\xfaW\xebb\xbe^\xb7~=\xfc\xf0w>\xb2\xb3\xb3\xdd5\x8dzP\x88wZ<\xbd\xc7\xc1+W\xb0\u007f\xf5\xa9\x9c\xb2\x1e\x8bvx\x13K\xe8\xa7\f\xc7\x01;\x17U.\xf6^\xd09\xa0?\xec\xd1\xdd\xe80_(7\xf9h\xeep8s8\x98\xf58\x98{\x9c,\x04\xb3\xdec\xde\v\x96}\x80%\x9c\xaa4]`\u007f\xf4^\xa1\x94y'8Xx\xbc9s\u061f;\xccc\x11G\x9d\x0fMPf\x04\xfb1,gP.\x05+JL\xf2\xc1\x9b;*\x06\x97\n\xb7p\xef\x87_\xa4*\x86\x19\x13g,`\xb0\x80A/\x9c\xb4\xb2E\x9e\x05\x9a\xb9S\b\xc0\xe5b\x13\x8f.C\xf8\xa3*\x95\x9c\x95\xba\xe4\x03-
\xb0\x1c.\x04\xaby;\xf7h\xe6\xaeH\x10\x02\x06\xc9\u007f(\xffKE7N\xa5\x9d/e+\x83\xf2E\u0645v\xe5\n\xf5\x10\x86{\xf2*\u06fe\x18SS\u044dK6k\u031b\x8dRE\xcd\u00a39r\xaa\u0685\x069\x8b\xf39\x1c\xa3\f\x9e.6\xaaf\xe1`\x17>\f\x8cI-\x1bb\x17\x1d\xe9N\xd5\xc9
\x8cU}Q\xd0\xc3&\xedS5\x17l\x9d9\x8f3w\xdc\x03b\x0e\xf3\xa5\xc0.\x12\xc1\xa2\xebppp\xf0\x9f\x

1f\x1d\xbcy)\xc0\x97\xebb\xbe^\xb7~}\xfa\xd3\u007f\xf3_\xbd\xfb\xdd\xef\uedb7\xb6
\xe2\xc5\xfb\xc8Y&\xcc\xf7\xaf\xe3\u0367\xfe2t\ub738\xd2U\xe0/\x03\xdex\x1d\x86\x86\xa1gT\b.\
x17\x1e\xcb\xeb\x1d\xe6\xd7:\x1c\xcd\x1c\xf6\xe7\x1e\xfb3\x87\x1b3\x87\xfd\x93\x1e\u05ce{\\\x9
b9\\\x9f\xe9\xe0\xf2\xc6\xc2c?\f\xe1\x0e\x97\xf5\xdb~\xf8\xb3\xa5\x93\"\x86\xae\x1e\x98\"tb*`\x89\
xbe\x1b8\u0144\xe4\x94a\x9dW\x00\x99|\xaeLv\xee\u041c\xf8\"<a\u012e\x97\xea\xa24\x9c\a\xc4
a\xa5Y*\xd6\x1c\x8b\n\x87\x82N#\x18D\xad1\x1d|\x14:d\xf8\"\xb0)\xc2\xee\x0e\xb0'\x1e\x1c\xad\f\
x06\xa7\x90\x95\x92\x9b\xe6\x19y\xa0(E\x98\xb1
\x17M\xee\x05\xf6D\x19,\xec\xe4T\x81\x00U\x00}\x9d\x105\xac\xef\xd5\xec\x02\xb9C7'\x0e<s)d\x
da\a\x88%u\u5147<\xc5k{\xe2\u00c6L\xba\u0641T\xd9K\x9c\x99Ta\xf8\xc9E89\x0f\x9d@\xa9T\x
88\xea/\xf0\xf6\xcb\x0fa\xbay\x06\xe2\xfbt\n%f\xf4]\x8fG\x1f}|\xeb\x9f\xff\xce\xef>\xfc\xad\xf2\u072
f\x8b\xf9\xb7\xd1\xfa\xdc\xe7\xfeM|\xf7\xf0\x81\a\x1ex,\x18\xec\x93\x1e=\x1d@\x8c~~\x8ck\xcf?\x
0e\xd7-
S\u01e2\x1d\xa0\xd3\xf3\xb6\x0f\x82\v\x118xx\x1e\x1c\xa8\xc3d\u035c8\x98\x13\x95\xd5\xfb\x88\
u007fz\x95a\xf7=\xb0\U001024e0Z<(D.\x87\v\x8f\x93\xa5`\xdey5\x99Bf<\x8e\xc0\xb3\t\x1aQ\x88
%;\xf5\r\xd9\x18\xc3\x0e\xbb\xa2\u03608\xf2\x17\x1bDs\xa2\xdd4R\x105a5e3\xe3\xd442\xc5$\xaf
E\xd6tR\r.\xc9I\x8d\x8d\xaf\xbc\xea1P\x04`\xef\xf3\t\u0123\xb0\x8b\x11e\x9a\u031c\xfa\xe9\f.\x97/
f\x18\xe3_y\x14\x91\x82\b`\x16N!\x90N\xf2fYz,\xa4\r`0\x9f(\xbd\x04\u02bc\xcd\xc4%\xaf\xbfQ\xb4
\xca5'\x0eX\xf4\xa1y\xf0\xa9\xa1\xc8\n\xd6p\xedz\x0f;\xeb\x95YS\x0e\xbf\u3f03\xa0\x05\x9d9\xf1\
xc4KK\xa2tm\x12S\x86R\x8c\\\x9c%\x9d\xbb\xf8^L\xb7\xf7\u0f7e\x9eHGt\xce\xe3\xf5\xd7\xdf\u01
1f\xfd\xf9_\xfcd\xfc:\u007f\xf0\x87\x9f_\x17\xf3\xf5\xba5\xeb\xe3\x1f\xff\xe1\u0521L\xa7\x93\xff}cc\
n\xdb4\x19\x13\x06\xe0]\x8f\xa3W_\xc4\xec\xcd\xd7@\xe0\u0288\x88\xc4\xeb0\xc8\xf5\xc1|\xcb\x
c3G\xbb\\\x1e\x84\xe0:\xf5\u07d8\u0303\xcf4\xe5#u\nA\xa6\xfa-
w\x87\x92\xa9u\xc3r\xb7\x12@\xa4t9v\x12X,\x85\xb0u\xd8\x05\x16\xb62\u547f\xac\xca$\xb9`q\x1
f\x86\x8b\x9dO\xf4MB\xa6>&D\xa7t(\x1c\x90<x\xe9\xc1\x91S\x1ep\x17\xf2\xc3fYN\xd9t\n\xbb\xe2
\x02xfW\x84\x82\x94?c\xe4\x80/|\xbd\xd9\r\xdeV6\xa3\xd3pn\xa2\x84e\x9b\xa5W\n\uaa7b\xc0\x8
8)\u02d8A\v\x06\x9e\xf6\xc3Y\x86\bx\u0583\x8f\x97jX\xe3\xa5\xe6\xa6\"_|;\xd7\xd9F\xb2\n\xf0\xf
5\x04=\xbf+\x03S\xb6\xda\xf63R\x16c2T\x9a\xb9x\x87\u035d\xdbq\xdb\xf9\xfbA\xa4\xac\x96b\xfe
*'\xb39\x9e\xbfr\xe5'\xfb~\xf6>\x00\xf8\xc1\x1f\xf8\x18~\xff\xf7?\xb3.\xe6\xebuk\u05f9s\xb7\xff\xf1\
xddw_F\xdb4U\xcc\x16\xb1\xc1\xec\xfa\x1b\xd8\u007f\xe9Y\x95H\u01ee(\x98ei\u01e2]\xbab\x8c^
9\xce<pA\x14e*\xb4s\x8fvQ\xc3\x15\xe2W\x9fu*\xe0\xe1*\xc3\x19\x18\t!\xce\x0f&y\x81\xe9}`X\xc
8\xc8\xe0M2\x1c1L\xd5I\u07ae\x19;\xa7\x00x\xc7\xee\x91{\x8ff\xa6Xw\x15\xcb&\xa8L\x14\u01ec
z\xc9\x03\xcdB\xd2\xd03\x16\u0728F,q\x109-
\"o\xe4''\x0fp\x1f\xe1\x1a)\x95\xf3Z|g\x1e\xa6\xaf;\xechx,2\u019a\xa9\xf7\x8f2T\xa3Yz4s\t\xbcr\x1
9\x84\x84\f\u01b6oa\xaf\xbb\xe2r\x9c\\\x12\xa8\xb6F\xee:-
\xe6K\x97\xe62)\x05%|W\xc5\xf0%\xcd;8x\xac\fO\x00\xf1\xd4\"\x85\x1d/E\xf8%n'Q=*R\u51ca\xf7\
xb0\xb6\xc5\xc5\xfb\xbe\v\xcdd#\xe5\x8c\n\x00f\xa6\xa3\xe3\x13<\xf5\xd43g~\xe37\xfe\x97\xbf\x1
5\xbf\u0697\xbe\xf4\xa5u1_\xaf[\xb3\x9e|\xf2q\x00\xc0\xe5\u02d7\xaf\xdeq\xc7\x1d/\xd8@A\x8c\x
8f%\x1b\x83\xf9\xc1\x9b8\xb8\xfa\x14L\xc8pL\xc3\xcfp\xa3\xfbP\xd0)\xc0.>(>c\xaaL\x1c\xf2A\xa0\
x05}\xe6\xd1,\xfch\x81\x1a\x0e\xff\x860\x80\xc88\x04\x90:{\xa7\xf0\x822=h\xe0'^\x1f\xab\x85\xea\
xae0\x16\xfe\u0215O\xf3\xba\x82sL.\f\x17\x17\xbeN\xfe\x19r\x97\x87\xb5+\xe2\xb9K\x9f\xd9\x1f\
x82\x84\xeb\x0f\x85Q\xd5\xd7\xe128{\xac\xf4\xea\x06\xc1\x9d\xcf\xd0G\xb1\x91f\xbeuY\u0425F>
\n\xf8E\x06\u05d7\n\xf8J7\x86|Z\xa1*\xf7\x94\xea\xd3C\x89\xa6\xac\b\b\xa4\xee\u02a5t\x12\x0e\x

b4J\x0eZO\xe7\x82e\x80\x0e@3\"\x13\xe4\xfa\x9d\xc0\xc4al\u0090d\xe5^-
\x19WD\xa5NW*x'\xb2[\x12\x9b\xcaG\u058a\xee\xe0g/\xbe\a;{\x17\xe1\xbd\xcb\x1b\x1a\x11\x8c\x
b1x\xf3\xcdk\xf8\xccg>\xfb)\x11\xd9\x01\x80\x8f}\xec\xe3\xebb\xbe^\xb7f=\xf8\xe0C\x00\x80\x9f\xf
a\xa9\x9f>\xd8\xdb\xdb\xfbB\u06f6`\u00c1k\xabC\xcf\xe5\xc91n\\}\x1a\u02d3\xa3\x18\xe1[\r\xb1H\
x04pN\x99\x06\xe2A\x81s.&c\xc2%\x83\x80{\xc1$\x14\xf4lBU?\xectJSJ\xa7\xb4\x91\x11\xdePJ\x
e2\x00.\x19<\xb0\x82\xa2\xed_\xc9o\x96\u02ba5}\xdf\xc2@\x8a\x05hO<\xec\u04af4\xa1c\xf0/8\f\
xf3\xe6n\u0415\a\x16\x8b\x8c`\xf9E\x9b\x1f\v\u06eaD\xb6\xa8\x86\xc1k\x9e\x9d\xa8|\xdd\xe7\x93
\a\xf9P\xec:\x19\ud307\x8c\x9c\xea4Q|\xbe]ze\x89\xf8b\x17\x93\x9b\x1f
\xe8\xb4\x1dz\xe0\x9f^}\xef\xb8\x11\x10\xb41\b\xc3\x15\x9e;U\xbe\x96\xa6\x95\xc1K\xc7v\x92\xbf
F0\u02eaY/>\xe1\xec\xbe4v\v\xc5\xdb\xc7\x01k\xf8\xfb\xf8\n\x15\xf2\xe3tO8\xd7a\xb2q\x06\xe7\x
ef\xf9\x0e\xdd\xfc\xbcK/\x9c\x8d\xc1\xc1\xe11^}\xed\xb5\xef\xfbG\xff\xe8\u007f\xfay\x00\xf8\xc1\x
1f\xfc\u063a\x98\xaf\u05ed]Dtt\xd7]w}\xfe\xe2\xc5\v`f0s1\xe8\x13\x1c\xbc\xf4,\xf6_z\x16\xc6\xd8,n
\xf1>)\xe7\xbc\xf8<\x14r\x1e\xe4\x94\xc6\xe7M\xb0\xc1\x1d\u0d26WS\xa5f.\xc1r\x14+\x18\xab\xa
1\x12\xd7\x1cU\xe1\xd7E
\xc8\xf0\xb9\xf7\xa7\x88XV\xc1\xe1\xb2\x1e\x92\x14\x03O\xc9\xe3M\n\x02\"vY\xb0c\x9c`\x12\x06
\x99+\xf3\xbf\x016-
\b\xdd\xf1R\x8a\x84\xc8\xe0\a\xe3d\xe5\x1fJ\xb1C\b#\xa5<\xd5s\x88\x81\xca*b\xfeN\u007f~^\xe6
\x0e\x9dD\xa9\x9a6\xc0C\xa3\x90\aF\xfc\xb1\n\t)\a\xbe\xbdv\xe5\xa7\u007f\x8dr\x97X1\x1b\x1b\x
d1l\r\u060ai\x86\x13?\xe2
J\x10\xa7\xb3\x06\xbb\x90L\xf1\x14\x9d\t\u0639\xb2W\xf4z\x06\x8f\x98\xd8]\x87\xc1e\xa4\x12\"D
\xcb\xd5\xd9\xdbTZx\xe6{\xae\xb8\xc1\xa2\xea\xd3\xf5\x1d\x8cmq\xe9\x81\x0fcs\xe7|\x8a\xac\x8
b\u0670ll\xf7\xd2K/\xe3\v_\xf8\xd2/\x8a\u021d\x00p\xe5\u02b3\xb4.\xe6\xebuK\u05af\xfc\xca\u00
7fE\x00p\xe9\xd2\u017f\xb8\xef\xbew)\x9c\x12&\xfeD\x00\x1b\x8b\xa3\xd7^\xc4\xfe\x95'\xc0\xd6V
j\xf8H\x0f\x13x\x888\x888\xc0\xb9\x14\xa8\x9b\n:\x90\x94\x82\x1c\xe0\x05\xd3\t&'\x0ev\xa6V\xba
\x15d1H\xa5\xf7#X\xb9\f\xba9v\x81\xc5\xe2O\xa3\xcb\xe5b(R\b\x8aV*L\xd1*\xa6D\x8b\x88o\xe7\
x0e\xddvZ\xe0J\xd1\f\r\u00c4(C3\xe4jy+W\xcaI\xaa\x8bz\xd9\xd9S\xd1\b\x13%\u07d5\x15nf\x81\
xfbG\x0f\x97\xf8\u0112\x00v\xe1t\x18:\xac_\x18H\xff\xa9
\xf6\x84\x8d\xc4v\x82&Z\xc8V\xf0\x04\x9d\x82w\xadZ\x13\x97\x11v\xe9\xe4B\x83kV~\xc5\xe81\x
e0\xf3\xa0\xd8,\u00bc\x82\tv\xa9\xe2
.6\xad\xfc&\x89:\x1b\xa3\xb2\xa9\xb0\u01e5p\x8f\xb31\xfa1(\xdd\xf3\\\xec\xf4\xe2k\x1e;\xe0\xe1\\\
x87\x9dsw\xe3\xc2\xfd\x0f\u00c7\xe1\u007f<\x1a6\xed\xc4^\xbb\xbe\x8f\x17^\xb8\xfa\xf0o\xfd\xe6
?\xfeY\x00\xf8\xd1\x1f\xfd\xeb$\"\xebb\xbe^o\xff\xfa\xb9\x9f\xfb9\x01\x80_\xfa\xa5_|\xf5\xce;\xef|\x
acm\xda\xec\xdd\x1dD\x14\x8b\xc3\x1b\xb8\xf1\xc2Sp\xf3\xe3\f\xc1
\xe3\xe6)\xbd\u073b\u0421\xbb\xc0\xb4\x90\xba\xdb\xf5\n\x85\x18'\xb0\xbd\xa0YxL\x0e\x9dJ\xe6
\xe3\xb16\u05e4\xf1\x8e\xfa\x94\x8e\x90]\xe0\x97\xdf\xec\xe8_\xbd\x16I^\x1d2\u0491&\xaf\x16\x
d4dA*\xbeV\xbb\xd0P\x05\x19qy\x8c\u03efYFW\xc1\xac\xd8DE\x9b\x1b\xd4\xf3\"T5\x16:\x1e\xeb
n\xe9\xf4\x83\a\x05\xf6\x8dv\xe2\xc1\xaa\xa0W/\x18ve\x98\xf7MN<\xc5&\xd9,|R\u050e\xa4\x88\x
dc\xf4ZW\xaf\xad
\xa3\xa4\x82=\x8c\x8d\xca^\x0f\xd9K\x05ZX\xa9Sv\x8e]\n\x9a\xc0\xa3\x1fK,\x11`esY\xb9FD\xf5\x
e6\x01\xaa\xff,J\xfb\xe3\xe6D:\x87\xe9\xbb\x05\xect\x13\x17\xde\xf5\x1f\xa0\x9dn%\u07bb\x04\xd
eU\xd3N\xdd\xd3O?\x83/\u007f\xe5+\u007f[D.>\xfa\xe8c\xfes\x9f\xfb\xb7f]\xcc\xd7\xebm_\x1f\xfa\
xd0w)&j'\xcf\x186\xff\xfc\xfc\x9d\xe7\x035\x91**\xd7\xf5\xe7\x1e\xc3\xd1+/\xc04\xadv\xe0\xa5\xf1

G\x14S{e\x1c\b<\xc8{eYt\xf1\x18\x1ce\xeb\xc1;\xc5i\x81\xb1\v\x8f\xc9A\x8f&J\xb7\x8b\a\xcf\xd7\xf4o\fO\xc7\xe5\u0411:\r:&\x8f\x82_>\xa80\x05$\x91\xeb\xd6\bH?\x14\xfc\f\xb1\xf3\xa285sI\xf8\xb9\x14\xafKB!\xb4\xa1\x9b,a\x1d\xf6\xa5(iX\x1d\xa9\xda`(%\xe2\x14\xf0\x0f\x8d\xb4\xd3\xe5F\x83<\f.C\xb2\xb9\xd3\xd7s\x1a#h\xac\xb8\x9b\x0e\x85@\xe8\x94O:\xfd\x0fW\x85U\x18\u02cf\x1a\x9c\xc8(\xc7\xe3\xb1\xcfg\x01\b\xd0\xcc=\xda\xc3\xc0s\x97b\xd0.\x999U~C\xa9\xe1\xc4B\xec\x15\xa1\xc2\xc8]\x8f\x12\xfd
\"\x8a'T\xa6,\xff\x0f\xafD|\x8f3\xe7\xef\xc5\xd9K\x0f\xc2\xfb\x0e\xd9\x17\x18h\xda\xd6\\\xbb~\x03O<\xf1\xe4w\xfd\x93\u007f\xf2\xbf\xfe4\x00|\xe2\x13?,\xebb\xbe^\xb7d\xfd\xea\xaf\xfe\xb7-
\x00\\\xbct\xf1s\x97.]\xd2b\xce\xf9\xcc\xcd\xc6`\xff\xc5gp\xe3\xca\x13`c\xab\x87(S\f3\xd4\"\u0781\x9c\x16\x00vJ\xe9\xe3h\x0eUL\xdd(\u061e\u0699C\xbb\xdf\x05N\xf4\xc0s|\x80qc\x88\x84\x14\xf6\xb0\xb4\xc2C\xce_\x84N\x03\x89cM-
\xe3L\xab\xe8\xa2\"^,\xbc\xde\xe4\xa5\x12\xbeo{\x1c\u0516\xa8\x94\xf0\u02b4XH\xe5\xd8WBN\xe3\xfc\xbcS^\xa3d:\xa0\x10\xad\x0e\x15\x8bo\x1e1e\xd3\xf9`C\x90\xa9\x8a6\rC\x05r\x93\x1a\f\xd2Aj\xb3Tj#\xc9i\a#y\v\x10\xfe\x94\xfa\x9e\xb8\xfa%\xf5\x05\x95\x8fx,\xd2\x19W'\x95\xf9\u03dc\xdeO4vB\xc9>\xe8\xc4\x14f@Y\xe1)\xc3\x13\xc9\u0419\x11\x12\xfc\x85\xa2w\x8b\xcfP\x1f\t\xd80\x9c\xeb\xb0y\xe6<\xce\xdd\xf5\xbej`\"\x04\x80\x19M;\x91\xc7\x1f\u007f\x02\x9f\xff\xfc\xe7?%\"[\x00\xfco\xff\xf6\xffA\xebb\xbe^o\xfbz\xf7\xbb\x1fp\x00\xf0\xf7\xfe\xde\u007f\xf3\xdc\xed\xb7\x9f\xfb\xea\xe6\xe6\x14q,\x19\x8d\xb7\xba\x93C\xbc\xf9\xd4_\xa2;\xba\x0e\xd3\xd8|T%$\xc3\"}\x8b\xb8\xb9\xcb<j\x8f\x94\x0f\x19\xbdT*\x8c\xd3++\xa1=\xec\x83(\x87\u01a1\x84A\x8dM\xf4\xe4^]\x12\xcb\xd82:\xad8W6\xae\xa7\x15\x1fZ\xc1{\xd3\xd7\r\x83\xcbx\u02a0\xe4\xdf\xe2\x03\x8b$p\x8f\x03V\xae\x94\xb9\\\xa9J\x19~]~j*\xcf\n\xa4!c\u00c2\x12`\xc6`\xf8JI\xec\xa4\xddm\xf8\xb8\xd3\xc1\xa1HM\r\x94\x91Q\x83B\x1a\xc5F\fy\x8b\xe6{`t\x83\xdaO^\x863\x89\x91\xe6>\x0eX)am\xd10Ka\x16\xea}\xba\xa7J\xe8>\x9d\x98R\xfa7W
\x8a\x14\x9b\\:qb\x10[\x18\xed\f\xbc\x87w^\a\x9f(\xf3R)\xcd\f\xc8Z\x9c9\u007f\x0f\xa6\xdbg\x83\xcd@\xaa\xf80\xb6\xa5\xfd\xfd\x03<\xfb\xecs\x9f\xfe\xb5_\xfb\x1f\x1f\x06\x80\x9f\xf9\x99\x9f\x95u1_\xaf\xb7}\xfd\xc2/\xfc\xa2\xbb\xed\xb6\xdb\xccd\xb2\xf9\xc4m\xb7\xdd\xf6\xff\xdd}\xf9.\xf4}/\u027c?@\x02\xaf=\xf6e\x1c\xbe\xf4,l3\t\xceq>\x1dU\xe3\xfb\xd2;\xf8\xbeW[\\\xef\x8b\xe3o-
\x91/\xebP,\x90f.h\x0e\x1dh\u9aeeQd\xbcq\x8d\x0f\xbd\xe9$u\xbb4Zf\x8ax\f\xa1\x9a7\"7\xb9\xbb\xa5\xc6\xc8++\\\xaf\x05\x9b}\x10\x13\x85anb\xb0,\x94}\x81\"H\x19\xa1;&?\x024\xc8\xc8&\x82zH2\b`\xae\x03W\a\xf0svd\xa4\xbe\xe0\xca{\xa8\x0f\xf9|\xd5\xd9+A\xc4\x1c<X\x82\xfb\xe0\u0403e\xf8\x9e\x8c\xee\x9bTAF\x84\xa1\x9fM.\x8c\xa3\x9e4)\u03142S\u01c5T\xab\x18\x80R$\x03Qq\x02+1p\xfd\xbfY1\xd3)\x83)\xca\xc6$\xaa>\xa9\xf8s\x1a\x9c\x16\xc5\x10z\xe9\xb1{\xe7\xdd\xd8;\u007f\xafr\xe1#t\x13\u00a4A\x8cg\x9e}\x16\xaf\xbc\xf2\xea/\x8b\xc8\xc6\x1afY\xaf[\xb2\x88\b\xbf\xfe\xeb\xbfF\x00\xf0S?\xf5\x93\u007fy\xf1\u2965xob\"\x9a@\r\x85N^\xff:\xae_y\x02\xd2/a-
g\xe7\xba8\f\r\xc5\\\\\x0f\xdfw\u06a5\x8fv\xd6T\b\x8f\x8a\x01\x97\x17\x98\xb9\xba\u5c53\x95;N\x06tD\t\x1d\x99\x06<\f9\x8c\x94\xc3\x1bP\x93\xffd\xb5\x8dEYOO\xeb\u042b\xfa\x19\x1c\v9t\xda\xe4\x94niB\xb0\xb1Y\x043\xad\xd8\xfe\x06j!\xb9\x01\xbd\xaf\x1c\x02\f`\xa5\xea\xba\xd1\xc8\xfb\xe5\xbf\x1b\x1c[\xa4\xd8\x00L\x1f\x95\xa7\x92\xb1\xf3Y\x98e\f\xf6\x85\xb8!\xc488._\xaf\xe4L%\f\x83\xa7Qdq\xa6\x8f\a\x89Ho\xb9Jf\x8aT\xc1\xcf)t[\n3.h\xf0\a\xa1\xf0\x81\b\xefk\x016
6A\xe5i*A\x9c\xe6\xd4r\x1a\xf6S\xe1\u0260\x11\x82yc(\xad\x99=A\x83\xa2\x8d\u00d9\xf3wb\xef\xfc]yK\x8a\xd0\f1\x88\rn\\\xbf\x81\xabW\xaf\xfe\xf0\xff\xfd\u007f\xfd\x9f\xbb\x00\xf0\xc5/\xfe\x11\xad\x8b\xf9z\xbd\xed\xeb\xe3\x1f\xff\x98\x03\x80\x1f\xf9\x91\x1f\xfb\xfc\xe5\xcbw}\xf5\xfc\xf9\xdb\x

d1w\u02fe\xa4Uy\xd7\xe3\xfa\xb3\x8f\xa2?\xdeGc\x1b\x18\x04\xaaWY\f\xc5C|_\x99\x11U\x02\xf5
\xd8\xed\xc6G\xa5\xe8\xdac\x92\x85\x9diA"\x95\xd7I*(\xa5wy\x18\xf4\xd1P\xc82R@\u0190\x02\x
19\x83~G\xf3\u03a4\x1a\x8cR\xb4\x9c\xf5\x99\u007f\x1e\xfd[\x9a\x90\xc0\x13\xa7\xadTl<\xa3\x8
5\xfc\x1b\x01\x9aO\x83(d\x84:B\xb5\x1b\x19I\x18\x86\x16p\x8bYJ\xa2*\u018d11X:\x81\r\xb2\xfd\x
a4\xac\x8c[aAK\xadh\x84+\u01bac\x9b#N\t|\xaeOP+\xe8Q\x18jK\xf5\xb3\x87\u035b\t\x9eB\x91M]
=\x03\xc4\xca\xfd\xe6P\xd0\xd9d?\x18\xc4
\xf1\u0708\x94\xe6]\x12/\br\xb7\x1d\x87\xd7>\xd0:M#\x98\x9e\xd9\xc6\u0785{a'\x1bJS,n\"6\x16\x
ce{\xbc\xf0\xc2\vw\xbd\xf0\xc2\xd5\xef\x02\x80\xef\xfd\xde\xef\x97u1_\xaf\xb7}\xbd\xe7=\x0f\xc9#\
x8f\xfc1\x11\xd1\xebw\xdcq\xfb\xffs\xee\xecYx\xef\vb\xb9>\xba7\xae<\x89\u064d7u\x10JE@3\xe7
\aL1G\xa5(\xd2\xe0!\x1d\x01Z\x93qU\xc4H\xe1\xb5sl\x8e\xbd\x8a\x8a\x06Tp):6re\xfaN\x81O\xd3\
xd8\xd9\xff4\xcd=U\x8a\xc8\xf4\xc5eDj*\x03)}`\xe7D\xb7\xc6f\xe6\xd0\x1e\xf5)\r\x88J\u067f\x94\x
d4I\x19\x96\xbd\u0453\xc0J!<EQ\x996G*`\x87\xf8y^\x031RwN\x12R\x89\x94r\x88\xc0\xaf&\u039c
tv\x01+\xf7\xab\xbf4\x19\xa3\xff!\xe7r\x96\x19Cc\xe3QZ\xf9\x93!\xec\x94\u007fPR\xe5T\xa1)\x18
H\xd0\b\x10\xcb\x19R\x01g\u061c\b
\x03b\xabPKpWK\x97\x8fJ\x1b\xe0\x9c\xf4\x14\xedv}\x14\xc6E\n\x14k\xf3b\f\xc1\xb4\f\xd7\b\xce\\
\xb8\x17[g\xce\u00c5\x04\xa2\xf8:\x8cm\xe0\x9d\xe0\xd5W_C\xd7/\u007f>\xfel\xff\xee\xdf\xfd\x9b
u1_\xaf\xb7\u007f}\xe4#\xdf#\x00\xf03\u007f\xeb?\xf9\u02fd\xbd\xbd\xeb\x8d1\x94\xe4\x99\xe1f\x
9f\xddx\x1dG\xaf^\x85\x13\x0f\xb0Y\xc1\xbf\xd3\xc3\xe9\x1d\xc4wY}W\xe9\xb0\v\x81J\x10\xc2\xe
815\x14\x1e\"\x8dm;V\xbbU\xc8\xd0]\xaf@/\xfa<8,\x83\x12\x12\xaf\xbb\xc4%\xe4\x14\xfa\x9fHE7\
x8c\x83\xac\x92Yq\xca>TI\xe9\xb9W\xf6\x8aI\x83\xcf\xfc\x99\x9c\xe4\xfbC\xea\xdc\x18LR\x1f\bV
>(m$\xcbPb\xbay\u05df\xbas\n\x19\x9e\u02c0\x9d\x17\x97\xc8\u012e<0\x91\xeaW*\x83\x83K,\xd
82\u0632CQ\xa7!\x14\xb4B\x16\\9\xfdH\xc1<\x1a\v\xd2(0\x12\x80\t$\xa1|3\a\xf8#\x8a\u07b8\xf0K
\xa6*\xfb4m\xac\x83\x81\x01\x11'\xd4=J\xffS\\!)\xa4c\r\xc1X\x02Y\x82\xa0\u01d9K\x97\xb1s\xf6N\
xf5t\x8e\u015c\t\xcc\x16\xc4\x16\xf3\xf9\x1cO>\xf9\xd4GD\xa4\x01\x80\xcf}\xees\xebb\xbe^\xb7n}
\xe8\xe1\x0f\xff\xdb\v\x17\xee|~oo\x17}\xdfgO\v\x01\xdcb\x86\xebW\x9e\x84[\xccA\u01aaA\u007f\
xc1Z\xa0\xf4Pz8\xd7\u00fb\x1e+\x86\xaaT\f\xa0\xca\xce8P\xfe\xa2\xb8\x86\x9d\xa0=\u0283\xc5\x
15\xf1G\x90\xf0\x93_\xc5P\xa4\xea\xe2\x8a?/\xa8\x8bc\xf6-
\xabX\xb6\xa4\x0en\x98@A\x85\x95\x00\xb9\x82\xe5\xe2\xf2\xd00\xc1+.G\xd9}ckL55\xb0\x88\x1
d\u0242\xab\x93DkH&BA\u9be3Mnd\x10\x85\x84\x9f(\xf6\xa2\xb7\u45a7\xe9\x04\xc6\u0520r\xca
ah\xe8\xb78\xfc)e\xa4\x87\x8fT\xd9\xf45\x83M\x04\a\xa8\xab<\x19\u0122\x9aN\f\x12\x06\xf4Qz_
@Q\xb2\x12*\x121s\xb5\xb5\xe0\xb09H\xe8V\x98\x04l\td\b\x04\x87\x8d\xdb\xce\xe0\u031d\x97al\
x1b\xf2I\xf3\xa9\xc2X\x8b\xc5b\x81'\x9ex\xe2\xd2\x17\xbe\xf0\a\xef\x06\x80\xbf\xff\xf7\xff\xfbu1_\
xaf\xb7\u007f=\xf3\u0313\xf1A8\xb9x\xf1\xe2\x1fmnm\xa6@\x80\xe8$\xd7-
\x17\xb8\xf6\xdcc\xe8f'*\x87\xa6\u0324\b'\xe2\x80C\u01a3jW\u3728\xbb\xd3\xe8R\u0211\xd5\u04
85\xa2\xe7\xb2_\xf7\xe4XM\x9ed\xa5\x98K\xea\x80\xe9\x14%\f\x8d1%\x86\xe1\b7+-
C*x]*\xb3\xfa\xb5\x80Q\u022b\xcb`\x12\xfb\xf8\x82\xc5\"\xa7\x15lz\x8b\xc2.\xab\x88\xd1h\\\xdb\x1
0Y\xa2jf\xaa8x\xac\x87\x14:q\x1fT\xa2\x8a\xa3\x93\u01ea-
\x02\x9dVvW\xdfJ\xa0\xa5R8\t\xaa\u079ep\x9a\xccW/:G\u0604\u00b9\xa0l\x00L\xf8\t\x9cT4\xcfT\
u04a5(\xe2\x12\xc2\u028bBO#\x9d\u007f\x86Z\x8a\xcf#\x82p\b\xb1f=\x8cRCZ\xe4\x01\x90e\x9c\
xbb|?\xa6[g\xe0\\\x9f\x8b9\x11\xd86Xv=^~\xf9\xe5\xc9g>\xf3\xd9O\xc5\xd7\xf8\x87\xdf\x04\xc1\x1
5\xebb\xfem\xbe\x1ex\xe0\xc1\xf4\xfeG?\xfa\xe1?\x9aN\xa7G\xc6\xc4\xcc\xc30\xc8\xf3\x1eG\xaf\\
\xc5\xc9\x1b/\xa9\x10\x83\xb9\xa8)\x02\x1f\x06R\xca=\xef\xe1]\a\x1fb\xb6\xea\xb0\xc7\xfc\x9c\x9

7\xf1g\xe4<\xa8\xf3\u025f\x1b\"\xe0e\xe8\xd0\x175\xf7\x90\x12\xcd/\x17M\x00xK5\xcc\x18>]\xb8\x14\xd6|\xc4\u0560\x86R\x0e\x9e\xads\v\nf\x18\x8e\xa6H\xa8M\xd9\xc2\x00\x00
\x00IDAT8)\xbcXV\xecIO\u00cf\xdf\u00bf@\xc6\xe1\u007f9\xa5\xa9\x97B\xa8e\x96>\xd1\x15c\x88E3\xd37\x13\x95\x95n\x84\xbb=\x80\x97\x86.\x8b\xa5W\xfa)\xad\xf6MN\x1e#\u007f\x9d\xfcS\f\xc8\x18u\xe2\fp\x8a0\xa7\xcf\xe3\xa1\xe3b8\x19\x96\xfe\xfbz\x8f\xac\xc2\\\x91\xd1R\xe3\xff\xf9\xdeT>>\xc1\x18F\xc3\x046\x04j(\xcfTHp\xf6]\xef\xc6\xd6\xde\xed\x99\tC\xf1k[x\x0f,\x16K\xbc\xf0\xc2\xd5_\x14\xd1v\xe4\a~\xe0c\xebb\xbe^\xb7n\xfd\xec\xcf\xfe\xed/\xdc\u007f\xdf\xfd\xf3\xad\xadM\xb8\xbe\x17}(\x14W\\\x9e\x1c\xe0\xe0\x85'\x01\xef\xc0\x86S\"\x83\x88\x04?\xf3l\xbe\xe5]\x97\xa0\x96\x88G\xe6\u0388*\x0eu6\xb6\xaa\x8b#\xa0I\xf0\xed\xb1\x03\xf7z\x02PEcY@G\xf0\xd813\xaa\xb1z)\x03\u0339\xc4\ah\x1c+\x18\xfa\x81Se\xbaNE\x96\xe5[\xe1*CE\x10\x9d\xf2\xe7\xc5\xffK|\xa8\xe2\xce\xe5\x01\xb0\x9c\xe67&\x85\xd4\xdf\x15)J\xc9#\aI\xf1\xba\xf2\xbdG\xf6\x17\x19\x19gF\xd8'\xceB\xeaKwsc\x97\x15\x9az(\xe4\xc4\f\x84\x82\xee\r%\xd8#\xcd-
\x80\"\x14e\xe0\u0452\x9c\xc5hpU\a\x11t\x81\x01C\x05\xc5QBRJct\xf0\u026d\x01Y*\xf2p\x1d\xb6\u039d\xc7\xf6\xd9\xf3`b\xddD\x8a\xae\xde6\xad\x1c\x1e\x1e\xe2\xd1G\x1f\xbd\xf8\xf9\xcf\u007f\xee\xaf}\xb3<\xdf\xebb\xfe\xef\xc1\xfa\xa7\xff\xf4\u007f\x8b7\xe3\v\x17/\xdc\xf9\xc5\xe9\xa4\xd5\xea\x1a\x8f\xad\x04t\xb3c\xbc\xf9\xcc\xd7\xe0\xbb\x05\x989\xfb\x92\xc7F,>\x04\u041b\xdd\xf7\u02d0+\xba\xda1V&V\xa5\xb1\xb6HV]\u01b4\xa2\x85Gs\uc495k\xeavob\xfe$\x83.x\xd4\xc1\xae<\xdd\xcbi8\xb5TM5%\xbea\u0465\xcap\xa3\xa8\xf39+S(\x19:\x0f\u07ac\x1b\x1f\xf9{\x1a\x9e.$'n\x14\x19\x11\xa7\xa62E\x05k\xf8\xa7v\xa16\xb3\x11\xb2\xc8\tH\xa74\xd3R\xf2UJ)S1\xad\xa6\xb7\x8a\x1b\x1a\xf9\xf3\xe1)
u\xe2\x041\f\xd70<S\x1d\x18]\\\xeaZ\x80\x95\u00da\xa9\bo\x8e\xf7A\u2167\xaf%\xa9\xc9\xd0_\x9b\xde\xef&2X\x1a\x03\x9ed{\x00\x0e\xa6sv\xd2\u2d8b\xf7\xc0\xb6Suk\xa4lf\u01a6\xa1\x93\xd9\x1cG\xc7Gw\xfc\u025f|\xe5\xa7\xd7\xc5|\xbdn\xd9\xfa\u0407>\x94\xde\u007f\xf0\xbd\xef\xfd\xe2d2\u0566&\xa5\xa2\x13|\xdf\xe3\xe0\xe5+\x98_\u007f]\x99\x03\xc9\xf1\x90\xaa\"\x1d\xe5\xfd\xcew5v>\f/(\x9eB*\xe2\xdbb\x06g\xe4g\xb3\x13\xb43\x87\xf6\u0105\xa1\xe7M\x9c\xb8V\xa8\u007fr\xd3#\xbe\xd0\xcd\x0e\xffR8\rR\r\xe5$\xde{\x194\x11_\x17UFP\xdf\x10\xbcpj\xbd\xa3\x9b\xd7\u0081mp\xec\x8c\xc7\x10\x9c$\xc4q\x92=r\"\xa7<\u57be\xb5\xbeE\xc6@\x962TCd\x10\xed<\xdc\xedFH\xe8%\xbd4P\x01\xc50\x84\b\xce0\x96\x1b\x16\x8e\xf5c\x98\x907;2'\xc8E\x9c\x03\xee\x1e\xf9\xe6\x8cA\xe0g\xb5\x1b\xe4\xc6D\u007f\u007f\u0094\xbd]\x02\x8b\xa5\x1a\xf6\x8aR\x16\xcf\xdcu/\u068d\xcd\xc4j\x89\xd1sD\x1c\u063a\x1eW\xaf^\xfd\u8e98\xaf\xd7-
[\xef\u007f\u007f>\t>\xf4\xde\xf7\xfe\xebK\x97.\xa1i,\xbc\xb8P\xd4\xf4A\x99\xdfx\x037^x*Q\xc4R\u0716\xe4\x87\"&O\x88\xef\xe1\xfb%\x10x\xe7\t\x8fL\x83\xa6\xa2\xc8\xfb\x12w\x8e\f\x11\x9f\x86\xa2\x14\x859\xc7N\xa9s\x812&\xb4*\nO\x1e\xd6TW{Yq\u05ebq\xf3\x9a\x98!#\x85\x8b\xc6an)7\x10\x1a\xc0\xf7rJm\x96Uo\x93\xb1#\xc3J=\x1c\x16w\x1a$,\x15\xc5\xea4\xdb\xdc\xc0\x8fO\xc3\xda\x12\x16\"T\xac\xf1\x15\xcf\xf2JDu\x9a\x1b\x1a\xa1\xb6>\xa7Qx\xa9d\xe0\xd0\xe0\x8ax\xd6\xee\xdc4\x16\xbce\x80-
\x86oY?\x87\x00c\x18&\f\xe2I2\u03fb\xf6\uf29b\x8c:\x03Q\u0237\x8dM\xc8*\xbd_\xcd}\xa2\x10\x89\u00d1\xcd\x1b\xf5\x04\x8d3\xa4\b\xd1\x10\x11v\xee\xb8\x18\xc4C9\x10#\xfeLM;\xa57^\u007f\x13\x8f?\xfe\u0136\x88l\x03\x99l\xb0.\xe6\xeb\xf5\xb6\xae/\u007f\xf9K\x04\x00?\xf6\xd7?\xfdg\xf7\xde{\xf7\xd3M\xd3hDVR\xdd1\xe6\x87\xfb\xb8\xf6\xdcc
qJ\x1b\xa3\x82\xf6\a\xa9\xfe\vx\x88\xef\xe0]W\xe0+EWKuzs\xec\xce5\xe5G\x87q\xa5l\x9ez\u581bE\xf0p\x19\xaad\xe8f\x92\x14Z%\x91\x8c\u06a7\x8eu\u01b2\x8aU\x8f\xd6\u027a\xf3\x8b\xf8\xf9

P\x12Ie'\xff\x8d,\xba9\x12S\xa6\xf7\xe4\x14zI\x85o\x05\xe2\"*\x04WaP\xeb\xc7\xe0\xa2\xe1\xf5\x93\xe2@D\xd5\ub5caX(\xab\xf6\x03\xdf\u0211$\x16X\xd6n\xb8\x9dXl\xec\xb6\xd8>\xdb\xe2\xccn\x8b\xad-
\x8b\xa6a\xb4\x86\u0476\x8c\xa6UL\xdb2\xea\xc0\x0fP\x96\xebG\x8f\x97\xb2s\xe7\x12#\xaf\xef\x17\x1f\u031c\r\v\x98\x00\t,\x96\xd8\x1c$\xff\xf3P\u0427{g\xd1n\x9d\xd1\xe6\"]\x1f=c\x18cp2\x9b\xe1\xf0\xf0\xf0\xae\xbf\xfa\xab\xbf\xf8\x00\x00\xfc\xf1\x1f\u007fi]\xcc\xd7\xeb\xed_\x1f\xfe\xf0G\x03M\x9b\xbaK\x97\xee\xfa\xbd\xe9t\xa2\x8f\xacw\xc1\x82\x9f\xd1\xcd\xe7\u063f\xfa4\x96\xfb\xaf\x83\xad\xc9).T\x14\xe4\xf8p\x06E\xa8s]H5/\xf3\u054ad\x9a\x92\xba\xe8\x8a.\xdd\xe70\t\ntE\xd3\x05VF\x80\n\x84Vs2\xa5\xd0\x12\xae\x18\xb7~\x03\xfc\xe9(u\x8c\xc3\u0541\x04i\xd0c\x16'\x848\x00\x1cD\xbfU\x1c\xc7R\x19:\xdc0d\x9c\xaaBc/\xa0\x1c\xfcRNQ\x1a\xe2\xff2,\xfcaX\xb8\xaa\x16-
\xa14\a\xac\x8d\u0525\fz\xc6MR\xb6o\x86\xb9W\x1ba\xcd!\x8f~\xf0d\bm\xcb\xd8\xd80\xd887\xc5\xd6\xde\x14\xbb\x9b\rv\xce4\xd8\u07b6\xd8h\x19\x8d\x01\x8c%\xd8\t\xc3\x1a\x0e\x8c\u0161\xd9\x0eU\xb9\xaf\\\x9aq\x85W\x10\xa3\x12\x95\x88$A\xb6Oh,\x83\x1b\x02O,LksFn\x81\xbd{\xefa\xa6\x1b\xd8>\u007fIgH\xdeW\x87\x18f\x03\xef\x81\xfd\xfd\xfd3\xbf\xf3;\xff\xe2\x12\x00\xfc\xb3\u007f\xf6\xdbx\xea\xa9\xc7\xd7\xc5|\xbdn\xddz\xcf{\xde\xf3G\x0f<\xf0\x80>`\xde\x05\x99\xa6b\xc8'o\xbe\x82\x83\xaf?\acl\xa8\u0252\x1f\xcb\x02\x1fW\xec\\\xe1\x96\xca|\xab:\xfe\x8f\xb4\x99\x1e\x95?y\xea\"]\xc6\xd1c.e,\x9c\xe9\x9fS\u0462\x0f3\xccV
\x90\\\xfch%\x03\x13\x99\xdf|\x1a\xb6]R\xe2X\v\x87p\x8ex\x13*,\vF\v\x1c\u0764
\xca*\x82Q\xc09\xaa\xf5\xa1J\x81+\x18\fr\x19\xc5F\xa3\xd7\xc93\xe0\rF\xc83\xc5FC#\xdb\x16\x8d\x1b\x98\xd1\n5h\xc0R\x1a3\xa3/60\x1a\x02YD\xb0\x960i\x18v\x8baw,l\xdb\xc0\xb0\x81\x99\x18\xd8M\x06\x1bE\xc4\r+\xef\x9b-\x83,\xaf\xe0\xe1U\xe6I:\xb5D\x99~hB0
\x06\x91\x84\xd9+\x81\f\x83\xdb\u0415\x97A\xd8\xe1g\xf0\xe2\xc1\xd6b\xe7\xceK*\xa4\xf3!\xa0%\u078fLX.\xbbn:\xdd\u061b\xcdN>\x01\x00?\xf1\x13?\xce\xefy\xcfC\xebb\xbe^o\xff\xfa\x87\xff\xf0\xd7\t\x00\xee\xbd\xf7\xdeG.^\xbc\xf8jc\x9bp\xe3\xea\x83\xc0\xd6\xe0\xe4\xfa\x1b\xd8\u007f\xf1Y\x18\u02e3]nv\xec\b\x959\xf0\xce!\xbe\xa0r\xdf\xe4\x1c.\xb5\xba3\xc2.)\xa9'\f\xeb\xa2\x19\x94\x84\x81\x95\u0400!B\xb4\xc2\x1d\xaf_\xaf\u0704\x0f]\xff\xd5X0\x03\r`\tg\x19\xae\x89\xafC\x8a\r\x81\x060\r\x12\x1bf\x15\xe2\x19\xf1\xfd\xa5\xba\x97EQP\xca\x19\xc4\xf0\u02d4\xbe&\xf1\xfa(;$~\\\x0fM\x85W7\x9b\xb1\xba^l)+\x93\x05\x1a\x83\u042bv\u007f\x15\x98!\xca\xef3k17\x13\x02\xefZ\xa0\t\x12\"c\xd5\x17h\xa3\x05\xb5\x16`\x1dp2\xb4\x93G\xfc\xfd\x13\xaa\xe3I9\xbbH\x94\xc4D\x81)\xb4\xa3\xe1\xda\x10)\x83\x85\x98@\r\ubc35\x8a\r,\f\xf5E@\xc6`\xe7\xf6\xf3`k\x03\xa6N\x15m\xd36\x8d\\\xbf~\x1dO=\xf5\xcc\x1e\x00\xfc\xf2/\xff\x1d\u007fxx}]\xcc\xd7\xeb\xed_?\xfe\xe3\x9f\x16\x00\xf8\u0527\xfe\xa3\x97\u03de\xbd\xed\xf7766re\x10\xbd\x1d\x96\xb3\x13\x1c\xbc|\x05\xdd\xecP]\xe9Rvc=K\f\xed9\xe0{H\x10\x11\x95\xb2\xe7\x8a\xee7r&\u03f0\x01\xe5\x80\xe5\x10t\x11y\xd3\xe4D\v\x91\x89l\x8a\xd3<ds\x95!\x19)\xd1R\xcf\xeah\x80\x03\x97\x90,\xa1\x86O\x84\xf5\xe5\xf5\ra\xb9\xc1\x81\x0f]\f\xe3\x88\u01bboz+\\y\x05W)\xe0eZ\x05\xb9K\u0590\xe4\xa1f\xec\xca]\xa3\\m\xcf\x04grO,\x85h\xaaJB*\x06\x895\a\xb5\xf4\xbf\xc1*]\x92\x86\x8a\xdb\u04f0\xad\xfaZ\x90!\xd8F=Px\u04c2\xa6&\x83Y\xcc0M\x03\xbb5\x85\u0759\xc2\xd8&\xd9\u0732!\x98\x86\xd57%\xe5\xf7\x15\x82\xa9(\xe5/\xb3EC\xb2\x90@\n\xd1\x18\x85\u0344U\xb6\xdfj\xd7\u03e7x\xb0\x8b\b\x8c\xb1\u063e\xedv\xd8v2\xfa\xb3\xc589\x10\xbdWD\xee\x05\x80\xa7\x9f~\x86\xd7\xc5|\xbd\xde\xf6\xf5\xe0\x83\xef\v\xdd\x12\xcd.^\xbc\xf4\xe5\xbb\xef\xbe\f\xef\x1c(\x8e\x86\xc2\xc4~\xff\xe5+8|\xed%\u0626\xa9\x8e\xcb5\xc3!\n3\x02\xd4\xe2\xba\x11\xbe7\xad\x1e\xc1\xa5\xd8\x1cbw\x9e\xe8\x8bH\x1e`\xc9\xe40\xc0\ae\x9ed\xecH\x890\xda[\x8f\xa5\xe3\xe4\xd7P3+F\x91!\xa9s%\x84\x00\xdad\u041e\

x05My\x80\x9d\xbfU\xd1\x1e\xc3$V\xedmW\x93\xe6J\x04x\x10\xedS\x86'\x93@,\xc1[\x86\xb7\xe1\xf5\x18(\\a\x8aN\x1d\x94\x8a\x9a0\xadz\xbc\x9fB6\x1c\x8d\x8e\x1b\xaa\x80\xb0z]\x87\u078a\xb1(\x9b\xa9\x01m\x1b\xc0\xd2\xc0g\x8c`\xda\x06vw\x03vk\x03\xb6\x9d\x80m\x03f\x03\xd3\x18\xd8\u05a4\xa1<\x06\x8c\xa6\x12\xaa\x12\x91\xaa[\x8fX\xb9@\x87\x9e\x86\xb5\x90SC\x85\xdch\xe8p\xa9\x1b\x9ca\xc2\xc6\xd6.\x9av\xa3\n\xc8\u0223!\xa2\xae\xebqrt|\xf6\xd5W_>\a\x00\x8f=\xfe\u013a3_\xaf[\xb3\xbe\xf6\xb5\xaf\x12\x00\xdcs\xcf\xe5G\xee\xba\ubbbe,\xb0\"\x0eD\x8c\xfdW_\xc0\xe1+\u03e3i\x9a\xe0c\x11\x1e\x8eA\u01d6\xb2\x15\xc5\xc3E\x9f\x8c\x02\xfa
.`\x05\xa2\xd5\xe3\xfc
\u0693nR\xefF}\xb4G>Y\x04+_|\xb5\u03d5S\x99}eU\x8f\xe2\x92\u0254\xb1\xb9\xa7C\xba\u026e\x85m\xb9:\xfa\v\x8d\x9f\x16\b7{\xd1R\xc3\xff\x03\xa3/*w4:E\x84\x946\xa7|\r\x1a&lX\xc6dj\xd0L\x18\xb6\xe54|\xf4\xa1\xb8\xa7\xa4
\x1a\xbb\xe8\xe3q|+\xe3\x8f\x11\xf5\xd1\xc0\xb9<'%\x05\xac\xdcX\x02m\x19\xd0\x06\xe3\x14!+hja6\x1b\x98I\x8bf2E\xd3Na\xdb\x16Mka\f\xd73\x0f*7\xf6\xe8\x8cX\x9b~I\xc8\x00U\u02a3r\u02a9\xd5\u007f[Z\xd5P\xe0\x9e\x97,\x18\x81\xc0\xb4-
\xectR\x04GSv\xdc\x14\b\x11\xe1\xe5W^Y\xfe\xee\xef\xfe\xee\x12\x00\xbe\xf4\x0e2Z\xd6\xc5\xfc\u07f3\xf5\x81\x0f|\x87\x00\xc0/\xfc\xc2/]\xdd\xdd\xdd\xfd\xc3\xcd\xcd)\xbc\x12\xcaCm
,\x0e\xf7\xb1\xff\xea\x15\xf8\xff\x9f\xbd7\x8f\xb7\xac*\xafE\xc7\xf7\u0379\xd6\xde\xfbtTA!\x14\x05\x04J\x1a\xa1
\x16\x02\u04bdkh\x05\x1b\x82\x1a\x8d\xb9\x11\x8d\x06\tQ\xaf\xbe\xe73\xd7\x17\xafz\x93\x97\u0118\xe4&\xf16&1/O\xa3\x98\\rc\xf3\x8c\x82J\xb0\uf0c8
J+}[\rU\xa7\xdf\u035as~\xf7\x8f9\xe7Zs\xad\xbdNU\xde\xfd\x03\xcab\xcf\xdf\xef\xfc\xaa\xea\xd49\xbb_c~s|\xe3\x1b\xc3\f\xfdh\u007f\xfa\xf1\x16$\xa6G>$\xd7\x06\xaf\xe8\xaa\x11Jc\xd2i\x8c\x01\xb6\xd4*\xd3v\xb8\x93\x8a\x86AK\x15\x9c\x0e5\xa1\xc9SW\x8d2i&a\xb4\x13\x1d5\xb9_\xa5\xbc
\xe4\aht\xa6\x15\xba\x8a\u0419Q\xe8L+hE\x15\x17\xbd\x06\xc3R\u056bM\x93\u0644\x8a\x925\x0e\x13\xf1\xb5^\x83\u0268\xf9e\x85\xc6q\x87\x19\u074c\x90+\x82\xee1\xb29\x8d\xbc\xc3\xc8\x14\x95\x94QJ\xbbT\x81\r\xb2\x16\x19^\xc7\xf5V\xdf\xf5&]EcL\x98\xe2\x00\xa4=\x06\xcd(\u007f2h\v\xe0p\xfeT\x81N\xa0BXC\xe9\x0e\xb2\xbc\x87\xac\xd3\x01)\xed\xfd[h\xfc\xb5\xabT3TV\xfa\xc4\xec\x11Ny\x1f\x16\xad\x19\x943\x90q\u057f\x89\xb2D\xa1\xaa\x91\x1a\xab|\xeb\x13\x86T\xd6I\x9e\x8e\x84&h\xa5\xdc\x1a\x8eF|\xff\xfd\xf73\x00\xec\u0639s\x02\xe6\x93\xf5\u052d#\x8e8\\\x13\xd1#\xb3\xb3\xb37\x1cu\xd4Q0\xa3\x91Kydk\n\xcc?z/\x06\v;\xa1X\xfb\x12&U\x12D\x02@\xa8\x94qE\x99b*\xe1\x1a\xaf\x9c\x93d\x99\x9a\xe2\xa4}n?\x1eiym\x81t{\xc4|\"ii\x83\xee\xb5\x15\x8c\x95\xcf5sP]Li\xe8\x0342M\xd0\x04\xe4\x1d\x86\x9e\xd3P\x1d\xdf@s%\x8d\x91p4h\xabZe\x1c\xf2ZmZb\xe5\xdc\xd0\xc0K\x8b\x9fJ\xc2\xd9g\x8a\xd0U\x80V\x04\xd6\x04\x9e\u0460\xf5\x19t\x87\x91k\x86V\xa1R\xe6\u051aaO\xc6_\xe3\xc8\x1d\x1d\"\xd3MZ\xf6\xd0\x12\x90\xa0*\xc92\xf2\u0291i\x05\xear\xcdc\xb1^\xea\x87[\xeb\x06\xc0-
\x15\x88\nJu\xc1y\xc7\xef\f@K\xb0E\x9d\xb6\x8a\xcf\u0445MKE3\xad\x9c|`\aPV\xe2>\x1d\xd1\x17&\"\x95\x04\xd79\x81\x90\x82\xeat\xab\t\xe0z\xe6(\x0f\x87#\x1cr\u0221\xcf9\xe6\xd8cO\x04\x80\x13O<\x91&`>YO\xd9z\xc3\x1b\xde@\x00p\xcc1\xc7\u0733n\xdd:\x80\xa0D\x9cP\x19@\xc0\xd8\xf5\xf8\xfdX\x99\u07c6,\xefTC\x87\x89\xee<\x1e\u0465\xf4\x96\xb65\x99\"\xb5\x8d\xaaK+\xea\xd6\x12\u071b\xd3\xf5.\x02\x9b\x932\x84\xbaV\xd1\u05fa\x9aT\x13z\x8f_\xe6k_g5W\x12\x02\xb2,4\xecf\x14\xb8\xabj\xb9\x11\xaa\xa7\xa0\xa6T5\x04\xd3\x12j\xbc7xlS+J\xed\x90\"\xf5\x83K\xa4\v$\xc5s\xffxY\x13::\xd2\x04\xf0^#\xd3\njZA\xf5\x14t\x06t\xb4\x0fapLp\u068f\xd3\xd7\x01y\xcf\xc3U\xe

5\xc9E*\x8bc\xec\u1d4d\xbd\x06\xa5\x00\xce\x18\xd4U~\u0693\x9atT\x83\xaa\x12\x01r\xf2\xd59\
xa7\x16\x02\x04h\rR:\x8c\xf3S\x12#\xc7`j\x8e\xf4'v2\x01\xb8UG\x81s5v\x1ap\xceS\x85Rg\t}\xea\x
96R\xe0\xbc\xeb\x1fZ\f\x9c\x8e\xb1\x89DT\x14\x85\x9d\x9d\x9b\xa5^o\xea(\x00\x98\x9b\x9b\x9b\
x80\xf9d=uk\u077a\x03\x1c\x00\x9cw\xde/\xdc\xc1\u0337\xe7y\x0eg\v\x11g|\xe3Gi,\xee|\f\xbb\xb7
=\xe8\xa5cTM\xdd\u0544\x89R\xc9\x1a\x05\xce'\x11\x89\tm\xa7\x14\x8c\u0491Lj\x16\xeb5\xee
6\x16IU\xe0\xe4\x1d\x15\xa9\x15\x1c\xc7i\x1a)9\x01\x19s\u05d6\xf6\f\xe5\u06a4\x8f\xe7vuF\xe0\
x0e\x81g\x94\x97\u01e5A\u011a\xa0\xa6\x95\a\xa9\xb4\xa1\xb8FSQ\u068e\x0e-
\x9e\\5\xb9dT\x0fQ5\x06Z{\x96\xe4\x1b\u00e4\x80NF\xd0\u0468J3h\x86A9\x81\t\xe0\xf085\xc3\x
d3/Lp\n\xb0:q*DK?\xa39\xfa\xb4\xc6ND\x8d\u05d6\x92\x97\x93\x13\x9e\x9a\xa7\x14\xa8\xc3\xe1\
x90Q\x97s\x96\xbe)\xb1\x9bA\x04\xea2D\xa1\x8cx\xf3AQ\x1cr\xf0\xc2\u0670T\x1f\xb5\xbc\xfe\x8
cr\xa8\x88\x99\xa02\xdf\xf8\x14\xaa\xac\x1f\xc4:\xd8d\xb2\x13hX\x00K\xa8\xcc\xf3^5\x80\x84T\xc
1\xe4\xc7\xfe\xad1\x18\r}\xf8\xeah4\x9a\xd0,\x93\xf5\u052ds\xce9\xc7\x01\xc0)\xa7\x9cv\xffA\a\x1
d\xf4\xc0\xec\xcc\f\x8c1\u039a\"8!\x12\xcch\x88\xed\x0f\u078ea\u007f\x1e*\xcb\u00a7%H\xbe\\\x
e5']\x06\x068\a\x17\xc2+\x90D\xb6\x8d\x83\x9c\xd4t\xd4\xf5\u03a4\xd7\x00\xebL!\xef0\xf2\x8e\x1
f\u91a2\xbaN\xba\xea?%\xf2f\xaa\xf3\xc92>u\xb9&\xa9\x90L\x161\x03:\xf3\x928\x9a\u0460\x9e*
G\xe3\xd3\xdf\xd5S\x8c\u038cB\x96yN\u0595\xd4\u0178;\xc9\xda>-
\rZ\x02i;\x93jz\u9997U\u0515g\x9a\xa1\x94\x97\xfa0\x13\xb8\xcb@O\x87\x14\x1f@r\x0f\x8c\xc4\x
04M\x9e*\"\xf2rO\xa7\xa9\xee\xf9\xbe'=\xa5\x94n\r{5*\x88'\x18\xad\xbd\x82\x85\xbb\xcaS,\x94\x0
6\x83H\t\x98\xfe-
Hc\x00}u\x9e\u6002Q\x99c\xc5Sa\u0512K\xf2z5\xe8'o\xc9\xe2)\x16Q(\xbb\x9eN\x04\xce\xf9l\xd0\
xd42\xb7\xa2p<\xedb\x84\xa0:\xbd$\x90C\xea\xc2\"\x11\xb2\u01a2(F\x02\x00\xa6(&`>YO\xdd:\u3
333\u4aab~C\x11\xd1\u02a6M\x9b\xee:l\xd3a0\u0188\xb3\x06\u058c\xe0\x9c\x05k\x8d\xc7\xef\xf
d!\x16w=\x06\xdd\xe9\xc2\x11`\x93(\xb3\x94\xe2\xf0
\xee\x12C\xa2\xb5\x9cI\xea\xf6\xb3c\xf0!^>\x96k\xdf\xc0\u02fb\xaa\xa4\x0f\\\x18\x84\x19\x93\xc6
Q\u04c0\xbc\xc1?\xb7\xc8\xeeh\rV_\xe0\xc7\xc8YyP\xe4Y\xed'-
\x93\xe9\u0258\x80\u011a\x90\xcfhtr\x86\xd6\x04\xd1T\x8d\x9c'\r`\u0683\x9aE\x92>\xa04\x8d\xb
bHZ&\xe5+\xc0\x12\xf6\x01\xc4Y^\xa90\xa8C\xa0\x19\xe5\xb9\xe18\x80\xc5\x04\xf4\xfc\xf7\x88\x
c3\xfb\xe7\x82\x12I\x13D\xb7\xbc\n\xb2vw\x81\xd0\xf4\xea\x19\u007fK\\\xb2)\xaa\x0eC\xcdy\xae\
u0725\x80\x99(^\xa89\xd1+~\x03\xa7\x8c\xc0\x91\xdf&\xa0\xa3\b:\xba+&'\x99\xca\x06(\xbc\a\xa1\
x8a\x8e\xbf\xc8\xda\xd36\xfe\xbd\x8c6\xb9\xa86\x11\xa9'\xa1\xc6\x13\xd1\xc8:Xbd\u0769R\xcbN\
xa9\xe82\xa6T\x85\xd4#\xff#n\x02\xe6\x93\xf5\u052e#\x8f<\xd2\x01\xc0\xb1\xc7\x1e\xfb/Z\xeb\x9
5,\x8c\x83\xfa\xe1\x1f\vf\x85\u015d\x8fa\xfb\x83\xb7\xc3:\x03\xce\x14(\v\xfa\xdc\xd8LK\xa4g^\xa2
hC\xac\x9c\xad\u05ffM[\xd2\x16
M\x1cR\xbdicO\x81\xa7\xb5\a\x84\x10\xc2.k\x17\x8d\xb5\xbf\x8c+\xa6\xa5\xb5:o\xdeH\xa4\x05\x
b4\"\xcf\xefv\xa8.QoT\xa6\xdcc\xa8)\xafl\x81\"X\x15\u031b\x90\xd2Bk\xf9\x9cP\xc3tP\xd6d\x9fR\x
18\r4/\x90\x11\xb2\x8e\xaf\xca\x19!\xfalZ\x81z\xaa\x9a\xf0\f\x0f\x96;\f\xee(
$F\x95U\xb6\n\x03F\rm\xfd\xf8\xabKk\x9cp$y.u\u0263V\xecO8=\x06\x05\xae\x9c\x92\xa8\xc2\xf4
P4.\xa4\xf1o\x06u\xd9[\xd4R\x10\xa5\xc0\xf7?)\x9eLP\xc9\x05k\x9f5\xf6\xef\a8|V\u00c0\x10$-
\"\xa8Rf\xc1\xf7c$\xfa\xbd\x130\xb4\x82\u0551\x83\x03\x81\xb3,\xf9]j|\xac\xbc1\x1c1=\xed\xd7\xf
4\x04\u031f\xa1\xeb\xcc3\xbd\r\xf3\x9b\xdf\xfc\x96\x1f\xaf_\u007f\xe0\x83\xddn7T-
\x0eb\x8d\u709d\xc5\x03\xb7}\r\xfd\xe5\x1d\xc8z=pF\x90\x8e\x82\xcb\bN{\x1f\x10\x1f\xfb\x85r\x0
4\xdaK\x14M9\xfe\\\xa75\xc6\xf5\xe0m\x93\x98\x1c\x9ax4\xa7\xc0=\x0e2\xc0\xa4:o\xd8\xd9\x12\u

045e
~\x8f\xf5qZ\u1ae0\x85\xe6\x1e\x033\x1e\xfc\xaa\r\"5\x8f\rn}\x99o4R\xee\x83\x16\x9c\n\x13\xa2\xdc\u019f7\x06i\xa8v\xa6G\xb3!\u062c<\u02de\x02\x00\xc9\b\xbaC\u0202\xf2\x8e\x18\xa0@e\xa0\x1e\xcf\xe9\xa1J\x11\\\x8fa\b\xb0\xae\xbe)\xfaf\xe8\xb8\u007f\xb8\xb4\x9e[\xc6#D\x9b\x9a\x17\x9f\f\xe77\x1a\xee\x12hN\x039\x97\xbf(\x8d\x91\u007f\xa1\x9a\x8d|\xf5w\x11\xff{\xe1I\x92?\x02\xfa\xd70\xce>p\xdd\n\xd8KE\xeb\xd6\xc9\x16\xe13\x8a\xd8@\x0f\x95x\x00\xef\xf8\x15\xef\xdcY\xa0?tX\xec[\x14\xd6\u007f\x86I\xe9F\x82\x91\xff\U000948cf]t\xd6w\u775bT\xe6\x93\xf5\x14\xaf\xf3\u03ff0\xba(\xfed\xfd\xfau?\u0770a\x03\xac\xb5\xe2]\xf7\x83\"\x85\x15\x9e|\xf8n,l\xbf\xcf\x1b\xf8+\xf6\x13\x99D\xb0\f\xd8\f\xb0\xba2w\x92\xc0=W\x9as\xc1\xda\x01\fI\x85\x9aL\xceHP\x88P\xa0:hFA\xe5\fR\xf1\x18\xbd7\xb6\xb6\x9c)\u017f\u05a7U\x82\xdeXi\x02g\x04L)\xa0\xa3\xead1\u01951\x04?\x19*]*9WI\xc3\x15\x924\xf9\xba\xfa\x1c5\x0f\x1bYc\u04d1f\x0f\x00\xa1\x19\x9c\xb3w\xfe\x8bw\xa1}\xf5K\x19\x97A\x105\x13*\b\\N0\x8a`\u04f4\xa1\x14`\xa9\xfd\xddi\xdb\xf4Z\u01cc\u04aa<#p\x87\x81\x19\r\x9aR\xa8\xfb\xa47NQ2~\x88+\x1fZ9\xe0\x13\xdfW\xff\x99\x92`(\u6e1a\ue31f\x1b\x0f\u8f8fc\xadC!\x82\u00b9Z\xfb\xa44\xe5\x92J\x8e(\"0\xd6ayP`a\xa5\xc0\xa8\xf0\tMD\xde;\xc67\xfc%\t\x88\x0e-Sq\x12\xa8\x97,\\O\x135\xcbd=\xb5+\xfd\u031dx\xe2\t;gg\xe7 \"\xe4-C\xfd\\=+\x85a\u007f\x05\x0f\xff\u4ef0v\b\xd6Y\xa9.\x88\xb2A\x973\xac\xe6Ro]F\xd1%\xdaqjuR\x94\x1aE[ce\xb9\xb27\xa5)\x055\xa3\xa03\xafn\x88\x95W\x1d`\xd0N\u153a\xe0\xbd\x1bps\x1cl\xe9\x84\nW%\xb7/T\x9bq\xa9\xf5\"\x15\x83\xa75(\xe3j\u011f\xd1Ry\x03k\xfd\x8d\xd6<Q\x8c\xd3,\xa2}\u056b9\xe8\xa5\x15\x01S\f\xf4T\xf3e\xadQX\x96\t\x83\x8c`y\x8di\u073d`P\x05a\x82f\xeef\xfa\xa8\x15\x03y\xe6O74\xa7\xab\xb1\xfd\xe6)\x85ZB\xa4\xa4\xa5E\xd0\t\x05D\xb0\x9appp\xe4`\x15\xc1\xe4T:E:\x9d\x18\x8cY\x813\x0e\x16\x80!\xc0\xd80\xa1\x1cA\xbc\xe4\xcc\xfdk\xe1\x9c`0tXZ\xb5XZ\xb506\x1e4%\xc4\xd4E)dj\x17\xe0\xdf\\fR\xd6YdZ\xef\xda\xcb\x1b8\x01\xf3\xc9z*\xf8\xf3#~\xa0\x14\x17Z\xab\xa4j\xb1
Rp\xce\xe2\xf1{\u007f\x84\xfe\xfc\x0e0\x14\xb8\x1c\xfc\xf1U\x95U\x84\xd14\xc3LQHm\tIDh6\xbb\x1aR@\u051b~5\xea3p\xcf>4\x83\xa0f4\xf2)\x15T#\t\xf0T\xa7\xe6\x86lN*\x15H\v\xc5\xd36d\u009a\xc19{\xf9\\\x87\xf7\x98\x19\x9dn\x16\x04\x01\xf7\x18\u064c\xaa,\xb7\xa3\xbae\xec\xfe\xd6\xf2i,\xdd\xd3\xf7t|\x800Aw\x152\xe5\a\xa9\x88\b\x92\x93\xaf\x80U5);v\x86\x10\u007f\xa5\xbb.\xc3i\xef\xc4S\x029\xd57\u023dGUS\xe3tB5\x1f.\xad\x83zeV\x83:\x9c\xf0\u048d\xd3F\xe3\xce\xd6\x1cD\u04be\xdav\xe2`\xc5\xc1\x88\xf8\u07ae\x02l\x87\xe1\xf2\xe0K\u00fe\xf9\u8303-\x1c\x9c\x15\x18\b\xac\b\xcc\xc8a4t\xb0&j\xca\x05\xd6\n\x8c\x15\f\x06\x0e\x8b+\x06\v+\x06\xab\x03\v\x91@\xb7\u01e0\n88[T\xba$\xaaNP\xd69\xe9t;\u0635{\xd7\x03\xfdA\xff\x16\x00\u063e}\xbb\x9b\x80\xf9d=m\xeb\xdcs\u03fbijjzg\xa7\xd3A)7ta\u0519\xbd-\xee\x8e{o\v\x12=F\x9a,L\xd6\a\u43ba\x04\xa3\xf7\xe0\xf3]K\x9f\x97\x96\x01\xa2\xca\xdd\xce{\x88$\x84j\x87\x90\xcd)\xe8\x8e\x0f\xfeu\x8d\xea\x97\x1a\xb7O\x89-\x89\xec\x85b\t:\xe4\xdc7\u0290S\xf0\xcfn\x199\x97\xf6\x84#\u0584,\xd0Aec\x98\x81\xf1\x12to\xdc\xfe\x9e+c\u02bd\u05ca\x8a\xaf?\u00df\nr\xaei\xb5\xdb6\r\x06!\xeby\x8e?\xea\xf7K\x1fo^\xab\x12o\xb6\x8eS9\u0478\u007f\xbc\x1f\xcca\u042c\x02\xa6\xfdI!\x86HS\xb2\x11Hi\xd4YW\xec\xa4\xf3_\xf1?\x1c\x93?\xf9\xc1\xf7s\xac8\xff\xfeG\x13\xb1\x98\x02d\x1c\xc4\xf8\x11|'\x9e\x901\x01\xe0\v#\x18\x0e-
F\x85Aa\x1c\x8c\x15\x14F\xb0\xb2j\xb0\xb4R\xa0?p0\xae\x1a\xed\x0f\x01\xb9\xfe\xb1:\a[\x8cJs8F\x95\xc0%\u21080\u055b\xeao\u077a\xb5\x0f\x00\x87\x1f~\xf8\x843\x9f\xac\xa7o\x1du\xd4\xe6\x1b7m:l\xc7\xdc\xdc\x01\xb0\xd6W\xd5\xe2,\x98\x00\x9de\x18\xae.\xe1\xf1\x9f\xde\x02\x16\v\u03

6aVF\xb1\x03\xf4\u04075\xdb\xcc\xf3\u65774j\xe0\"\u0524F\xa4V\xa9I\xfcD\xaa\xc47<\xdc\x17Okd3\n\u0228\xd6\xe8\xaa\xe1\f\x8d\xfbp\xcbX5\x99\xd6\xc2\xe1\xd7\x19\xe0\x8c@\n\xbe2\xe7\x16\xf0M\x1c\x1e\x9b@\xcc\xe4\x95-
\xd2\xe3\xaa\u008d\x8a\x8a1\u007f,\u00b8\xba\x86Z\xcf\x0f)\xb0\x11\x13\xb2\x0eA\xa9\xa0\x9bf\x02z\xec\xf9}\x1e\xefC\xa4\xb5\xb3\xf8\x97\x14\xbd\x8c\u045b\xd3\xe8L)t\x82>=\x866H\x8b\x87\x0e\xed\xf5TQ\x1f\xb5\xcd4AM\xb1\xaf\xca3\xaeM\xbf\x8fW\xf4\xf5\x9eE\x9bs\xbb\x930\x18\x95y\xf9eMJn\x05\u0537\x90\x91\x85\x14\x0e\x12\xaa\xf1H\xa1\x18\xe5i@\n!\xcc\u00d1\xc3p\xe80\x1a9\x8c\n\x87\xfe\xd0a0r\xb0\x12\xde\u007f\x92@[\x95.\xbb~\xc3q\x16f\xb0Z\xf5\x17\xe2)\t\x028G\x04\xe0\xd0C\x0f\xd5\xcf\u007f\xfeY\x1a\x00\xb6n\xdd:\x01\xf3\xc9z\xea\xd7\xea\xeaR\u025fo\xde|\xf4\xad\xddn\xb7\xf4\x87vA/\xabX\xc1Z\x83\xf9m\x0fb\xb0\xb2\ryG#SA\xcd!\x04v\x04e\x04\xd9\xd0_\xfdN\xc5\xea\u070d\xe5\x97Q\xb3\t\x88(WKR\x85\xd8?\x1e!\n\xd3\xfbA/\xad\b\xd9\\\x06\xdd\xf5\xd5y=\x88\x81\u01a3gbe-
\xcd\u071b\x06\xe5C\x04\xce\xd9Osj?%X\xaf\x14\xc7C\x96K\x15\rR\xab^\x82\xe91l\x90\xff\xf9F\xf1\u06aa\x90\xf4d\x10gU\x05kW\xc8\x1c\x1c\x10uTph\x02M\xd79\xe9\xb5n!\x16\x9b\x19y\a\xc8|Z!\xd7\u0790K\x01Aw\x1e\xbcT\x80\x16E\v\xb5l\x9a\t\x81\x15$\x9dY7\xf0\xe4=\xae\x01\xb4\xb4%\x12\u057a\xa6\x91\xb0\xa1\xb1S\x013Aw\xbd<\x95\t\xd0\xca\a\x99\xa0o\x81\xbe\x81+\x9c7\u010a\x95\xbe\x00\x8e\x04\x96\x91\x18i\x01\xd6\t\x06#A\u007f\xe8\xd0\x1f8\x8c\x8c\v\x1e\xe7^\xd6\xc9YH5\xe2\xaa\x1f\xa0\x14C\xac\xc1p1\x04N4\xe4\x87\xceY\xd7\xebu\xe1\x9c\xdc\x0f\xe0^_\x99o\x9ap\xe6\x93\xf5\u052f\xa9\xa9\xd9\xf2\xef[\xb7n\xfd\xfa\xd4TO\xb4\u05be\xba\x15\a'\x16\xcc\f\xa52,=\xf9\x04\xb6=r;\xf4\\\x86,gh\x95\x00\xad\x13pa\xc1C\vK\x0eF\t\f\xd5u\xbbcQo\xa9~;\xb5\xa4\xd5\x04d\x1e\x10kV,\"\xd0S\f\xbd>\x87\xed\xb2\xd7GG:\xa6v\x1f\xd2N\x8d\a\xb4\x8a\xfe&e\x85\xa7\x025\xa0B\xfa\x8c\xa65=KJxJ\x9bg%\xb5OP=\x05\x9aU\xde\xe8JUz\xfc\xb6\a\xd4v\u016fE\xc2pF\xd0S\f\xa5\xa9<I\xc4\xe9\xd4:\xf7\xd3\xc2S40Xk\x82\x9e\t\x92O\xae\u0326\xa2\x13\xa1$\x00M\x8d\x8a|L\x16\x9aL\x8ef\x9a\xc0\xb3\x9e+\aS\xddL\xad\xa5\xd3Y\u007f\xc8U\xd5-
\x8d\x82_qP\x181\x81!\xc8\bPp\x80q\x80\xf1\x9f\xb1\xd2R\"d}\x9a\xa8\xa7\a\u00a0\x92\u007f\x8e\xd6\t\xac\x00.9m(\xe5\x03+\x94R\x89j&l\x80Z\xc3\x16C\xf4wm\xf3\xb2HR5%\x901F\xe6\xe6\x0e\xc0`0\xb8\x8b\x88\x16^\xf4\xa2\x17\xd1s\x9e\xb3e\x02\xe6\x93\xf5\xf4\xae\x8b.\xba\xf0[\x9b6m\xea\xfbJ#L\xb39\u007f\x06\xd5Y\x86\xc1\xd2\x02\xb6?t7\xa8\xe7)\x0f\xa5\xfdD\xa1\x8b`\x12\x02\x9a\xc5:X\x16/\x0f#i\xaf\xeej\xff\xac\x1aK^\x06\x16\xbc<\x12<\x90\u0130.\x9bUP3\x1aN\a\u007f\x91\xf0\xe5\xe5\x80M0\xa3J\xcb^\xa59\u051a}\xde\x03&\\\xc7\x1d.\xb5\xe5cvYM\xb5E\x02\x98\x04@30\xdde\u032e\xcb05\xad\xd0\xcd\u067b.\xeax\x8c\a@\xe3\x94\n\xed\x85='\x00\xaa\xe3\apJ\xaf\xf3\x0e{\xc9\x1f\xb7\x19\x8b7\xa7\u007f\xa8\xc6r\xabP\xe5\xa3\xe7\x03\"4\xc3\xdb\xfc\xe6\xec\x83\x1bH\xb0\xc6\x1b5\xf6\xc0\xe2T\xacb@\a\xf5\n\xe5\\\xdf\\\xa4\nInk_4\x03M<
\xfb\xa5\x90\xf4<4\x95\xbe\xe5*L\x8dJ#W\x02\xc1\xb3\u0769\xd4\u0650\xab\x93\x14\xa3\xac\u0115\"?\x90\xa6\x19\x14\x9c\x18\xabM-
(X\x98P\xf4\x971\\\x9a\xf7\x8d\xfd\x84\xa4\x13q0\xa6\x90\x03\x0e\x98\u00f1\xc7\x1e\x93\x01\xc0\x05\x17\x9c\xff\xb4N\x0eM\xc0\xfc\x19\xbe>\xf5\xa9O\x00\x00\x0e<\xf0\u0ece>\xfa\xe8Gz\xbd\xa9\xd2\x19.\uabd9\x15\x9c5Xx\xfca\xac.<\t5\x97\x83\xa7\x95\xf7\x87\xa6DE\x1d\x91\xd7Yx]\x8b\v\u0375Jw\u0754\xe1J\xf4\x91F\"\xb5K.\xc6t\x06C\xc4k\xc1;s\x1a\bt\x8bS\x1c\x86\x98\x1aa\xcb\xdc\xc6\xf3\xd63D}\xfed\x18\xf7\u05be\xf9\u064c\x19m\xe3\xaf\u04d4\xe5X\xf12\x019\x13z3\n\x9dYo\x99\x9be\xdeF\u05c3\xba\xf7\x15'\xaa\xb3\xe5BX\xdb\xfa@<_\xac:\\\x05\xd1k\xf2~1\x19\xd5\xf8\vj4&\xdb\xe8\xee\x9a\xe8\xa7\xe7\xd5;y\xc6\xe8jF\xce~\xf2U1\x1a\xfd\x8c\xb4\xdf\xd14m\xa

7\xd2aR\xcdi\u040c\xae\xc8\xee\xb4\x0e\x97\xba\x15\xa646E\x8a:\xf1\x04\x94H\xbc\xc4\x10\x85\xf3\x95wF\x1e\xd0\u00f0\x90K\xbd\xde\xc5\x1bq9\x82\x9f\u007f\xa8\x92\x80|SVy\xb5\x92\xf7\xdca\xbfy+x\xbf\xf3P\xb4H\xd2]\xf1\xc6\\\f\x81C\u007f\xd7v\x14\xfd\x95\x00\xf0\x15\xc5\x146!\xdd\xe9\xe48\xfc\xf0\u00df\x04\x80\xa5\xa5%\x99\x80\xf9d=m\xeb\xe7\u007f\xfe\xe4\xf2\x83\xbfn\u077aok\xad\xc1\xcc\xe5
\x85\x04/sR\nK;\xb6a\xf7#\xf7C\xe7\x19x\x9a\xa1\xa7uh\xa2I5=\n\xf1\xde\xe6\xc5\b\u018d\xe0\xc4V\xa1\x01\x81\u05e6\xf4\x9c\x9e$u1{p\x85\xae\x18\xe4\x88\xe5Qz\xae\x00t\xa7\x14\xbas\xba\xfc\xf4\nU\xea\x86X\x02S\x8b\xdab\xac\xc2$T\x94H\x1e\x00=\xa1d\x9aus\x9b7z\xfc\xa9r
\x91\xbd\x15\x80\uacbf\xed\x8c=8\x04\x1d\xbbR\xe3qw\xed.\xe8~\u0491;\xde\xf5\u0417\xaaa\xc4}\x8a\x13\xfb\xef\u0118\xab1\x8b\x19\x03\x8e\x13\u0303u\x1ek9\xf7\xb6\xb4J\x91\xa72\xbc=;\x14q\u02f3\x8b\xef\x1fjS\x9b\x9e\xd3\x06\xf4\x8c\x02\xcde\x1el\x91p\xea\xd4n\xddPg\x81\xa8\xfcQ\n\x95\xb8\x98\xa0N\xb1\x0e\u0384\x93F\xb0\xc5\x15M\x18\x8a\xc3\x10\xe2O\x80\xe1\xb9\xfbA6\x82\xe1\xc4m\x92P\x16\b:X\x003W\x95\x81\x88\vCC\u0276\x15\x1d;\xb5\x82X\x83\xc5\xc7\x1f\xf4j\x16nD&:\x8b,\u02f8(\xcc\xee\xf5\xeb\xd7\u007f\x1f\x00\xf2<\xc7\x04\xcc'\xebi[\xc7\x1e\xfb\x9c\xf2\xef\x1b6l\xf8\xce\xd4T/\xc4sUm(\"\x06\xeb\f+\xbb\xb7c\xd7#\xf7\x81Y\x815C\xcdh\xa8\x9e\xaa\x86)\"\u8245\xb5#\x98\xa2\x8fQ\xb1\nk\x87\xde\xf3\xc5\xd9\xd2]Q\u0106/W~1I\x98\xf4\fc\xd2\xc1x\xabn\x80\xe4\x03\x18z\xebrds\xda\xda\x15\xcc\xfa\x00\x00
\x00IDAT\xcb\xd4J>\xc2S(\xd4\xdauk\xe14\x94\xaf|\xbd\x8b\x93o\x80\xd6+Q\x19\x9bYoU:\x96\x9bT\x00\xa5.{\xfe8\xfa\x83D\x80\xe1\x90A\u065a\xd8\xd3(\xa6\xa3\xe41\xe7\xb0?y\xfa\tS\u028f\xb8'`\xd8\u6552\x9ef\xcaSN\xf2\xbf\xc2~z\x95\xf3\x8af\xa2\xb1I\x9e\xda!\xa1\u04a4'3\xa6Y\x97\xc1\xeb3`\xca\xd3@\x92h\xddkV\xb7\x90*\x90[\u0195;\x12\x93}\x8ad\u01f1\x01t\x9d\xaf\xdc]NXq\x82%\xe30\xd4@\x911LF0\xda\xcbb\x8d\x06\xa0\xb8z\xadu\bmV\xfe\xab\fW\xa1\u051f?$\r\x05\xc7N\xef\xb2K
\xa5`\x8b\x11\xe6\x1f\xbc\x1b\xb6\x18\xfa\x1f\f\xca\x18\t\xee\xa0J)\x1c|\xf0\xc1\xc3\xcb.\xbbt\x01\x00\xce>\xfb\u0327\xf5Z\xd6\x138\x9b\xac\xb8\xba\xdd\xfc\u01a3\x8e:\n\x8f>\xfa\x18\x86\xc3a\xe9\x12GD\x10(\x8c\xfa\xab\x98\u007f\xe2\x11\x8c\xfa\xab\xfe\xfb\xcaA\xcd*P\xe1\xe0\xfa6\xc4o\x95\x1a\\X;\x80X\x03
\x03\x93\x06\xb3\x06\x83\xcb\v>\xba\x06z\x05\x01Cu\x14\x9c\x10`\x05\xb0\x9e\x1b\x85b\by\xef\x13J<NX\x13:\xeb2\xb8\xa1\x83]\xb1~4\x1b\x16\x10Sj\x9b\x01\n\xa1\x05\xc1\a\xbb\x96_\x1a+0\nc\xe3\\\vs\x1e\xab\x96\x13\xa3\xf1T&\x1f\xd5?\"\x94\u0394xN\xba\xe3\x00c=\b\a\u037bP\xe5\xe1\x1d\xd52c\x03VqX*4f\x81@\v\x84\xfc\xcc\xd2\xc6\x11I\x0fI)\x82T\xf1~M\xd59%\x9b\x12\xe5\ft\x150t\xe5k\xe2'|\x1but\x12^MU+\x02Z\x13\xb2\x034x.4=\x1dJ\xce]Z\xec\u039c\xf3\x14\f\xd9T@*\x89s\xa4\x84\xea2q5\f7\xe0\x9c\xc3\xca\xc0`\xbeo0$\xff\x9e9\xe7sf\xbd\xef\xb9\xe7\xe7)\xe1\xc9#`G\xc6Mhl\u05b8z)\xc3\xc90\xca>Ii\xac\xee|\x02\xcb\xdb\x1e\x868\a\xa5\xf32\x95HB\xf3s\xe3!\u03e2\xf5\xeb\xd7\xfft\xfd\xfa\r_\a\x80\x13O<Q&`>YO\xeb\x1a\x0eW\xd1\xe9La\u02d6\x93\x16n\xbd\xf5G\xcb\xff\xf2/zf8\x18\x02\xf0\u04c2\x95\xf3-ai\xc7\xe3X~r\x1bf7l\x84+\x86\xd0\x19!\x9f\xd3\xe8;\x81\x1b\xc5\x04\x1a\x02\xc8\xf9\v\xac\x18\xa1pE\x88\xfe\xd2 \xe8\xb2\u05258%)\xe4+'\u0470\x8e\x80\x91\x03\x93?:\xc3V\x01\x10D\xf5`\u0342l\xda\u00ac\x8e`\n\a\xb2\x0e,>$\x83\\\xa4\x05\x18\f\x05\"\r0WG|\xf6\xea\x15\"\x04\x8a\x85\x92\xe0\x04)C\x12j2:T\x91b\u0360\x8bT\x16)B@\xc6\x1ex\x87\xce\xff\xdb\x010\x91\xa2\xf0\x95\xa6OWJ#\x86\xaa\xe3>\xa9\xb0\xc1\xc4\u02b1\xc3\xc0\xb4\xcf\u03ecB\x9fi\x9cFj\x99\xb8tR\x0fe\x02\x00\x0e\xaf\x81\xf4\x18n\xd9\u07ce\x15\x81qRZ\xfd\x8e7=\x93\xb4#&d\xd3\nj]\xe6O\n\xae

\x9e\xa3]\xca\x12]\xe0\xb3\x1dPX\x815\x0e\x19\x01\xb9B\xeb\b\u007f\x93\x11s\x02\x14\x85\xc5\xe2b\x81]\xf3#\f\v\a\x8aJ'?\xc5S\xc6\xd3\xc1Q2\x1c\xdc4O\xf0^+U?\xb7\xaa\u04a3\xcbbT\x1f)\xad\x01g\xb0\xed\xf6\x1b1X\xd8\x05V\xc1\x1f>\xbc\x98\xe2,\x9c\xb54w\xc0\x1c6o\xde\xfc#\n;\xd8!\x87\x1c6\xa1Y&\xeb\xe9]\x9d\xce\x14\x00\xe0\xbc\xf3.\xd8\x01\xd0u\x9dN\x0eV\\\x82m\xe4]\x89\x19K\xdb\x1f\xc3\xf2\xce'\xa0t\x06\x12\x81\"\xa0\xdbc\xe8\x19U\x1an\x95Lp9@d\xe1l\x01S\f`\x8aU\x14\xa3U\x183\x805C\x883\xc12\xd7\x02\xec d\x01k
\xa6\x803\x05l\xe1\xfft\xa3\x11\xccp\b;\x1c\xc2\x0e\a\xb0\xc3\x01\xa4\x18B\xe5\x06\xbak\xe0\xdc\b\xd6\x19X\xb1!\xe9\xc8\x02A^im\x01kG~,[ll\u00b3\xf6\xc7j\u0242\u0331\u64ce\xb2I\n\x12P\xcd\xc3[\xea\xfcC\xb3\x92\x95\xa4:\xcf\"7\x14\x9c\x1f\x11}\xd1\xdb\u87e4Q\x9c1\xc2\xfe\xe3m\x87g5\x90Q-\x10[Z\x06\x98\x9a
\xe6d<r\"\xd5\xd3HF>s\x93+\x93\xb3fFv(\xa8=\r\xe2\x04&4b\xb3\u0408\xf6z?\a8\xaf\xf9v\xf1\xcb\xf8?\xad\x11\xd80\u0623\x9c\x94\xdav\xa4^\xee)\xb1\x1f\xfen\x1d0\x18Y<\xb9{\x88\x1dO\xf61\x18\xda\xd0@\x0e\x8d\xf9\xa8|R\x04V\x9e6\x8b7A\xd2\xd0\rIr4\t\x96\xe7\xcc\xc1\xc6!\x86\x91\x10\xcaD\xa1\x9dw\u0782\xc7n\xfa\x1a\xec\xa8\xef]\x13\x9d\xc0Yo\xeflM\x81<\xd7\xc8\xf3|\xfe\xf4\xd3O\xbda_\xb9\x8e'`>YI\xe1E\xab\x87\x1f~\xf8]\xd3\xd3\xd3
f\xdf\xedOB\x1eH+\xac\xcc\xef\xc4\xf2\xce\xed\xbe\x92\x0e\x95\x17A\xd0\xe9)`JyYXc\xea\xaf\xdc\x10\x9c\x83u\x06\xce\x198[\xc0Y\x03gM\x00\xd8
gd$\xf3\xde!\x8a.\x06_XS\xe6\x8c:k\xe1\x8c\x05\x91 \x9fV\xc8z\xbe\x8a-
\x9d\xf4\xa2\xe1W\xb0*ub\xe1\u0107N\x1bW\xc0\xb1\v`\xc0a|\xbf\x81\xadR\x85mV\xb4\n\xd5hri\xce*\xa5\xb1\xcde\xd3NU\u0464\n\xa5\xea\xa2\rZkWe\x96D\xf5M)\xcf\xe9\x97\x1cxJ\xe4\x8f;\x9f\x8c\xa9E\x9a\x8e\x84)\xf5\x92\xfa\x86k\xaf\xba\xa9\xf4\xf3!\xb2-\xe8\xb7]\xf4-
d@\xf5\x18\x1cl\x82\xe3\xd4e\xf0g\xf3TMi-
+\xc1\x06B\x90\xb1\x0f\x98P\x8d\x87]\xfe3\xa1\x87\x00`4\xb2\xd8=?\xc2\xee\xf9\x01\x8a\xc2\xf8p\x8d\x92\xd3\xf6@L*\x84N$\xc0]?\x85T\xb6\xb9\x84DvJ\x95\x92\x89\x98\xfdO\t@J\xa3?\xff$\x1e\xb9\xe9+Xy\xf2q\xb0\xca\xfc\t\xc1\x85\xfe\x8e\xb30\xa6\xc0\xf4\xf44\x0e;\xec\xb0]\x97]\xf6\x8aoN\xc0|\xb2\xf6\xc9U\x14\u0177\x0e\xdbxX\xe0\x1d\x15\xca(s\x02\x88\x14\xcc`\x80\x95\xf9\x9d0\xa3Q\xc0Z_\x8du\x14\u041bQp\xb9\x97\f6r\x18*\u03d6\b\x80%\xd8\xfa\xafJ\x96X\xaf\n\u02f3vR\xc6E\xa0\xf6Gg\x01\xe7\x84\u03acB7\x8c\xbb;F\xd9\x18\x15\b\x1c\v\xac\n\x80$\x0eV\f\x80\x02\x8a\x02\x9f\x9dq\xbdQW\x17\x0f\x06k\x01jt\x12\xa9\xe5\xe2I\xa3\xd0B\xdaE\x8f*_t\xf6\x80.M\fo\x94\xcd\xd1+>\x86P\xd0\x14\x97\x89GT\x93\x82\xd4E\xd6\u037e/5.\xf0Z*O\xec\xefR\xe0\xcecfh\xfa:\xa3\xca\x1a\x05|\xd5\xee\u008d\xaaYU\xf3)\x8f\xef\x8b\rU\xb9\xb5\xc1'\xc5\xfa\u05eb\x9e(\xd5|\xd5\xc2\xf0\x8f\x8b\x01I\x82\xfe\xd0b\xe7\xee\x02\xf3\v#8k\xc1\\\x8d\u06f3\xf6\xcd\xe1\xb2\xe3*\xf5\xbc\xd9J8%\x89\u02c3\x84\x8a\u07a7/)\xed7\x030\u0574\xee\xc4\n\xfd\xdd\u06f1\xf8\xe8}\x10k\x03\xd0Gu\x90\x83\xb5\x06\xa6(033\x83\xe3\x8e;\xf6>\"z\x12\x00>\xf9\xc9\u007f\x9c\x80\xf9d\xed\x1b\xeb\xee\xbb\xef\x04\x00l9\xe9\xa4\xe5\u00cf8\xbc\xac\u0309\t\x04\xf6U\"\xfcD\xdd\xf2\xae\x1d\x18,-&\x86C\xfeb\xec*`jF\x01yt\xe6\xab<?\xa2b\xa0\xf4\u00e6\x84\x91
\xef\x8bR\x02y\xb8@\tU2M\xac\xb0\xa5q4'\x110\t\xb2)Fg\x9a\xd1\xc9}x\xafP\xb0\xe8%\t_\u0443\u011f\x00\x98\x05\x10\x03a\x1b\xae\x82\xc6\xd0\xd1\xd8\xe8z;\x10U\x99\xa6\x8dvc\xfc\xb9h\xde\x15\xa3\xdd\xc2\xc8x\xaa3O\xd17\x86\\\x18'>\xa6o\xba\nBn\xe1e\xb0v\nP\xb2WP\xeb\xccO\xf5=E !P\x18\x8d'\x178\xf51J\u0217\xb5\xa6\xcb0]\x8eA;\xfe\xfdp\x80\xb5a\u042c\x11\xfc
\xe2*Z\x05\rN[\x00q\x95n\xdc9\xc1j\xdfb\xe7\xae\x02\x8b+\x05\x00\a\u05be\xe1\xcd\x01\x80)\x18a\x89s\x15\x98\xa7\x13f\xd13\x87\xabH=\x0e\x95\xb8\xd2\x04\x9d{\xa9\"\x10\xc2:\x90\xc8E\x89`

G\x03\x98a?)*$4a\r\x8ab\x88\xa8\xf0\u06bcy\xf3W\xe2\xcf\x1c}\xf4\xd1\x130\x9f\xac}c\x1dw\x9c\x97(\x9eq\u05bfy\xa8?\x18}+\xeftA\xac\xaa\xd4rT\x8d\xa3\x95\x85y\fVV\xc2\xff\a\b\vWc'gt\xa7\x19\xc4\xf0\xa6]e\xe8n\xc2\xf1\x12jy\x90e&X);\b\x00
\x15\x18\x94\x19\x8b\xb1rNS_\xc4\x1f\xbfy\x8a\xa1;\x04\x1d\x8e\xe3.1\xfdB\xd4\xc0\x97G\xec\b\x9e\x1e\xe0\xd3@\aB\xaa\x85oC\xf1*\xe0Y\x1az\xe9f\xba{\xa9\r\x0f\xf7\xe7\x0f\x02T\x02e\x19\x87GU\x15lC\x83\x8f;>v\r-
@\xdc\x04ri\x99\x19\xaa\x019\xd5O;)xQ!\xa0U\vg=\xfd\x9d\bw\x12\x1e\xdd\u007f\x99\x0ec\xd0e\xac\x1a\x81)B\x90w\xe0\u015dqej\x0f\xa5\xe1\x13\xc9\xccB\xd4us\xa05\xac\x93\xe05\xee?\x13\xab}\x8b\x9d\v\x05\x96\x87\x16\xc4\x02V\xe2#\xf1\xcaCb\xf2\x9eK#\xfd\x87jm\x87\xca)\x80\t\xa4\x18:S\xc8r\r\x1d\xfcW\x1cE\xe9hu2c\xa51Z^\x84\u9bc2X%/\x9a\x835~~\"\x16'\x87\x1ez\xc8\xee\xf8:\x9fz\xea\xe9O\xfb5<Q\xb3LVmm\xfe\xb9\xc3W\x0e\u07b8iG\xa7\xd3\xf3i+H\xfdH\xfc\xc51Z]\xc1h8,\xb5\x85\x92\x14\xd4 \x87\xbcC\xa0\x19F\u007f\xb7\xf7\x97\xa6\x94d\x0e\x90\xcee\xc3-
H\xc2\xe2\xd0OHN\xe7`mZ\x8dL\xfaQs!\xaa\x0f\u02e4\u0702\x06\xa8KP}\x81\x90\x85%\x01\a~\x87\xc4_\xbcB\x14\xb4\xeb\f\xb0@THV\xa2\n\u009bp9V\xf3\x8e\xd1\x04\x95d\xb2\xa6\U0010ea99)\x05\xacz\xa4d\x10\x14\x8b\al\xa0\x94\xf4\xc5M\xcdi\xff\x18\xb3\x9c\xa0\xa6\xb5\x0f\x8a\x96=m+U\x00H\xfaR4\a6\xd3\xef\x8d%\r\x15\x0en`1\n\xde%\x14\xbd\x14\xd2<M\x006#\x8c\xa6\x18\x05\x03\xcb\xcb\x06d\x1cf\xba\nq,\x81bt
\xca~h\x99\xe6\xe3l\u0578TD\xe5\xb0\x12B\x0fAD0\x18Z\xcc/\x1b\f\v_\x89\x97\xea\x11\x9bn\xf0\xae|l\xc4\xfe\x9f\xd4\xf0\xe3\xe1Pi\xc7\x0e\x06\xeb0\x05J\xd1&\x82|\xcf6\u0775\xa2\xd6\\,V\xb6=\x84\xe1\xf2<tw\xaa\fhv\xc1\n7\xfe\xbb\xd7\xebbyy\xe5\xdf\x02\xf8\u043er\xedN\xc0|\xb2\x9ak\xe3\x99\xe7\xfc\xc2\xf3n\xb8\xe1+\x18\xec^\x00)\xa9\x1d\xca\t\xf0\xc7\xd0b\x94\x9cj\xa5\xacV$\\h\xd9\x14C\x8c\xc2p\xd1\xc0\x19_\x98\nU\xb3\x99\x02\x0f\x12\xc2\xe4A\x98\xbd\u04e2D\xb3\xc5p\x81ID\xbaRYR\xcf\u03d4\xb2J\x0e\ua64e\x80r\x01\x0f,\n\xe3e\x82\x04\x80\x8c\x1fqt\x01y\x9c\x05\x9c\xb0\xd7);\v\x04\xa3\xa5\xb4\xc8\x1b\xeb3\"\xd5i7\"\xdfd<yG\x82|\x8e\u00b4%\x86\x9e\xdea\x0e\x15\xb9\x15\x90P\xd5\xf4e\xae\xd4$]\x86\xebr\xb2\xc1\xb4\xa5%U\xe1\xc4\xcdnb\x9b\x86\x9d\x1a*\x15q\x80\x85\xa0\x00`\x00X+!1\u067fY\x92\xb8?\x1aM\x18t\t\x05\t\xa4\xf02\xc3\xdd#\x87~\xdfzj\x8b\x83\xb5C\xf0\n7\xc6\xc1F\xee\\\xa4\xb4e\x88\xaf\x9bV\xc0TW!\xcb\x18B\x04c\x1dV\x87\x0e\x85\x15\xef`\tW\xa3i<\x90W\x80\x1d\x9fY\xe9?\x1e\x92\xaa#`\xab`M\x10?\x1a*I\x1ct`8\xe7B#\x9a\u02a2Du:X\xd9\xf18\xe6\x1f\xba\xa7\xe6\xea(\xe2\x81\u071aQ\xad\xfd\xb0c\xc7\xceC\xf7\xa5\vw\x02\xe6\x93\xd5\\\x1b6\x1cr\xd8\xcfM\u036e\u00ee]\xbb\xa1\xda2\x1a\x9d\xf5\x04)b~f:\x89R\x01A6\xad\x00\x03\x8cV-
\xac\xf5\u0382`\x02\xd9\x18\xca\xeb]\x0f\x89\x01\xe1\xd0\u0434\xfeZq\x89\x93_\xd9\x04%jL\xbe\t\x85\x0e\xa7\b\x06\xa8G\xc8\xfa\x04[8\x18\xe7*P3\x14\x1a\xae\xec+G\xb2Pb!VA\x94\xf3\xda\xeeXR\v%\xc0\x910\xe2\xd2R\xea\n\x1aTTR\xf5\x89\xa7\b\xa4\xab
+6\u8ba5\xa2XJ<\xa1\x10\x8b\u7a56aF\xe8\xaad\x8a\x9ch\xccy2Up\xb4\r\x06\xc5\x1d\x86\x1a^\ua945\x8e\xf8\xb8\xb4\xc5\xc2\xc1u\t\xd3C\x82\xf2\u0495\xe0sB\xb0\x00\n%\x18d\x84B\x04(\xbc:%\u01ad\xad\x1a\xa0\xdf\x0f\xef\xa1\x04\x9f\x13\x97l\xb6\x89\xc1\x16Q\x95\x02e\x1d0\xb2\x02\xa5<%\xe7\xc2\v\xad\xf2
\x89uRN\bc\xcc\xf5>(e\x92\xa0\x13\nt\x8ab\nn\x90\u0546\x16\xdd\x12\xe3g\u060a\xef\xa1P\xa2,\x15'Py\x17\x8b\x8f?\x80\xc5G\xef\x87\xeet\xcaW\xcb9\vS\f\x83\x84\xd6?\x15c\f\xb2,\xfb\xf1\x04\xcc'k_^J\xb2\x1e\xf4\xd4\\\xf0Z\xf1\x17%S\f\x8c@p\x94c8i\xda\xc4V\xc0\xc6
\x9fW9\xa3\x00\x11\x8c\xfa^\xe2\xc6\xde\xc9\xc8W\xa4H\x94-

\x11rC\x05fC,\x1a\xc7H\x9c\xb8W\u0109\xcbF\x8ee\x82
\x90\fP=B\xc7\x10\xdcH\xbc\xe2F\x82\x1b^\x11\xa7\x0e\t\x02\vk\rP\x10lt\xe3\x8b<)5\xaba\x97T\x
b8\xf5\xd9~\u007f\"\xa9\xa8\x89V+\x98N\xb0\xf6-
<\xdd\xe38\x86\x02KH&\"HHj\xb2\x9a\xd1\xe9*d\xaa\xf1\x1c\xc7\xfeY\xa5\xdeT\xed\aJz\vi\xd6k\x
e5\xd3\x02\x01\x8a\xc2ai\xd5`q\xd5\xfaA\x9c.\xc1A\xa13p`\xeb\xfb\rF\x01\x85\x16\x8c2\xf2\xcd\x
d88^\x1fv\xb5\xb8\xb9\t\x04b\x1a\x94\x17*0\xadK#C\x89\x1c6'\x87\xca\x10\x8b\x94o\x14W\x95\x
b8\x94\xf7S}\xb4\xd2\xcd\x01e3\x99\x88\xa1\u0201(\x9c|\xe2\xe7\x96c\x8f&\xc8B\x9d\x8f\x92\xa
b\x1c\xd4}\xf3\x95\x15\u00cd\x06\xd8\xfd\xc0\x9d\x18,\xecB\x16\f\xe7\x9c\bL1\x825E\xf9\xd2+\xa
5h\u00c6\r\u0634\u9c3f\u0717.\xdcI\x03t\xb2\x00\x00\x8f\xce\xfb\xa0\x8a!\xe0\n\x95Cwz(5\xbb\x9
cp\xc2\"P\x9d\x1eDe(\x8c\r\x9ab\x8c!\x98\xa7?\x04\xdc!\xe8\x1eC+\u007f\xaeO\x8d\x8d\xa8\x9cL
ta`\xc8+\x14\x9c\xf3\x15\xb5\xb1\x16\xc6\x1aX\xe3\x1bm\xb1\x11\uab03\x18\xaf?/\xbf\x1f\xa5\x8
e6<\x98\x1eCu\t\xb9\xf6r;\u07d8\x8b\x9e\x1fA\x12I^\u007fn\x8d\x81\x1d\x8d<'Z\xb3i$\xa4\u0588\
xd1\xcc*V\xa0\xf1\xc9:\x91\x96\x89\xc3\xe6\x16\xe9\xf3:\x05\x04\xcb>\x0e-
R$\xc2\xc1\xf5\x11\xde,J\xcf0f\xa78\xf8VI\xbd\xa1\x97\x10+^bQM\xf5P\xc9Q\a\xba*\xe9@F\xc1\x
87\xb3\x82\xe5\x15\x83\xed\xbbGxr~\x84A\xdfxgB+X\xe9\x10\xe6g\x15\x16f\x18\xcb=\xc6j\a\xe8\
xe7\x04\x13\x8f\x1aQ1b\x05b\x91\f\v\xc5\x10\x90\xcay\x90\x12\xba\xa7\xd4}\x97\xaed\x95N\\\xe5
\f\x95\xb1\xf7\x92WTy\xe2\xc4Ppjf\xc4&\xee\x9a\xd1KE\a5\x14\xa5\xf9\x11\x1e\xe0]\x98\x9aR\xe
1\xe4`k\x01\u05b1\xa0\x10\xe8N\x17\u02cf=\x80\x9dw\xdc\f\x8aY\xb8\x10?\xec6\x1a\u052a\xf2\x
03\x0e8\x00'\x9d\xb4\xe5;\xaf~\xf5\xbf\xfd2\x00|\xf4\xa3\x1f\xa1}\xe1\x1a\x9eT\xe6\x93\x05\x00\x
d8=\xf4\u0235\x00\x18C\x1aJg\xb59\xf5*\xf4\u01c2\xbb\xb3(\xa8\x8b\xa20\xc8\x1a\xc1\x03\xa9C\
xac\x8b\x00\x94\x937\xb4\n\xea\x87\xf2\x8c\v\x01\xbb\xc8!W\xbc\xafq\xc0\xd0z\x10\u0395
S\xec+kix\x17Z4-
I\x93O\xb6\x1fUW#\x87\xac\xf0\xaa\x898\xaa\x0f\x17=Fbc\xcd\xc2Y\x02F#@\x00\x95\xe5\xe5
\t\t\xb5\xba\xff\x95`\xbf\xb6J\xb0^F3\xbc\x9e]Y\xb8\"!=\x92\xecN\xc7\x00:\x8c\xee\xb4*#\xdd(\x95\
xcd5n[\x12\xa9f\xa4\x87\xbc\x9dkh\x16\xbb\n\x83\xe1\x04##X\\5X^\xb1\x18\x8dl\t\xc2\x12\x1f_\xd
c\x178\xfe\x9b*\xf0\x1e\ubf3a\xc6\xf7\xa4\xb2\u0265\x1a\xabS*JJ
O\xaa\xf0*\x14\xa3\xf1\xbc\x82&<Rg\xf1=#\xe6d\xca\u022b`\xc8\x01J\x04~~\x88\u02ea\xbctc\x88
\x9f\xc7t\x82@\xa4\xf2M\xcfr\xb8\xa2\xc0\x8e\xbbo\xc1\xf2\xceGAJ\xfb\xcd\xdb\x1a\x14\xa3\x01\
x9c-
jo\xe7\xf1\xc7\x1f\x87\xe3\x8f?\xfe\x0f\xe2\xbf_\xff\xfa_\x97}\xe1\x1a\x9eT\xe6\x93\x05\x008\xe9\
x909\x01\x80\xc5\x02\x87\xac\x16\u05bb\x1c&\\e\xea\xe67}\xe0\xc1\x98=`\x0e\x19,\xfc\xc0\xa0\x
8cM\x19\xa6L\x05g\xec\xdd\x15\x93\x19\x13?\xfa\xed\xbc$\x90\xaba!\x02\xa1p\xc0ja\xe1D\xbc\x
cb
P\x1b\xa7\xafM2\xa6G\xf2x\xec\x0f\x15\x1du\xfc`\x8b\n\x16\xb4%\xe7\xea\x82Wv\x1c,r\x02\t\x13\
xa5\xa5\x85@\xac\xd0\x13\x80\xaa\xf5^\x81Z\xd5Xk,\xb6`\xb9
\x04EG\xdb\x00[\xe9\xa2)\x9c\x1a\x1c\x00\x97\x05\xcfsS9GJS\xb7m-
\x9c\xf5\xd5t)\xd1s\xbe\x81\xe9\xa5\u05c2\xe1\xd0ai\xb1\xc0\xd2B\x81\u0565\x02\xf3\xf3#<\xb1\
xad\x8f]\xbbF\x18\rmU\xd1\xd7N\x15\xb1\xd2\x0fQl\xc6A\x8c++\u007f\x89\xa7\xaa8\xa9\x19\x95L
I\xf0\x06q\xeap\x15@[3(S\xe0\x8e\x86\xca\x158S\xde.\x82\x93\xbe\x04\xd59\xff\xb2\xefL\xd5f@\
x9a\xbd\\3\xe3\u048a\x81\xe0\u9c4c\t\x8a\xbc\xdd0\az\xceIU\xad\xdb`\x0f
\xa9\x1ds\xb8/\x95eX}\xf2\tl\xbf\xe3\ap\xd6\xfaq0\x11\x98b\b3\x1cT\x9f\x05\x00sss\u0632e\u02d7
\xdf\xf1\x8e\u007f\xff\x15\x00\xb8\xfa\uafe5}\xe5\x1a\x9eT\xe6\x93U\x81\xce5o\xd6w\x14\xb8\xa0

\xdf_\xc1\xea\xe2n?\x05\x1a\xab\xbf0m\xa7\xf3\x1e\xd6\x1f\xba\t\xb33S(V\x17\x92z\x80\x12n\x93\xc6\f\x94J3\xab\x91\xe7\xaf-
\x00\x16\xef\xa0'\xd6gF\x92\xf8\xe6\u0520\xb0`\x89\xa3\xdfA\xd9\x10\xaap\"\x19\xdb5$\xb8;\x96\xd1a.\xc9\x15\xed2\xa8O`\xeb\xedt]\u0d8dq\u022c@9\a\x10{G=\xb1p\x96\x01\xb6
\x1b\xf8[\xa8\xa0\xac\xa9\xaah\x91\u0532@jsBi\xa5]\x9d\x16\xa8\fM\xb6\x10\x14Q\xb7\xe7*\xab\x00G\x80Q\x02\xeb\x04f\xe4\xfc\xeb\x03\x86\x0e\xbc\xb9+cM\xbd\xd4/N:FS\xb0\xa8\xe9\x87\x00\u00e1\xc3\xc2r\x81\xc1\xc0y\x8d6\x80Q\xe1S\xe9Yy~:r\xdd\xe5\xfb\x14\xabo\x97\x94\xdb\x12o]\xaa\xed<\xf5\x1dK\xacl\xab\xc9\u02eaI\u0351\x0e\xa1h\x18V\xff\xfd\xaa\xf9\x9b\xf4\"\xa8R3\tG\x99d\x90\r\x86\u03e0\xd8\U0001e2c0\xe1\x87\xc6R\x99\xb9$\x9b0\xc37zm\x943\x82\xe0\xca\u03ea\x80\x94\x02\xc4a\xf9\x89\x870\u063d\xdd\u007f\x9f\t\xd6\x14(\x86}\xdfSI\xda\x14\xa7\x9f~\x1a\xce<\xf3\xcc?'\xa2!\x00\xbc\xeeuo\x90}\xe5\xfa\x9dT\xe6\x93U\xae\xdbN\xfe\x8d\r\xfd\x81}\xc1\x13\x8f>\x8c\x9d\x8f<\bV\xaa\xac\u0108\x00g\v\xcc\x1e|\x18\xd6o\xfc9\xc0\x19T\u04cf\xa8\x94\vc\x9d\xbf0,B\xe2\xa9\x16\xf2a\xd1\xcez\xa7\x0f\x82\x83\x83\xf5^\xe7#\x83bP\xc0\x0e-
z$\u0202\u06d2\x18T\u0561\x95\xd2B
\xe9\xc0\u008a\x84\x01\x14\t\xb4\xb8\xaf8U\xc7Ws*\x84/\u0107U\x14\x82\xd1\xc0\x86\xa9R\xcf\u04fbh\xa6dm\xe0\x9d]\b\x9c\x96\x92r(\x9b|\x0ep&\x04)X?0\xe3\x8aP\xc9Z\x01Y\a\x8a\xa3\xed\x85\xff?\a\xc1\xd0\tL\x11\x86glh\u0205\xaf\x11\x01\u00d1\xc3\xea\x8a\xc5j\xdfb0\xb0\xe8\x0f\xfc\u07d7W\r\x16\x96\r\x96V\f\x86#\ak|pC\u007f\xe00\xbf\xe0\xc7\xde\x17\x17\v\xec\xda=\u0136\x9d}\xcc/\x8e0\x18\x85\xdb\xe8[X#\xd5LVt\u038a\xbd\x8exR\xb1\xc90VR\xb1\x8f\xd9\xc1DM6\x85\xca[q\u5552\xf9\x04#\xd5ap\xaeJ\xc30j&8\xa5\xf9$TQ/\xacB\xb0v\xbc\xad\xf0E\x89\xe3B\x9c\xfcd\x12d\x1c>'\xe1q\u0692\xc2\x0f\x15:3\x1c\bB\x89q\\\xe2\xb5C\xe4C\x9b\x8b\x95\x05
\x19\xdb/\x86\xab0\u5d27\xff\xe1\r\x1b6\xe0\x94S\xb6\xfe\xc3\x15W\xbc\xf1\x06\x00\xf8\x9b\xbf\xf9k\u0697\xae\xdfIe>Y\xf8\xf1\xbd\xf7\xe1\xa4go\xc6\t[\x9e;\xf5\xd5;\x1f;\xf9\x87\u07f8\x01K\xf3\xbb\u041b\x9e\t@\xee\xa7\xe4\x9c1\x98y\xd6&\xcc\x1c\xbc1|\u0429\x96r\x1f\xf4n\x151\x13)W\xeb`\x04@\x16\xe4w\u05abc\x14\xfb\xc6\x15\xe9\xc0\x8f:\a%\xc0\xb4\x06:\xca\x03\x8e\x8bMX\xc4\x01%*/0W\xfaySP(\x04\xa1DPL0\x93o\n\xc6)?[m>\xce:\xf4\x97-
h\xca!\xcb\x18D\xde\x1f\x06\xca\xfbs\b\xf9\xd8;(\x8b(\x87p\x0e\x15\x9d\x13\x1eC\xc3\b\xb7\xa4\xfeK\xccw(\u04da\n#X\xb5\xbeJ\xcc\xc3\tB\xc2\xc4g\xa1\x01\x03\a3\x12\xac\x8c\b\xc5\u040f\x9cKR\x95\x1b'\xc8\x14a\xaa\xab\xa0\xd97\xf7V\xfa\x06\xab}\x13\xaa2\x82\xb1\xe15\xabq\xedR6n)\xf5+\x97\xd4i\x91\u01a8\xa1\xb4/PU\xbc\xd4H\xf2\xa9\xf2[\x91\x84;\x8c\xadTb\x99~+n\b\xd1@\x8b\t\u03b0\xaf\xa9-
\xfb\r\xbcv\x18\xf3\x958\x91@IP\xef\xc4SS\xb0\xe8\x15\x00:X\x05\x1b'\xb0\xb1!\x90\x98l\xc5S\x8dw_\xd4\xe8\xce\xccy\x96\xc9\x1aXk1\x1a\xf6\u02e6\xa7\b\x9c\u059aO=\xf5y\x0f\\y\xe5\x15\xff7\x11\x15\xef~\xf7\xbb\xb2+\xaf\xbc\xaa\x98\x80\xf9d\xed\x13\xeb\xd1G\x1f\xc6\xf7\xbf\u007f#Nz\xf6f\x00\xc0*0\xfd\xc5O\xfcw|\xff+_\xf4MIq\xa5`W\x9c\x03\xeb\x1c3\x87\x1e\r5\xb5\x0e\xa6X\xf5\x95.\xb3?\xaa\x82@*\x83\x90*\xe9\x8e\xc28X\xe7+(f@2\a\xd1\x02\t\xc1\x13\xba\xa3||\x99V\x81&A\xa0W\xa2\x02\xa5\xba\xea\xa5\xd4LK\xe2\xc2\xe8\xabp0\u01d3|#\xc5GJn\xdc)@\x8a\xe8#\x13\xaa\xf3U\x83\xfe\x12\x83f3?5\xc8\x02h\x80\x11\x142\xe45\xe9\xe5m\x05`)\u00c4\xb9\n\xa2\xaf5\xfc\xaa\xb6\x00\xacu\x81\x86\x16,\xf7-
V\xad\x03\xeb\u0434c\x81c`\xa4\b#\x0eC;V`D`FR#\x92)\xf8\xc7\x14\x00\x86}\vf\x82\xb5\x02\x13\xaai\x8bJy\xc3A@\xed\x15-
R\u6596`\x1cOP\"\xed\xa3\xa2\x8d\xac\xcfR\xcd\x14\xab\xe8\x98e\x1a\xfdf\x12\x95J\xbd\xf1\xd9\

xcc\fM\xfe+\x00\xb7\x8fi\xe3\xa4Z'\x90\xadxwo\xf8\xe6\xca\x17=>\xe4\xe8\x89n\x1d\xe0\x1cC3\x82\xaa(<\u007f\xf8M\xb3\x90\xa81\x0f\x1e-
I\xaf\x85\xc2\xedw\xbb=(\x02\x06K\xf30E\x81b\u0507\rMO\xa5\x94Xk\xe9\xb4\xd3N\u0165\x97\xbe\xf4\x1f\x8f?~\u02dd\x00\xf0\xf6\xb7\xbf\u077c\xef}\xef\xc7\x04\xcc'k\x9fXw\xdf}\x17^\xf6\xb2_\x8a\xbcs\xefw\xff\xe0\x0f\xfe\xe8S\x1f\xbb\x1a;\x9fx\x04\xdd\xee4\x9cu\xa5\xc1\x96\x19\r0u\u0421\x98;j\v\x06#\v\xed\x1c\xf2L\xc3\x19\x83\x1dw\u07c6'\xee\xba\x15B\x8cl\xfa\x00p\xd6\x05\xe7=@\xf5\xa0\xf2.\xa6f\xa6\u041b\x9a\x02e30N\x83\x95F\xa65\xa8\x97Au\xbd\x91\x94\xb0@LQU\xf6\xc2U\x05(\x95\x9a\xa2\xe4Y\xa3a\x1e\x04\xcc\x0eJ\x92\xb0\x8b8`\x14mX\x15
\xdaW\xf2\xb6\xec\x82\xf91\xf6\xc1R\x01\xce\b\xdd\\{<J\a\r\x85R\x19O\xbd\x0f@\xf5\xc0\xb6\xd8\x0f\x95\xe4\x90\x12'
G\x85\xc3J\xdfb\xb5\x1fxk\x06\xac\xf6)\xf1\x8e\bV\xf9\x1eBt\x87\xa4\xc4G7N\xcdzJ\xc4\xd7\xdc\u01a1\xee\xec\x18'o\xe3\x10U[\xa8(Qb\\F\xe3\x1e\xea\xad\xddk*\xc3\x1b*Y`RE\x03\xa9\x16\xb0\u04b4S\xfdu\x92\xf4wc3\x93\x13\x85Kz\x14\x88\xe1\xde\xd5\xee\x01\"\x15n\xc3\xc5t@\xa8\xe8\tI\x95\xd7\x01\x13\xc0Ba\u0407`d\u071f,nbD\x04\xa5\x14\xa6\xa6\xa71\u06bd\r\xf7}\xeb\v\x18\xae,\x95\xde\xf7~\xa2\x95`\xad5G\x1f}Tv\xfa\xe9\xa7\xdf\U00096dfc\xf5\xfd\xff\xee\u07fd\r_\xfa\xd2\xf5\xbca\u00f3\u073ev=O\xc0\xfc\x19\xba\xe6\xe7wa\u077a\x03\xcb\u007f\xff\xe1\x1f\xfe\xc1\xef}\xfes\x9f}\xf1C\xf7\xdf\v8'\xce\x19\xf2>\xe3\x1a.\x94\xb2\a>\xfb\xe71w\xe4\xf1\x18\fG`\x00\x86\x18\xa3\x95%\xdc\xf7\xc3\xef\xe1\x81\xef]\x0f;\x1a@uz`\xa5\xc1:\x87\xd29T\x96C\xe5\x1ddy\a\u0719\x02\xeb.\x14w\xa0\x14\xa3;5\x8d\x83\x8f<\x06S\a\x1d\x84\xa9u\ab\xf6Y\x9b\xc0*\x83s~\xf8\xc3k\x83\xebcIH\xc0:\x0e\x15Q\xe0\xb4]\x90\x1aV\xc7\xe8\xf0\xe3\x04\f\x01\x8c\"\xd7\x1e\xa4n\x10A\xd1\x17,,\x10d\x16\xe8u3Tv\xb2\x94\x1a\xf0y^\xb9\xdcT\xa8\x1a)\x97d\x80'}\x88V0\x18X\xac\x0e,V\xfb\x06\xc3\xc2\x05\xef\x9a
\xcfVH@U\xca\u01a3$I=\xf5\xb1\xdb\x16\x8a\"\xfdy\xa97.\xd3a\x9dd\x8c\x145\xa3\x17Jm\x1a\xc2Tn\xec_\x06\xb9h\t\xe2T}\x0f\x8d\xe9Ij\xd9\x10R)\"S\xa5\vo\xfen\x93\xdbI\xbd\xc8%U\u0144!3\xcd@\x0e?\xbe/\xce\xc7\rJ\xe8$\xc7\u03c2\t\x83\xc4q\xaa\x94\x90\xbew\u1e72Bwj\x06\x18\xae\xe2'\xd7\xfd\x1d\x1e\xfd\xc9\xf7a\xc5\xc1\x98Q\b\xe6Vp\xceI\xaf\xd7\u034e;\xeeXw\xc9%\x17\xbf\x9b\x88\u6bff\xfe\xf3|\xe1\x85\x17\xbbo|\xe3kx\xc1\v\u039d\x80\xf9d=\xbd+\x1d\xc1\x17\x91\xb9\xff\xf2_>\xf0\xc1\xab\xaf\xfe\xf8ko\xbe\xf9\x87%\x8a8k\xabtwkp\xe0\xe6\x13q\xc4\xff\xf6R\xe8N\xd7\u02f5\x04\xb0C\x03\xeb4\xa66\x1e\x8b\r\xc7<\x01[\f\xfdE`\n\xd8b\b;\x1a\u0099\x11\xccR\x1f\xabf\x04gF\xb0\xa3\x01l1\x808\x03\xd69\ue7daC65\x8dC\x8f?\x15']r9\xd6\x1fz\x04\u020c\x82\xe1V<27\x89\x00jbQ\xa2\u0328\xa0E\x82\xb6:\x82*;It\U00061135\x9e\xb6\x98\x17\x82\x13\x8d\x19E\xd0\xe29[AU\u0757\x8d3\xa9\xc2\x0flc0&V\xd1\xe2\x04\x8b\xcb\x06\xbb\x17F\x18\x1a\a\xe7\u01bd\xbcSO\x97\x1a\xb2\xb9:pS\x92\xe99\xaec\x97\xba\x87n\x82\x92\xb4\x86\x85oi\xbf[\xfe\x117\x14.i#/1\x8c\x19\xc6u\x1bB*\xa5\x9a\x91\xf6\xa2\x1a\xad\x12\x03\x92c\x88D\u4f9aN\xc2$)\x9b\xdf\xe24\x1c\xf2d\xcb'\xc2\xfe}\xd1\u481c\x9f\xd8u\x89\xad\xb2\n\xdd\v#\xfe\xcb5\xf9\u007f\xaa\x14,L\nyo\x1an\xb0\x8c\x9f|\xe1\xefq\xdfw\xbf\xe4\xfb.\xc5\b\x90\xe0a.\"D\xa0\x13N8\x01/x\xc1\v\xfe\xf2%/\xb9\xf4\xf3\x00p\xf1\xc5/v\x1f\xff\xf8\xc7\xf69
\x9f\x80\xf93\x14\xc8/\xbd\xf4\xa5\n\x80\x15\x91\r\xef\u007f\xff\xfb\xae\xf9\xdc\u7bbd\xf0\x96[n\xad\x95\x83Q\x1b\xedL\x81\xe9\x83\x0e\xc5\xe6\v_\x8d\xf5\u01dc\x8cbe9\xd0\b\x04\xe7,Hi\x1cr\xd2\xd9\xd8p\xfc\xa9!\xb1\xd7W\xb4\xce\x14a\xaar\x003XE\xb1\xba\x8cbe\x01\xab\xbb\x1e\xc7\xf2\xf6\a1\xd8\xfd\x04\x9c-
\xc0Y\x17*\x9f\x82\xcb7`q\x95\xd1\xe9\x17\xc8U2\xf5\u01c9\xa7GI\xe2F\xf8\x960\xd4S9,R\xe0\x

aaG\xc6b8r\xe8\x0f\x9cW\u007f8\xa0\x13=;J\v\x19\x1f\xf8\u03240\xb2\n\xf3K\x0e#kq\x00\x1cz\x04\xe8tz\xb2\u00b32-
>r\xe5\xde\x0124e\x9d`a\xa9\xc0\xce\xf9\x11\x8cI%\x84\x187xI\xfe#\x86\x05c\xac9\xd9tkL\x01\xbce\x8b\x8b\xc8\xc8\xd5w\xa9\xa1\x1cI\xff,\xe9\x91D\xcf]\x01\xb8\x00\xcd,\xd0D\x96\x12y\xfc\x12\xf4\x03\x80\x97Fb\xb5\xe7<Nw\xa4\x93\xa2Mn=}\xec\x04\x82V\x04\xc5\xca+\x84b\xa5MH\x147\xfe3`\\\x90\x9f&\x8f\x9f\x13Y)\t\x90\xe7]\x14\vO\xe2\x8e\xcf_\x8d\xfb\xbe\xfdE8\xeb\xa3\xe0\u011a\xca\n\x00\xa0#\x8f\xfc9\x9cw\xdey\xdf\u007f\xf7\xbb\xdf\xfbV\x00\u0638q#\xfe\xfa\xaf\xff\n\xbf\xf8\x8b/\xdb'\xaf\xed\t\x98?3+r+\"\a\xff\xee\xef\xfe\xce5\xd7]\xf7\xf9\vn\xba\xe9\aHr}|S\x93\x18\xce\x14\xe8\u032d\xc7Q\xe7\xbe\x02\x1b\x9f\xf7\v~\xa0\x82\x01\xa8\x90\xbe\"\x1c\xf8G\x85<\xef\x95\xf7\x91V\x9b\x02\xf2#\xf6\xae2]rE\x81\xfe\xee'P\xac.\xa2;\xb7\x01zj\x16*\xebB\x84\xb0{\xd7*fg4\xa6z:p\xe0M\u08925\xa8\x02\xe9C
\xb1\x01\x86\x85\xc5`\xe80\x18Y\x18\xe3\x81=R\x046\xfapH\xd4>\x13D)\xb0V\x10\xad!PX\x19\x12F\xf3\x0e\xbd\x91A\xb7\xe3\xd0\xc9\x14\xb4\x0e\xca\x18\xaa&2]h\x86:\v\x14\xd6z9\x9c\x13\fG\x0eK+\x06\xc6:\x9f'\xd9B%4\xc1\xb9\x1ct\xaaH\xe3\xf2\xb9q\x02\x88\x92\x00\x9c$X\x8b\x14\xc4S~\xba\xd9xL\x8c\xcb\xca\xe9Kj\xfcn\xba\xebPL\xbb\xa7\x8a\x87\xa7j\xac\x1e\xa1\x01Zj\xc8\xc3mH\xc2\x1e\x11\xd5\x19\xa3\x9a+e\x83M\x8a\x8b\u00ddH\xf0\xed\xc9\x19\xd0a\x1a\u05c9\xf2Q\x85\xc4\xc1\x06\x82\xc0`\xaf%\x0f\x12X\xa1\xf43\x18g\x13|\xb9\x9eu{\x18-\xed\xc6\x1d\xd7\x06\x17\x17r=\x87\xe5\xfb+\x00fgg\xf1\xdc\xe7\xfe\xfc\xb6\xb7\xbd\xed\xadW\xfe\xe9\x9f\xfe\x19\x00\xe0?\xfc\x87\xdf\xdeg\x81|\x02\xe6\u03e0u\xcd5\u007f\x9fR+\a\xbf\xef}\u007f\x10\x80\xfc\xa6\xda%\xc5\xca'\x01\x883\u041d9l\xbe\xf0U8\xfa\xfc_\x02)\r7\x1a\x822\x9f2\xcc\xc6V\xa92\xd6\xc1\x9aA96^\xbb>%\x89\x14\x82\x97\b\xeaN\x17\xb3\x87\x1c\xdd\x18\xfcqp\u01a0\xb0\x84\xa5\x15\u03c1\xf6z\xde\xcat\fD\x02\xc00\x01\xc6x\xe0\x1e\x15\x1eH\x87A\xc3\x1dKiV\b\x8a\b\xcf\xdfZ\xf2&_D\xec7%\xad\x81\\A\xb1\x0e\t\b\x04\xe3\bK\xcb\x05VV\t\x99fd\xc1\x0f[\xc5\xd0\xdf
\xf4\xb1\xce\u06fd\x8e\x02\x95\x02\x04\x90\x0f\xd5z\x15\xa9\xd6R\x9d\x02\x15\xe8\xb4\xf0\r\xcdP\x89FQ\x9c\x84f\xa3jDRjnE\xf5)\x9a`f\x85\x10\x9b\u059cl\xad)rP\x81x\xadzO\xe4\x83\u008d\xff\x8f\x15\xb74@\xbf\xad!\xba\xb71\x9b\x90p\xa5D\xa0)\x80\x94\x00V\xa8|n\x12U;>\x85\xd9\x039G\u03df\xf0\xb9\f=\x14'\x9e\x03\xcf;9\x06\xbb\x9e\xc0\x1d\xd7}\x1c\x0f~\xefKeM_\fW}\x97\x84|\x91\xd2\xe9tp\xdey\xe7\xe25\xafy\xcd;\x8e<\xf2\xa8[\x01\xe0w~\xe7\xbd\xf4\u05b7\xfe\xef\xb2/_\xe3\x130\u007f\x06\xac[n\xb9\x19\xd7^{m\t'\u007f\xfc\xc7\u007f\xf4;_\xfc\xe2\xf5\x17\xdc|\xf3\xcd5
\xf71q\n\xceZ\xa8n\x0f\xc7\\\xfcj\x1c\u007f\xe9\x1b@*\x87\x19\xae\x86\x80g\xf1A\x12\x91\xdaH\x06y\xe0\xc4\xfb\x83K\x92\xba#i\x9b,\xd2\x16\xa6\xfcW\x1c\xc0\x11x\x89\x1a\x93\xaf\xa0\x17\xfb\x0eC\v\xf4:\x82,\xf3#\xf9)7\xed\xac\xa00\x0e\xfd\x81E\u007f\xe8\x82D/\x02\x0e'r\xba`\x9b\u01de\x04\x96\x8c\xe1Dy)\x89\xd6\x10\xc5A\x1aWy~H\xa2\xcb\x1e\x8e\x1c\x86C_\x067)\x13)'$i\\\x14\xd2L\x91\x93:s\xd1`\x90\xbd,\x12\xa8\x95\xb2\x8d\x1c\x8e\x9a\u0084\x12:%275uH\xa3\ua9b1\f\xb9\x06E\x938coU\xd1\xd4\x00\x00
\x00IDAT\xa5\x96\xb9\x14'\xad8\xa1`\xa8QqK\xbd\xc0\xde\xdb$\rS\xbb;{\xb5\xefx\u007ft\x8d`\xc9+1\t\xaa2H#\n\ue4e0\xd0\xecDp\\\f'P.Mi\xa0\xb3\fYg\n\x8b\x0f\u0789\xdb?\xf71<z\u02f7\xfc\xe7\x01@1\xec\x03b\xbd+(D:\x9d\x9c\xce?\xff<\xbc\xf6\xb5\x97\xff\xd1\xcb^\xf6\x8a\xbf\a\x80\xf7\xbe\xf7=t\xd6Yg\u027e~\x9dO\xc0\xfc\x19\xb0\xbe\xf8\xc5\xeb\xf1\x9e\xf7\xfcG\x01\x80\u007f\xfa\xa7\xff\xef\r\u007f\xf6g\x1fx\u04cd7\xde\x18='|a\xc7\xec\xaboc\x90\xf5z\xd8\xf2\xf27\xe2\xe4W\xbf\r\xc49\x86K\v\x95\xec\xaf\x04\x8az\xa3\x91\x9c\xc0\x99p\xe1\tU\xae}\"\xa5\xc3\xe0\xd8\xc5\x1e\xffO\u01fcFU\xeb&\x8e

\xb0\x05\xa0\xac\x83\x8ay\x8ep\xe5\x84gQ\b\xac\xf5\u0b78~\xeb\x14\a\x85XW\xe1\xd4J\x83Y\xf9\n,|\xf9\xdeZ](\x1e\x95\x13\x10\x80XB\xaf\x8fj4R\xd9L+u\x92\xb4VvD\xfd9\xa7\xfcu#{\xb2\xfa\x15N\ng\xa9R\x908\xe5\xbe\xeb\xbcu9\xdcEU\x95^\a\xefz\a\xb5\n\xaf\xa8\xd34\x14o\x93\xa9\x1d\xc0\xd1\u061c\x1a\x159\xd7\u0741\xeb'\x92=\xa8
\x83\x190\x14\xfc`\x94\"\x05)\xd8O\xe1\u0081\xc5A\xc4\u008a\xb7B`\x00\x16\x15u\x16K~?\x04D\xa5\xcc3\u02fb\u0419\xc6\xf6;~\x80\xdb?\xfb\xb7\xd8~\xd7\x0f\xc1\u02a7\xa1\x14\xc3\x158k\xbc7\xbbx
\xbf\xf8\xe2\x17\xe2\rox\xc3\x1f\xbd\xe4%\x97\xbe\v\x00\xae\xbb\xees\xfc\x92\x97\\\xea~\x16\xae\xf3\t\x98\xef\xe7\xeb\x9e{\xee\u0131\xc7>'\x02\xd1\xcf]u\xd5o\xfc\xee]w\xdd\u0163\xd1(9\xd52\x98\x19\xd6\x14\u0226f\xb0\xf5\x97\u07c2\xe7\xbd\xee\xdf\xc3\xe9\x0eVvn\ai\r\xb1\xa6\xd2A\xd7xP\n\u0578\xb7\"\x8d\xbe\xe1\xa5\xc4\u03c6\xe0]\x17\xc2'B\xb5\x1e\xbd5@R\x1e\xab)N\xc7P\xacJ=\xd8G\x85\x82\xef|Q\xa5E\xce\x00\x9d\xc7n-
%\x13\x8a*T\xe7\x1a\xa45\xa0\x14\x88t\u0253\x97\x0f?\x04B\xb7Z\x1f\xc6\xd9\xf0HE\u0101\x95\xb4c\x17\xef3\r\x9b\x93\xf1&\x1e\xa8\x85zH&dk\u07d2jH\xa7tMlJ\xfa\x12\x15\n%
\\\x01r\xcd\xf0\xb2\xe2\xcb+\xcd_Y\xd1S\xfaC\t\x8dRo\xa26\x00\xbaI\xfb\xb4\x1f6\xea\x89MkT\xe1\u9a40\xe1\a\xc6r\xf6T\xcb\xc8*82\u07bf'd\x8d\x86\xb8W\x18\x01\n\xe7S\x83@\xc1\xfe\x98*k`!B\xd6\xed\x82l\x81\a\xbeu\x1d\xee\xba\xe1\x1f\xb1\xf0\xf0O\xa1t\x0e\x10\xf9\xc13\x02AP\x14\x85LMO\xd1\xc5\x17\xbf\x10o|\xe3\x15\xef\xbf\u4497\xfc\a\x00\xb8\xf5\u059b\xf5s\x9f\xfb<\xf3\xb3r\xadO\xc0|?_\x11\xc8\x01\xe0\x83\x1f\xfc\xafo\xbe\xf1\xc6\x1b\x8f\u0736m[\xedx\u034aa\n\x83\xee\xf4\x1cN\xbf\xfcmx\xdek\xdf\t\u05dd\xc2\xca\xe2\"T\xa7\x9b\\t\xae2\x92J\bQ\x0f\x8c\x14\x9aR\f\xd2\x1a\u0119\xbf<\x03\x90;k
\xc6@|\x84{Y\x95\xc7x\xb8\xb1+>\x98g\x11\xd7K\xdd\nP\x13\xb8\xf3\x1dN_\x89+\rRY\xa0U\x94\x0f\x9d\xe6\xa4\u04a6\u0291\xd6S%\xd4Z5\x8a\xf8\tG)\xfd_\xfc\xe9\xc3k\xc5)\x89\xabk\x80T\x1aRD\ud217\x0e\u05d0\xb4P!@5\x0f\x934\x16+\x86\x84\xc6h\x97&\xa5CM\xfe=\x8d\x91\xe3\xea1D.\x1c\xc9@O\xeb\xd0\xd1^\x165\x03\xa3k\xdc~rPI\x8b\x80\x12\xed=\x90k\x06r\xae\xf8z\xa5\x18\xb0\n \xeb\xa78\x01(\x12\x14\x16\x18\x05-
y\x15:\xc5a(\u0281\x88\x91O\xf50X\u0609\xfb\xbe\xfci\xdc\xf7\x8d\xcfa\xb0\xb8\v*\xcb\x01\x10\x8c\x19z;\nq(\x8aBz\xdd.\xbd\xf0\xa2\x8bp\xe5\x95W\xfe\xe1%\x97\xbc\xf8\u075e\x9a\xfc\xc1\xcf\x14\x90O\xc0\xfc\x19\xb4D\xe4\xc8_\xfe\xe5W\xbd\xe2\xe1\x87\x1fFEM\xfa)8\xb1\x16Y\xa7\x8b3.\u007f\x1b\u03ba\xf2=pY\a\v\x8bK`\xad\xa1:]\x00\x80eU\x19O\xa5\xf3\x87AM\xe0\x91@\x81\x02\x9dA\xac|\xe5\x1a~\x9e\xadop\x8a5\xc1\xf9\xceU!\xbd\xa0\xe4v\x83\xe1S\n\xe0\xc9\xc6C\xb5\xe0\xe5\x00\x82\xac\x00f(\xed\x81\\\x02\x1fZ\xa3\x02D\xea`\x1a\x13s\x9a\x00\x19\x833b\xa3/\xda\xc1J\x04?)O\x02\xe0\u069eR\x8b\xe1\xack\xb2\u06c1\x1c\xa8\xf3\xdbq\u04e1\x96\xffo%\xa2\x1b\x8d\u0438\v\xd5F\xec\x13\n\x05i\xa5\x9fV\xe1\xa8O\xdf\xd3\xf8Sjm\\\x8a\x8c?4\xda\x03aN\xb5\x83Ju\xa6\xd1\xe4e\xa3\xa9\xab\xa2\x04?
&\x86\x8d\xea*r(,P\u0113Yz\xe2\x01\x00\xe7@\xa4\xa0\xf2\x0e\x96\x1e\xbb\x1fw|\xeecx\xf8\xc6/\xc1\x16#\xb0\u03bd\xe3\xa45\xb0\xa3!\b\x0e\xc6Z\xbbn\xdd:u\xfe\xf9\xe7\xe2\xcdo~\xf3\u007f\xbc\xe0\x82\x8b~\x1f\x00>\xfc\xe1\xff\x87\xb6n=\xd5\xfc\xac]\xe3\x130\u07cf\xd7O~r\x1b\xb6l9\x19\x00\xf0\xf1\x8f\u007f\uc94f>\xfa\xe81\xbbv\xed\xae\xae+\xf2\xb2\fq\x82\x93.|9\xcex\xfd;\xa0z\x1d\xac\xce/\x87\xa9z\x02\xe9\xcc\xdb\xd0f\xc6s\xec\xc1\xb1NP)\t(P#>E\xa0\xb2\xc0\xa52b\xcc\x03.g\f\xe8,T\xe5.\x96\xc6U\xa3TB|@\xb4aM\xa5\xe5TQ0\x15~q\x05\x86\xf0\xee}~s\t\u0380$e\x80p\x1a6\r4|\xc8\xd3L\xd14\xe6>z\u03d0\x94\xdar\x92\xca\u0696\xc6\xd0+\x81?\xaeZ\xbe\xf5*=\x89\

x9a\xab=\xb7\xc8l$\xf6\xba2N\xba\x97\xda\xf6Fu^r\xe1L\x15\xa5B\x95\u007fw)gL\xbcV\x9ae\xb3
4\xec\n\xdah\x95t\x83l&\xf7\xd5\x1fI\xf3w\tQh\x1a7\x81\xb4\x1a\x8f\xcb\x06\x0f{\x0e`\x0e\u0170\x
81n1\x8e`\x85\u0293\x0f#\x89\x00\x14\a\xad3\xa8N\x0f;\xee\xfc\x01\xee\xf8\xa7\x0f\xe3\xb1\x1f}\
xcfG\xc7\xe9\f\x80O`\xb2f\b\x88\x85)\n\u0670\xe1
u\xdai\xa7\xed~\xfd\xeb_\xff\x9f.\xb8\xe0\xa2\xf7\x03\xc0\xaf\xfd\xdaki\xe3\xc6\xc3\xe4g\xf1z\x9f\
x80\xf9~\xbc\xb6l9\x19\"BD$\xb7\xdd\xf6\xe33\a\x83!Dd\b\xa0\x13/0g-
\x0e:\xfc(<\xffW\u078c\xe9\x03\xd7aq\xf7\nD$\xf0\x92\xc1\xc7C)\x90RP\xd1\x1e\x15i2=UG\xf8hO
\x1a*\xeb\x18\xd2SzgST\x97\x04g\xc2\x04\x14\xa9\xbc0\xd3[n\f\xcd4\xea\xbc4\u0772TT8)\x15\x1
8\xe2Z@\xb1\x95\x17\xa8\xa8\x8d\x84d\xaem$^)!-
\xdcx\x12\xa9W\xbb\x91\xe4n\xa4Q\xa9\u05ea\xda
\x17\x045~!\xcd\x1aM<N\xa4\xbe\x89\x107\a\x81\xe2\xc0\x0e\x05\xd0G\u0660\xac\x85k\xc88\xe
aR\x83\xc3/\xc1\x9b\u05ae\xc4\xeb\xa5:\xc6\x1d\b\xcawQjD\xba\xa2\x10\xbc\x14\x9a\xbc.\f\xfd\x
94\x8aN\xf2\xa0.*\x00\xbaQpdC\x11\xe1\xfb-
\xe9+\xaf;=h\xad\xf1\xd8\x0f\xbf\x81\x1f}\xe2/\xb1\xeb\u079f\x80\x94\xf2\xfd\x9e\xf0\xe1p\xae\x80
\xb3\x05\xac)p\xfc\xf1\xc7\u0469\xa7\x9e\xfa\xf0\xa5\x97^\xfa\xab\x97^z\u0677\x00\u0aab\xae\xa
4\x97\xbe\xf4\xa5\xf2\xe2\x17\xbf\x14\x130\x9f\xac}j\xdd{\xef=DD\"\"\xeb^\xf3\x9a_=\xe2\xb1\xc7\
x1e\x03\x11\xa9\u0503\x9c\x00\x1c\u007f\u03858\xe4\u06130\x1ax\xcfVW\x83LI\xf0\x8eB\xb03j\
xa0[N\xd7ES\xa60\xd2.\xae\xaa\x16K>9!\x97\xe3,G\u0273\n%tJC\u007f!\x82:\x1e\xc8\x18\xef\\\u
06948\xa9\xe7\x12Q\v\u04a4\\uc*\xbe\xf93\xe5\x8f&\xf4\xcd8\xcd\xef\r\x00\xd2\xe0\x8a\xa6\x99\x
155\xeaWj\xdb\\J\x95K]iRe`V\u07c8\x13\xb2\xad1O{\x00\xdbT&)mO{|@u\xaf\x8bQ\x97\xccsz\xa7
\xe5f\xe2\xc1Z%_\x12(\x13\xc5\xde\xf91>O'\x02#\x04\x03\x05C\x0e\xa2t\xb0V\xf0fl\xf1\xa4E\x8a\
x91\xe7=\xd8a\x1f\xf7}\xf5\xb3\xb8\xe3\u068fa\xf7\xc3?\x85\xd2\x1d\u007f\xaa\n[\x89\xb5\x8583
\u034c\x13O>\t\xe7\x9cs\xceg\xde\xf3\x9ew\xfd\x9f\x87\x1ez\xf8\xfd\x00\xf0\xaew\xfd6\x9d}\xf6Yr
\u9957\xfd\xcc^\xef\x130\u07cf\xd7\xfd\xf7\xdfO\x00\xa4(\x86'\x0e\x87\u00e3\x17\x17\x17\xc1\xcc
dm\xf4iv\xc8r\x8d\u00f7\x9c\x8a|z\x16\v\xfd\x81?\xc6\x06\xdbP'M\x1e\xb5\x91\x05\xb9V\xc1\xab\
x02\xa8\xc7\b\x1d\x97\xfc\xbcC9\x86/\xa5\xa7t%\xed\xa3\xa6\xa1\x9f\xa4\xbc\xb0\x946\"13\xa2\x
04\xb4\xf8\bc\xd2\x10\u0575t\x94\xec\x0e\u049c%O\xe2\u07e9^V\xd6\xce
H(\x8b\xb5\baj\x96\xe4q\x8a\xb2\xac\xdd\u00c8y\xed%L\x9e{\x94(rL\u04a31\xfb\x15\xa0\x95%\xa
9\xfb\xbd4\x8a\u548f\x96\xd6B\x1a\x82q\x9a\xa5\xc9\xe2\xd4\xda\xce\r\x11P\xaa\x9e\xe1&M\x13
$\x8b\xb1\x1a\xd7T\x9d\xb9\x84\xaasX\xf4\xcd1a,_H`\x89!*\x03\x87f5\xfb@(\x88#\xa8L#\u02fbXz
\xe2A\xdc\xfd\x85kp\xff7\xfe\t\x83\xa5y\xa8\xac[\xebS8;\x92b\u0627^7\xc7\x19g\x9c\x81\xcb.\xbb\
xec#o}\xeb\u06ee\xf8\xe0\a\xff\x02\x17]t\xa1\xfa\xe7\u007f\xbe\xc1\x12\x91\xfc\xac_\xef\x130\u07
cf\u05ed\xb7\xde\n\x00\xf8\xeew\xbf3\u06b6\xed\t\x93$\x8c\xfb\x0f\xb9\x00y\xa7\x87\xf5\a\x1f\x8
a,Wp\xcb\x16\xb6\xf4\x84\x1e\xbf\xc0\xc7*\xb8\xe4\xf8.i\u015ap\xc4\x12\xc0\x9dR\x1e=\x863Km
N\xbd<-
\xd4F\xbe\xa9Fs\x8f\xc9\xff\"\xf7\x1a\xdd\fE\xa4\x1a3O\xe5\x13\x89\x96\xbc\xae\x90\xa9N\r\t\x05
_\xca\xdc\u028a|\u0301\xb0\xd1\xe0ly\x81\xa8\x85\x18\x1a\x93\"6\x90\xb9m\xf0\xa6\xed\xd0@\x
18\xf7\xe8*\xc1vM\xe2\x1a5#.iT\xeac-
\x80\x96;\u07db\u0405\x92\xcfN\t2\x04d\u0283\xb9&oM\x1b\u007fVQ\xf5Yt$p\xe2-
~c\x18\x92\x84\xe9Md\xfe\xf5c&(\ue049Q\xf4W\xf0\xc8M_\u015d\xd7^\x8dm\xb7\xdf\bg\r\x98\xb3
*l\x1a\x04g\x8d\f\a\xabt\xf0\x86\x83p\xca)[\x17.\xbb\xec\x17\xdf\xfe\xa67\xbd\xe5o\x01\xe0O\xfe\u
43f2w\xbe\xf3\xb7\v\"\xda/\xae\xf7\t\x98\xef\xc7\ub847\x1e\x02\x00\xdc~\xfb\x1d\xb4\xb8\xb8T\xe

5D\xa6\xd8\xe9\x04\x9c\xe5\xc1\a\xc4WK.a4\xdcZ\xc7p\xaa\x83\xebZ\xd7\xc3x\x85G5>\xb5f\xce\
x1a\x80\xbef\xe9\x9aR\xda\x01\xa0]\xf2\u007fI<\xa9\xaf\xf6\x1458\xe0D_\xbd'\xc1s\x8b\x11UIaP;
\xbe\xd5\x14\x1fi\x1a\x8fT\xd5p\n\xacT/\xe21\xa6\xf2\xdc\xd3\xc6\u0664J\xd2j\xba\xe1\u007f\x02
4\x95#\xed\xef\u025e\xda\a\xa9\x8c\xb3mci\xda\xd8R\vM\xa3\u0607HDJ\xc5\x03\xbd
#\xf2\xa1\x12a\xe0G
U\xe0r\x8d\u0593\xe4\xc4%Py\a:?\x00\xa6\xbf\x84\x9dw\u078c{\xbe\xf4I<\xf0\x8d\xcfbe\xc7c`\x9
5y\u014a\xf3\u039d\x9e\xa6rR\x8c\x86t\xc4\xe1\x9bp\xce9g\xcf_y\xe5\x1b_~\xdey\x17|\r\x00~\xef
\xf7~\x976o\xde\\\xecO\xd7\xfb\x04\xcc\xf7\xe3\xd5\xef\x0f\x00\x00\xcb\xcb+0\u01b46\x02\xfb\xa
b\xcb\u8bec\x04\x06\x84a}\xf9\\S-
H;\x05\\\a\xa5\xe4\xfbM\u0395\u05a8\xea\xeb\xf5jThP\"[\xab\x98XnX\x8dx'\x01\x81\xb3\u0271\xdf
5C\u0712Z85\x9cj\xf3\v\xa7\x96J;\xf9y\xda\x03\r\x9d\xe2\x0e\xed\x85\xf2Hi\x8a6\x00LM\xb6\xa8\
xc1\xd9K\xe3\xf5\x96\xc6\f}\xbbU@\x02\xcc2\x0e\xf6m?\x9b\x9a;\x8eY\xd36\xfe\xa4\x96\u075b
\xa1\n\xf7i@\xf1\xb8@\x94\xf8\xd3\v`
q\x0e\xcd[\n\xa3jZ\xa7Q\x81\x00!\x9f\x9e\x83\xd8\x02\xbb\xef\xbd\r\xf7~\xed3\xb8\xe7\x8b\xd7`\x
e1\xe1{\x82\x0f\x0e{7c[\xf8\x01\xb8\x80\xff\xa6\x18\xd2a\x87m\xc4%\x97\\\xfc\xf0o\xfe\xe6Uo8\xe
5\x94S\xbf\x06\x00g\x9f}6\x1dq\xc4\x11\xf2\xcaW\xfe2&`>Y?\x13kzz\x1a\x00\xb0n\xdd\x01\xd0Z
S\x1b\x98[\x01\x1e\xbb\xfb6<\xfb\xdc_\x84\xca2\xc0\x8cJ\xa0qu\xf4+\x8bT'-
<\xed\x1eh\x81\xf1\n\x93BVg2Q\x1a\xf9\xe1d#`\xd4\xef75dt\xf0\xde\x1c\xa4\xe3\b~x\xbc\x8d*\xb
8\x1cj\xa2\x06O,\xb4f\xb5\xdd\x046\xa0\xddKdOTC:H\xda\xe4\xe2\xa9\xed\xa4\u04e0^\xa2\x94\x
9dZ\xee\xdf5*zY\u3157f%\x1e\x01\xbf\xa5\xe1K\r\u03bby;{z\xee\x94\xd0m\x8a\xbc\u02e1\xa2\x9
4/\xf7wl\xc5W\xe2&\x98_9T\x8a\x14'\xa8\xbd\xd7\x11\u01f9\xd3\x05\xeb\f\xbb\xef\xbf\x13\x0f~\xe
3\x9f\xf0\xd3/}\x02;\xef\xb9\r\xce\x14\t3\xe7\x1b\xf7\xc4\u02a3\xb83B\x04\u06b2\xe5D\x9c~\xfa\x
e9\x9f\xfd\u0407>\xf8V\xa2\xfc!\x00x\xc5+^N\xaf}\xed\xe5\xf2\xf2\x97\xff\xd2~w\xbdO\xc0|?^g\x9d
u&>\xf0\x01\xe0E/\xba\xa4\xf3\x0f\xff\xf0?r\x9f\xe03~Y\xfe\xf0\xf3\xff\x88\xe7\\\xfcj\xac?~\v\x86\xa
3\x11\x8ci\ua269\x9e\u053e\x06pK\xa3Z\x17\xa9Q\xcc\xe9\xa19\x0e\x1d6\xc2\x1a*q\x9f\xa2\xa4\
"\x0e\xa0j\xa5Q\x95\xa2r\x1etX\xeb\u030f\xb1\xc6\xdfZ\x15\xf6\xbfF\f\xb2'*\xa9\xf5\xb4\xd22h\x9
3\x16\xff\xae\x85\xedi\xbb\u03f1\xf9$Z\x9b\xc7\x1es\x14h>\xb6\xa4\x1a\x8eT\x9a\uc067\xdf\x13\
x8d\x1e\xe9\x16\u037e\x12\xd7\fd\u025b\xee\xc4\x0f\xf9\xd8p\x92\xaa;$W\x03K.\x8c\xe9#PcJi\xa
8\xbc\x8b\xfe\xeem\xb8\xef\xeb\x9f\xc5=_\xbc\x06\xdbn\xfb\x1e\u0330\xdf\xf2\xfa\x87\xe9_\x888k
\xe9\xc0\x03\xd7\xd3\xc9'\x9f\x8c\xf3\xcf?\xef\xaf\xdf\xfd\xee\xf7\xbe\x93\x88\x16\x01\xe0\x8f\xff\
xf8\xfdt\xee\xb9\xe7\xca\x19g\x9c\xb5_^\xef4\x81\xbc\xfdw\xddz\xeb\x0f\xe9\xb9\xcf=EDD\xbf\xf0\
x85\x17}\xfe\xc6\x1b\xbf\u007f\xd1\xc2\xc2B\x01
k\xfe\xec9\xaf\xfb?\xf0\xa2w}\x00\xab\x05\xb0k~\x11&\t@p\xad\xf9\xedT\x06\u8d8dl73}\u04df+\x8
1<\x06\u0460R\u03a4\xe0\xe5\x12\x00\x8c`\xdeVIK\x8b\xeap\f@\xf1\xaf\xb4_\xc5\xda
\xb9\xa7\n\xbc\xf5\xff\xa5\xae\xf2\xc0\x1a\x8f3\xd5~G\xa0N)\x13Yc\xd3L)\x1d\xfaW<^iy({\xba\x8d
h\xa6\x16\u04d5\xe2\xfb\xc0I\u04d9#\x80\aKZ\x1d\xee\xa0pQ^X\xf5`\u01ac\u0723\xc6'\xdc9\xb3\
x02k\r\xcer\x10\x80\xc7~\xf0u\xfc\xe4\xd3\u007f\x8d\a\xbf\xfd\x05\f\x97\xe6[@<L\xff\x86\xb8\xb8<
\xd7x\u05b3\x0e\xc1\xf3\x9ew\xca]\xafz\xd5+\xff\xecW\u007f\xf5\xf2\xbf\x01\x80\x0f\u007f\xf8o\xf
8\x8a+\xaet\xfb\xfb\xf5\xce\x13\xc8\xdb\u007fW\x00r\"\"\xb3i\u04e6o\xcf\xce\xce\x02Q\xf1\xd6X\xf
fr\xcd_\xe0\xe6O\u007f\x14\x9c\x01y\xb7\x97\\\xe4\xf5\x9c\xb3(\t\x94\xa0Q\xe7\x16\x8a\xa2\xa6h
i)\x96#?\xad\xaa\x10\xf6\x9a\"\x06
\x98p$w\x12+G\x1a?U\x10\xad\xcd\xddJ{UL{\xa1\x81\xfeU\u07e3J\xf3\x9d\x98

\x8e}I\xe3g\xc6B{\x9a\xaf\x1daL\u07ce\x96\xd7xlR\u007f/\x95\xf4\xda\xd56\x95\x1b\xa6j\xde>\xb5\xbcv\x84\x92\v\xcf\u060f\xe0w\x15\xa1\u02c4\x9c\xa3\xb4P0r\xc0\xc8\tFNj@.\xd2\xdc\xc0\xc2ILk\xe4\xbd\x19\xe8\xde\x14\x88\b\x8b\x0f\u078d\x9b>\xfc>|\xed}W\xe1\xee/^\xe3\x81|,\xfeNy\x97O\u71cb\x0e<p\x1dN;\xed4\xbc\xfa\u056f\xba\xfe\x03\x1f\xf8\xc0\v#\x90_}\xf5G\xf9\x8a+\xaet\u007f\xfe\xe7\u007f\xba\xdf_\xef\x13\x9ae?^_\xff\xfaW\x11\xf5\xb3o|\xe3\x15\u07f9\xe5\x96[\x17\x1fy\u4479\xb6\x13\xbd)\n\\\xff\xa7\xef\x84\xd1]\x1cs\xf1\xaf@\x15\x05`m\xb8\b+\rw\x8a\x95\xc1\x03z\xbc\xb1\xd6\xc2\x17\x8c\xe5\x05Q\x18\xb2I6\b\x97T\xf6q\xb3\x18\xcf\xf8LO\x002V\x1d\xcbZ\x1c\xc3^\xa8\x92\xbd\xfdI:\xc9\u0674\u007fM+\u0174\xc1\xd7\x06\xde\xd8\x03\x95\xd2<\xcdP\v\xfdB{\xa4=R\u05d3::Sku^\xd1]%\x9d\x95\x9a`\x05N;>\x1e\x9dLmj\xf6\xd6\x03L\xf5\n\xdc\x04N\xdco\x9cT\xdbW\x89\xc2\xe8=|'Ck\r\u055d\x02
\x18,\xec\u0093w\u0742\a\xbe\xfd\x05<v\xf3\u05f1\ubf9f\xc0\f\xfa\xcd]\xbe\xf2\xdcwN\xc4\x14t\xd0A\a\xe1\xb0\xc36\xe2\xc4\x13O\xfc\xf1\x05\x17\x9c\xff\x97W^y\xd5\xdfGZ\xe5\xdak?\xcb/}\xe9/\xba\xcf|\xe6\xd3x\xd9\xcb^\xb1\xdf_\xef\x13\x9a\xe5\x19\xb4~\xe37\xde\xf8\xd9\xcf|\u6cd7n\u07fe}\xbc\xd4\f\x17\xcb\xdc\u01a3p\xc9\xef}\x14\x1bO\xf97\x18\xf4W\x82\xd6\xd7\a\x00\xa4:\xb8V}s\vHq\nz\x01\x8f\xe3\xf0\b\xc1{q\xb8\xd0\xf8L\xaf[\xb7\x87\x90\x9eVj\xa7\r\x19[\xa8\x96\xff\x15:\x05-
\x95jM~\xb8\x87\x8d\x80\xf6r\u007f\xf2\xff\xe3B\xac\x816%\x96\a\xb2\x97'\xd6\u0417\xb7\x9d\n\u048d$\xddDJks\xf2R\xc3\x0e\xd7\xe7W\xad\bF\xb6\xeag\xc4|\xcex/5\x9aM\x00\u07f7ad\xd3\xd3`\x01V\x9f|\x02\x8f\xfc\xe0k\xf8\xe9\x97>\x89\xc7\u007f\xf4]\xac>\xf9\xc4\x1a\xaf?\x05\x1fr\xb2\xb6\x18\xb9u\xeb\xd7g\xc7<\xfb\xd9\u063c\xf9\u87dey\xe6\xf3\xff\xdb\xdb\xdf\xfe[\u007fCD%\xa1\xfew\u007fw5]~\xf9\xeb\xe4\x99t}O\xc0|?_\x9f\xfc\xe4?\"J\xb0\xbe\xfa\xd5/\x9fu\xd5Uo\xfa\xd4\xddw\u07fdqO\xbf\xb3\xf5\x97\u07ccs\xff\xaf\xff\x86a\xbf\x0f\xe3\xbc\xe6\xc0%~\xdee\xf5\x89\xb4\x8a[;\x977\xf5\aA\x00s\xae\x02\xdek\r\xb8\xe8\u0451zm\xd5\xf4\xdb4\x0e\xf0.i\u6943N\xa9\x94\xb2)\xb3k\x1bY\x97\x96*|\xefT\xc5\u068a\x17\x91q*\xa6mx\x96\x1a\xb4Jk\xc61\xa1\xe6E\xd2\xd4\xfbKc6\xaa\x91i\f\x91\x8a\u058a?\xdf\u059c\xe6@\xa3\xa8D\xaa\x19u\xe21\xc4\u00c6\xc1\x1e\x13\xbc\xe6\u0460Q$\xd26Toz\x131\xf2\xde\f\xc4\x19,<r\x1f\x1e\xbf\xf5;\xb8\xff\x1b\xd7\xe2\xe1\x1b\xbf\x8c\xe1\xf2\xfc\x9a
N\xac\xbc\u07fe\xb5\xae\xd3\xe9\xf0q\xc7>\x1b'\x9cp\xc2\xe3\x17\\p\xde'\xae\xbc\xf27\xff\x84\x88\x1eM\u007f\xe7\xc6\x1b\xbf\x87\xe7?\xff\xccg\u0735>\x01\xf3g\xc0\xfa\xad\xdfz\ab(\xed\u55ff\xe6=\xd7_\xff\u03ff\xbfc\u01ce1\xfca\xe5#\xe3\x9e\xf5\x9c\xad\xb8\xf4?}\n\a\x1c\xbd\x19+\xbb\x97|\xb2O\xe9\xa1RM\xbb\xa4\x81p\xcdj=\x8eo\xa3\x99l\x8fj\x80\xa4\xe4\xe0\x13\x80\xb1\rpX\x8b\n\xdf\xe3\x9c\u0358\u007f\xeb\xdaMPZk\x8eH\xc6\xf9c\xd9\x03\xc5Q\xbb\xfd\xda\xff'\u05b9\te\xd5\u49db\xf7W5\x1b\x83\xdb\xe0\x1a\x9b\x95\xb4\x9cL\u0186_\xd1p/\xa0\xea=\xe2F\u056e\x89\x02\x95\xe2}T\xac\xab\xde,\x17\x14)#G\xb0.\xd17%Fk\xe9f\xcb\xf1\x15P\ny\xb7\v3\x1aa\xe7Oo\xc3#?\xf8:\x1e\xf8\xe6ux\xe2\xf6\x9bP\xac.\xb5\x03Sp\xe3,CK
8\xe4Y\a\xe3\xb8\xe3\x8e\xdbu\xce9g\xff\x8f\xdf\xff\xfd\xf7\xfd\x15\x11\u0756\xfe\u03bd\xf7\u0783g?\xfb\xd8g\xecu>\x01\xf3g\xc0\xfa\xc8G\xfe_\xfc\xfa\xaf\xbf1\x80\x89\x1cp\uee7f\xf0\xc9o~\xf3\x9b\x17:\xd7\xeeq\xca\xcc8\xe1\u016f\xc1\x19W\xbe\x17sG\x1e\x8b\xe1\xb0@Q\x14\x10g\xe1b-^*Y\xa8\x01\xec\xd5m\xa9\x84\x03\x88\xba\xe3X\u5e44#\x8f|y\xe4mC&\xd1X\xc5\f\xaa\uedd5{nC\xf9\xa6\xcdk\v\xa72\xc6}\xaf\xc5\xe7\xa0e\x02\xb26\xfe)\xf5\r\xa3\xb4}\xa5\xca\u05e6e\a\x8aI;M@\x8f\x9e.\xdeU\xb0\xb1\xb1H;\xdf.\xc9k]\xbe\xee\u912bT
NR\u077f\"Ow\xe9\xa0\x15\x8f\x16\u0205\x93\x92B\x89\u0787\"\xd28uH\xa9\x18W\xd1\x06\x81\t\

xac2\xe8N\x0ek,\x1e\xbf\xed{\xb8\xef\x9b\xd7\xe1\xc1\xef\u0740\x9d\xf7\xfc\b\xb6\x18\xb5\xa2Q\x19p\x12\xde\xebLkt{\x1d\x9cx\xc2\t\u063a\xf5\xb9?\xb8\xe0\xfc\xf3\xdf\xf1\xcaW\xbd\xfa\xeb\xe9\xaf\xfd\xdd\xdf]\x8d\xcb/\u007f\xdd3\xfe:\x9f\x80\xf93d\xfd\xf8\u01f7\u2913\x9e\v\x00\xf8\xda\u05fe\xb2\xf5\xfd\xef\u007f\xff\u05ef\xbf\xfe\x86\xb9Vr84\x167m=\a[^q%\x8e8\xfbE\xe8\x1e\xf8,\x18\xe3P\f\ap\u0468\xab\x85w\x8d\x00V\xafM\xa5\xf4\xb3\x8eG\xf8\xf8;&\xe1\xc7K\x87\xbb\xa4t\x96\x96t\x84\xb1\xea\xbcm\x80\xa6\r\xe9Z\x88\xf65\xbdRP\xc9\xef\x9a\x00\xdb\x1c\xe0\u065b>[jS\xad\xd5\xf3\xad\x03yx\xf6m\x8a\x97\xb0\xf9U:q\n\x95\xaf\xff\xa6kP[Q:\x18+\xed\u0606\xb4e\xe5N5\xc0\x8f\xf7i\xa4\xd2RF\x8f\x1eA4.KN\x19\xa8d\x89\x11\xc0\xe3\xaeKD\xe0,\x83\xce;p\xd6b\xdb\x1d?\xc4\xdd_\xfe\x14\xee\xff\xe6ux\xf2\xbe\xdb\xd1\xf4\a\x8a\x0f\xa0\x04p?\xf5C\x9dN\x8e\x99\xe9\x19l\u06b4\t\xcf{\xde)\v'\x9ex\u009f\xfc\xd6o\xbd\xf3CD\xb4\v\x00>\xfe\xf1\x8f\xf1k_\xfbknreO\xc0\xfc\x19\xb9n\xba\xe9\xfbt\xdai\xa7\v\x00|\xe2\x13\xff\xf0\x92\xff\xfc\x9f\xff\xeb5\xdf\xfe\xf6wf\xdb>\x16\x14F\xfa\xa7\xd6o\xc0\xc6\u7783\x13^\xf6F\x1cv\xea\xb9PS3\xb0\xc6\xc1\fVk\x8d\xae\x8a\xb2\x90\x06\x8f^\x81I\x12(_\x1bXI\xc1\xbcf\xb0DU\xd5O\r\xad\xf2ZMKip\xed\xcd\x1fn\xca'E\x9a\xbe1\x82*\xbb\xa1\x82cn\xb1*H=l\u02a1\x9e\xf0}\xa6\xf1\xfec\x94]J\xa8d\u02f0\x86\xe4a\xa6\x1b\x9dC}X\b\t\xe0\u01e0\xb44\x06/Fw\xc6\u7903\xc3c\xf4\b72\xde\xe5\xb0R\xdf\\R\x8a\xab&-M^\u0426\xb5B\xa9Ya\x8dNo\n\"\x82'\xef\xbb\x1dw|\xe1\xbf\xe3\xbeo~\x1e\xbb\x1f\xb8\x13f4h1\x81!\x1f$\xce\f\x11'\xe2\x84\xe6\xe6f1;;\x83C\x0e9\x14g\x9f}\xe6\u03a3\x8f>\xfa/^x\xf1%\u007f{\u0496\x93\x1e\x04\x80W\xbe\xf2\x15\xea\x13\x9f\xf8\x94\xdd_\u0331&`>Y\xff\xcb\xeb\xaf\xfe\xea/\xf8Moz\x8b\x03\x80\xab\xaf\xfe\u0605\x1f\xf9\xc8G\xae\xf9\xcew\xbe\xbba4\x1a\x19\xa5\x94\xb2\u05b5\xea3f\x0f9\x1c\x87\x9dz.\x8e\xb9\xe8Wp\xe8\xd6s\xa0{3(\x06\xabp\x81\x96i~\xa4\u04a37S\x134+d\xa5\x06\xa7]w\xf1\xab6\x95\xa6\xba\xa5\xc9\x17\xd7\x14\x19M\xab\xdb=\xad\x06h1\x118\xdcge\x0e\x15\x81\x8f\xcaA\xa7X\xc4:Y;u'\x1d\xb2\xa9\x01:\u0579\xf9\x96t\xb9\xbd6C}\x90\x03\x95\x9bb\xc6\x04\x15\xc0\x19\xa5\xc5,\xd5\f\u04cc\xabd\xa4\xae\xda&\x91\xe4\x8d\xd4\r\xba\xa4\x02o$\x9b\x05\xa5\xd2$q0\u0387C\xe4\xdd\x0e\x96\x1f\u007f\bw\xdf\xf0l\xfc\xf8\u068fc\xe7Oo\x83\xb4T\xe2Q\x99\xa2\x98\xc59g\x8c)p\xd0A\af\x1b6\x1c\x8c\x83\x0f>x\xfb\x19g<\xff\xd6\xd3N;\xf5\u00ff\xf2+\xbfz\x1d\x11-
\xc7\u07fb\xfa\xea\x8f\xd2\xeb^\xf7z\x99\\\xc5\x130\x9f\xac\xb0>\xf4\xa1\xbf\xe4\xdf\xfc\xcd7;\x00\xf8\xf4\xa7?y\xc1\xd5W_\xfd\x0f\xdf\xfa\u05b77\xec\xdc\xf9$\x98\u0649\b\xa7>.13\x93\x94\xc6\xf4\xc1\x1b\xb1\xf9\u0717\xe1\x84\u02ee\xc0\xfag\x9f\xfc?\u06fb\xd2\xe0:\xaa3{\uef7d\xbcE\xfb\xd3\xf2$\u02ca6\x8c\x1d/x\x19\xc6\xc66v\x8c\x1c\x98\x94\xcb\xd4\f\x1e\x02\xe4G25E\xe18\x19\xa8\u0250T`H\x82a\x96\x84Jf\x8a\u0270XP@\xa8\n!\xc9\x04\xc8x\x02\x89\a\x13\x1b\xe3\x15\x19[\xc6c\xcb\xd6bY\x8b%\xdbOz\u04b3\x96\xb7t\xf7\xbd\xf3\xa3\xbb_w?=9T*\x10\x92\xf4\xf7G\xb6\xf4\xd4oQ\xdfs\xbf{\xbe\xf3\x9d\x0f\x86a@OM[G\xe5\x1c\x8b\u05dc\xbc\xd3[\xfa\x83\x8bs\xc7\fj\x06\xee6}\v\xa0DNES\xe4aT\x80Y\\\r?\xc0\x02p\x80\xdb\x04t\x99\xda\xca\x0e\x93;\xd6sN\x10N\x929s\xd7\x103\xb2~GuCs\x00\xde\xfe\xbf\xbd9\xe4\xb6\xf0\xbb\xaf\u0248\xa3\xef'\xd9l\xdetx\xb4\u01aaZ\x80k\xda\xc9\xdat\x89\xbb{\x96{\x8a\xc1b\xc6)\x04\x96~\xdcV\xbc\x10\x97g\x8b\xd3[`\x1e\x9f$E\x85\x12\n
\x99\xb8\x82\xc1\xf7\xf6\xa1\xfd\xa7O\xa1\xef\u0777\xf2r\xe2\x84R\xd3\b\x8b1\b\xce\rn\x18,\x12)CCC\x03\xaa\xab\xab/,\\\xb8\xf0\xf5\x96\x96\x96\xc7ZZ6\x9es\xff\xde7\xbf\xf9\x10Y\xb6l\x99\xb8\xed\xb6\xbf\xf6\x17\xaf\x0f\xe6~\xe4\u01b7\xbe\xf5M\xf2\xe8\xa3\xff$\x00\xa0\xad\xed\xf0\x86\x1d;\x9e\xf9\xcf\xfd\xfb\x0f,<w\xee\x1c\f\xc3\xcc\xef8w(Ij\xb5M\v\xc1A)Ct\xc9\r\xf8\xe4_\u074dO\u0738\x19\x81\x92\x12h)\x1dF&\x85\\\x8d_>s/\x01\xaf\u0305\xe4\xe4\xa2\xc45\x9d&\x97\x1b\xcf*'0\x

cb88\u0324C\x90\x876\u0235\x17\x80\u02e6\xd56\xf9\xa2\xc4\xecj\x14
\x969\x94p\x81\"q\x15\x05\x85\xd7\u070b\xe4Q\x9c\xe4\xd0\x14\x94\x98\xf2=B\xecY\x1e\xf6\xa7
r\xe4\x92\xe6\xc6B@\xac\xe767\x99,\x03\xef\xca\xfeM\x1bY1\xc3\u01c6\xbbOA\xc8\xd5\xec\v\xa
b\x17\x80xL\xbe$\xd7H;\xee\xca\u0429e\x8aEe\x19JH\x8561\x81\xe1\xe3\xef\xa0\xe3\xcdW\xd0\
xf5\xf6N$\xc7F\xf2\x838\xa1`\x8cZ\x9b\tGEy\x05\xe6\u039d\x8b\x9a\x9a\xe8\xb1\r\x1b6\xec\u07f
6\ud2ed\x8a\x12<\xed\xfe\xbd\x05\v\xaeEkk+\u05ad\xfb\x94\xbf`}0\xf7\xe3j\xf1\xe0\x83_'\xdf\xfe\x
f6cV3\xa5\xa8\u07fe\xfd\xe1\xc7\xf6\xed\xdb\xf7\u064e\x8e3\x88\xc7\u3982\u015d\xa1Sj6n\x00\x
e0Z\x06\xc1\xb2J\xd4\u07f8\tsW\u0742\xaa\xeb\u05a00Z\vC3`his\xaa}\x0e\xaa\x12\xe1:\xda\xc3\
xd5\xe5\txG\xb3[w\xa6c\x80\ubc3c\xd9\xd6~kX\xb4\x10\xb3\xdf\xd0\xf9\x9a\x98\xf2\xb5\xc5S7\x9
dB\\\xad\xed\"\xbf.<WS\x9d\xcb\xc7\xcf\xec\xd1\x17N\x03\x15q\x0f7\x16\x96\xae\x9bf\xb5\xf7\x0e
\x95\"r\x9as\x9cL\x99\xbb^\x89)\x17$Y\xe9\xa3]o0\\j#\x87\xa3\x9f\xf99\x10\x17}\"\x11\xa7\x11\x8
8YER\xcd\xdap\x19e`\x8c\x81\xc92\x88\x04L\x8f\xc6\xd1wd7z\xf6\xfe7z\x0f\xfc\n\xc9D|\x16\x10g
\x14`\xcc\x04\xf4\xca\xca\n,\\\xb8\x10\x8d\x8d\r\a\x1b\x1b\x9b^\xfa\xcaW\xfe\xe1g\x84\x10O\x17\
xdb\xf7\xbe\xf7]|\xf5\xab_\xf3\x17\xa8\x0f\xe6~|\xd0x\xe2\x89\uf8e6\xa6\x06\xf6\xf1U\b\xa1\xb4\x
b6\xee\xd8x\xf0\xe0\xc1\xad\xdd\xdd\u0777\xf6\xf4\xf4`ll\f\x99\x8c\xe6\x90\x1d\xd61\x99\x10\nn\x
18\x10\x82#TZ\x89\xaa%\xabP\xbfv\x13\xeaV\u007f\x06\xe1\xca90t\x1d\x86\x96\xb1\x94/\x0e\x0f
mg\xb2\x1e\x05\x8a\x98Y,txh/Q\x92\xab84D~je6\xfa%\x17\xe0i\xce\x00\a\x96\xd3\xfa\xee\x06k7\
xbfo\xcb\xf4`78!\xff\t\xc0\xcb\x15;\x198\xb1\u032b\x18\x00j\x8d\x88c\u01247l\x8bX\xf7k3r(\x9e\xe
cf\x92;\xddIx\u007f\x0e\xe4\x11\xf3\xb8d\x89\xcc\xea\xec\x94\b\xc9\xcaE\t1\xff\xc6L\x92!@\xc1\x
98\xf9\x1a3\x93\x93\xb8|\xf6\x04\xfa\xda~\x8d\xc1\xe3\xfb1\xf4\xfeAd\xa6&g\xbcIJ((%\xd6\xf0n\x
01\xc6\x18\xa2\xd1*\u031f?\x1fUU\x95\xad\x1b6\xdc\xf4\xcb/|\xe1o\x8e\x10B.\x02@M\xcd\x1c\xf
2\xdcs\u03c8\x13'N\xa0\xa9\xa9\t\xb7\xdf~\x87\xbf8}0\xf7\xe3\xb7\t!\x04\x1eyd;\u06fe\xfd\x11\xc3
\x06\xf5g\x9fm\xdd\xd2\xd9\xd9\xf5\x8d\x83\a\x0f~\xb2\xa3\xa3\x03cc\xe3\x0e\xa0[]y\x84I\x96\x1
e\u06000\f\x04\x8a\xcbQ\xb5\xf0\xcf\xd0t\xd3\x16\u052d\u0744\x82\xf2J\xe8\x9a\x01-
\x9d\x02\x11\xe6du\xe1\xd2,{\x00\xd0C18\x15S\xe2\xa1f\xe0\x9a\xa4\xe9\xe2\xcdg\u0443_ej\x9a
\xa7\x89\xc6]\x90\xa49\x13\x1e\xf290\xba\x1bw\x04f\x1f\xe2\x90\u07e4\xcb\xedDh\xfa\u04d8\u06
7a\xc8z\x9d\x88<\x83\x96!r\xadj\x9d\x13\x8e]\xa0%\xae\xcf\x17\x989\xc9\xc8\xdd4\xc4\\\x9b\vc\x
14\x8c\u0240$;:|\xc1a\xa4\xa61y\xf9\x02FzN\xe1\xe2\u98f8\xd4\xf1\x1e\xe2\xe7;\x91\x18:\xef\u02
7e\x9d\x8d\xd1j\xf2!\x14\xdc\xd0\x05c\x8cD\"\x11,X0\x1f\xcd\xcd\xcd\u03efX\xb1\xe2\u026d[\xbfx\
xcc~\xfc\xe2\xc5K\xd8\xf3\xcf?+\xae\xbf~\xa5/1\xf4\xc1\u070f\xdfetu\x9d\xa1\x0f?\xfc\b\xfb\u044f^
\xd6,P/|\xee\xb9g?\xbfw\xef\xde\a\xde}\xb7\xad\xb6\xbb\xbb\an\x1e\xdd\xee\u04a3\x8c\x81P\t\x9c
\xeb\x10\x86\x8e`I\x05j\x96\xaf\xc3\xe2-\xf7
z\xddZHA\x15\\\x17\xa6\xfa\x85s\xcbIIx@z&\x14z=@\xdc\xee\x8d\"\xcb\xc5\u007f\x10W\x15x\xb8
x\xb7$\u041d1\x13\xfb\xf4\x007_\rOJ\xce-
\x030\x8f\xdcp\x96\x85$f1~'y\xb6\x1a\x87~q\xb6,\xe2\xfa\xbe\xbbx)\xf2\x8c\x87\x13\xaeS\xc6\f\v\
x84\xec{4\x87hg\a+\xcb2$E\x81\xa4\x98\x04|r:\x03#=\x8d\xd4\xf8(\xc6\xfa\xbbp\xf9l;\x86\xde?\x8
4\xf1\x81nL\x8e\\B21\xea=\xd1\xc8\n`\xdd\v\xc4*~\x9b\x1b\x91\x10\x86\xa1\x93\xf2\xf2\b\xaa\xaa\
xa2X\xb2\xe4\xba\xf77o\xde\xf4\xe4\x1dw\xdc\xf5\x8c-
'\xbc\uffbfS\xee\xbc\xf3\x0e\xe3\x86\x1b\xd6\xfa\x12C\x1f\xcc\xfd\xf80\xe3\xc0\x81w\u021a57\nW
\xe6\x1e~\xec\xb1\xef<\xbeg\u03de\u03df8qB\xb9x\xf1\x12\x00h\x84\x10\xd9\x01\v3+\x03Hv\nLQ
M=\xea\xd7nBs\xcb\x16\x945.\x84ZRf\x0e+H&\xad\xe6\x11\xc7\xff\xd6\xe3o\r/\xf0e\x8b\x9d9z\xf1
\xfc\xae0y\xb2qO\xf1q\xc6EL\xbe\x1a\x8e\xb6\xdb),\nO\xd3\x0e\x17\xf0(\xc3E\xbe4|Vt\x9f\xc9\x0

59\xde5N\x11\xd8,\u019a[\x8e\x9d\xf3r\x97\xcc3\xf7rY\xeb\x81\xec\fW/\xc9D
,\x9bXU\x91A\xe4\x00d\x85\x82gt\xa4'\xc60=2\x8c\xf1\xc1s\x18>\xfb>bg\x8ea\xe4\\\a\x92c1\xa4\x
a7&\xc0\r}\xc6\xe7I)\u02f6\u061b_\xad\xb1mB\x18\xe9t\x9a\x97\x95\x95\xc9s\xe6\xd4`\xfe\xfc\xf9
\xf1\x95+\xff\xfc\xdf\xee\xbf\xffk\xdf%\x84d\xc7\x02}\xe7;\xffJ\x1ex\xe0\x1f}y\xa1\x0f\xe6~|T\xf1\xe
a\xab?\xc3\xee\u077b\xc9SO\xed\xc8.\xbcw\xde\u0673\xf1\xe5\x97\u007f\xf2\x8d\xe3\xc7O\xac\x
ef\xec\xec\xc4\xe8\u8a1d\x9cR\x0f\x8d`\x1d\xbd\x05\u7812\x84py\r\xe6,[\x83\xba\x1bnA\xe5\u00
95(\xaem\x02Q$\xe8i\u00f4\b\xe0\x1c\x82\xebV\xd6\xeed\xa6\xe6\x14!W\xcb\xd1U-
\x02\xf3\xd3(\xf4K\xd5\x00\x00\v\xecIDAT\"\x1e\x83+K\x8d\xe1\xc9\xfc\x89\xfd\xe2g\aL\x9b\xe20r
&M\xb8\x9b\x99f{\x1dW\x1b\xd6\xecH\xfeL\x12;+[\xb4~n\xb8\xb2\xf3|\x1c8r\b!\x02\xaf2\x881\x06
EU!)\x12\x88\x00\u04898&c\x171q\xb1\x0f\xa3\xbd\x1d\x88u\x9d\xc4\u04296\\\x19\ua0de\x9a\xc
2\f{\a\xf7\xe9\x8b8~\xf2\x0emC@\b\x11\x9a\x96\xe1\xa1P\x98\u035bw\r\xea\xea\xea\x06\x97-
[\xba\xf3\xcb_\xfe\u04bfG\"\x95=\xeekuvv`\u07bc\x05\xfe\xe2\xf2\xc1\u070f\xdfG\xb4\xb6>\x8d\xa
d[\xb7\xb9\x8e\xf9B~\xe9\xa5\x1f\u07bdk\u05ee;\xfb\xfb\a\u059d<y\x12\xf1x<\xef\x8deN\x81qRc\
xb5\xa0\bU\v\xaf\xc7\u0715-
\xa8\xb8v\x05\xd4\xd2J\xc8\xc1\x10\x94P!\xd4\xc2\x12PU13`\x0e@p\b\xce\xc1\r\x03\xdc\xe0\x10
\xe0\x10\xdcQ\xc3\x10\xf7\xf4#\x97b&+\x13$n\xda\xc1\u017c\x13g$\x9d\xb3\v\x89lA\x93\xc2\xfd\
x1cv\xb6nj\xb6s\xfdP\xdc\xe1\xf6\x96\x99\x15q\xb3\x19\xb4w\xba\x12\x85\xd7\xfc*\x97\x02\x82e
pe\x16\x17-
\x1fx\xe1\x14\x88\x89\xdd\xe9I\x19\x98$\x81\xc9\x12\b\x03\xb4\xc9$\xa6.\xf5c|\xf0\x1c\u2f671\xd
a{\x06\xa3\xe7N#~\xfe,\x92\x89\xf8\fi\xa7\xbd\x11\x13J\u034d\x81x\x89\xae\xdc\xc7\xc2\xf2$onn
BCC\xe3\xe8\x92%\x8b\u007f\xfc\xd0C\x0f=\xa5\xaa^\x89\xe1\xe0`?jk\xeb\xfc\xc5\u40f9\x1f\xbf\x
efx\xeb\xad\xffEK\xcb\xcd\xde\xccS\x88\xf0\v/<\xff\u0653'On\xeb\xec\xec\xbc\xfe\xddw\xdb0\x1a\x
8fg}[\xbc\xa0nfw \x04\x82\xeb\xa0LFaU-
\u0095\xb5P\nK\x10(*C\xa8<\x8aP$\x8a`Y\x15\xe4\x82b(\x05\xc5\b\x14\x96B-\x8e@-
,\x01\x91\x95,gM\x00\x10av\xc8\b\xc1\u0351\xee\x9c[|\xbe\x93u\xe7v^\x12\xe25\xa1r+b\xf2\xa9i\
xa8\x95\x1d\xdb*\x0f\x9bj!y\x8a\x94\xf9T3\xe6>F\xb2lR.yd\x83e\xd6\xf8\xca*\x80:\x1b\x94p\x9aw
(1\x1b\xf8)\x01\xb3(-
JMb\x852\n\x89\x02Z:\x83\x89K\x03&h\xf7\x9e\xc1X\xdfY\x8c\xf5wc|\xa0\v\x13\x17\a\xbd\xe3\xe0
(3yn\xeb\x14E\x89W\xc6#\xf2\fN\x15B\x80R\x02YV
\xcb\x12\x1a\x1b\x1b\xd1\xdc\xdc<\x19\x8dV\xb5\xdeu\u05dd?Y\xb5jM\x9b\xfb\xf1\xdf\xff\xfe\xe3\
xb8\uffbf\xf7\x17\x90\x0f\xe6~|\x9c\xe2\x95W\xfe\v[\xb6\u070e#G\x0ea\u03de\xbd\xec\x81\a\x1e\
xb4\x95/U/\xbf\xfc\u04a7\xdb\xdbO\xdc\xdd\xd6vt\xfd\xa9S\xa7\x90H$\x90N\xa7=L\xc5L\n\xc603h
j\x82<\xa1\x14L\x0e@\t\x17A\n\x85!\a\v\xa0\x84\x8b\xa0\x16\x95!\x14\xa9B\xb8j.\xc2\x15s\x10(.\
x85\x1a.B\xb0\xb8\x14\xc1\x92\n\xa8\x85%\x90\x14\x15\xce
\x04\x0eC7\xcci\xedB\x98\x13L\x05\xcf\x0e\f\xa6\xd4\xd2e\u00dc\x96\x03
;\xcb\xd46\xb0\u02b5\x9b\xe5B@\xe3f\xe1\xd5\x00\xcd\u038c\x13v\xeaL\b\xe0\x18E\xb9\xc6\u06
f9\xac\xa9,>\u01b1\x036\v\xae\xb6\xbe\x9c\x11\xc7!\xd11,s\rL&f\x87'\x17\x02\f\xe6&\x96\x99\x9e\
u0115\xe1~\x8c\xf5w\"~\xfe,F{;\x90\x18\xee\xc3\xc4\xf0\x00&b\x17\xc0u\x87\xf3\xa6L\u029aY\x9
9\xb2A\xe21\xf52\x9d\x10E\xdeF,\xc30\x04!\x84\x84B!\x14\x15\x15\xa2\xb1\xb1\x11\x8b\x17/>\xd
d\xd4\xd4\xf8\x8b\xa5K\x97\xbd|\xd3M-
\xed\x00\xb0~\xfd\x8dt\u06f6m\xb8\xf3\xce\xcf\xf9\xea\x14\x1f\xcc\xfd\xf8C\x897\xde\xf8\x05}\xe3
\x8d_\xb2'\x9ex\xd2V\xbe\x84\u007f\xf0\x83\x176\x1f=\xfa\u07b7\x8e\x1e=\xba``\xa0\x1f\xb1X\f\x

9a\xa6\xe7-\t\xba\xb3C\x13\xe0I6\xd3\x16\x82Cp\xd7\fw\xca
\xa9A\u0221\x02H\x81\x10\xa4@\x10J\xa8\x00\x81\xc2R\x84\"QD\x1a\xe6#\u04bc\b\xa5\x9f\xb8\
x16\xe1\xb2JH\x8a\nI\x92 I\xd41\xc4\x12\x00\xb8\x01\xce-B\xc5eW\xeb1\x8d\xb2\xa8\ffm2
\x04\x19\x01d,x\xd2\f\xb3\u02d2[\x1bD\x96\n\xe2\xe6\tA\b\x93'2\xbf\xaf\xc3\xd0\xd20\xd2Ip]\x03\x
d7M\x1baSJHA\xc0A\xb9n\x82\xb3\xaeA\x18:\fC\x03\x84\x80\x1a.\x82\x1c\f\x9b\x12A-
\x8d\xf4D\x02S\xf1\u02d8\x8c\rc\xf2R?\x12\xc3\x03\x98\x8a_F*1\x8aTb\x14\xc9\u0118\xc7\a\x85
P\x06\xcaXv\x93\xb3\v\x964\x0f\x80C\x88\x19\x12Ka\x06!\x04(**FYY\x19\xae\xb9\xa6\x19MM\x8
d?^\xb0`\xc1\xd3\u06f6}\xb9\x8b\x102l?]k\xeb\xd3l\xeb\xd6m\xba\xbf2|0\xf7\xe3\x0f4\xac\x81\xd1
n\xe5\vmm\xdd\xf1\xb9\u00c7\x0f\xdf\u007f\xf2\xe4\xff-
\x1d\x19\x19\xc1\xf0\xf002\x99\x8cA\b\x04\xa5L2\v\x9d9\x1c\xac\xc5E\x10\xf7\x84g\xe7\xa2\x10\
xdc\u021b5RJ!\a\xc3P\vK\x10,-G\u025cFT^\xb3\b%\xd1Z\x14\x94U
\\Z\t\xb5\xa8\x14r(\fl\t\x80I2d\x89\x81P\xe6iN2\v\x8f\xe6f\xc2u\x03\x84\xeb\xa0\xe0\xa000\x9d\xd
611\x95\xc4\u0115+\x98\x1a\x1f\xc5\xd4D\x02\xa9\xa9\t\xe8\xa9$2\xa9\x142\x13cHO\x8cCKN\xc
2H\xa7`d\xd204\xf3\xab\x9eJ\x82k\xa6m\xb00\f\ap\xadZ\x02\xb1\xc0\x94s\xdd,\x06\x1b\xe6\xfb\x
b4\x1b\xb3\f-
\x03\xaee\xa0k\x19\xf3\u07fav\xf5\x05\x9d\x95\x8bRPP\xc7\xc7\x1c\x8e\u01ce;\x03'\xc43(\x9b\xe
b\xban\b\xc1%UUI$\x12AYY\x19\xae\xbdv~b\xf9\xf2\xa5\xaf/_\xbe\xec_n\xbe\xf93\x1e>\xfc\xb5\x
d7^!\xd5\xd5Q\xb1j\xd5\x1a\u007f1\xf8`\xee\xc7\x1fC\xec\xdc\xf9sr\xeb\xad\u007f)r8\u04ed\x1d\x
1d\x1d[\xbb\xba\xba\xe7\x0f\r\r\x05c\xb1\x18\xe2\xf11!\x047\b\xa1T\bn+(H.\x1d\xe3\xf6Gw\xeeT\x
e2\x92\xf59\xed\xa3B\x18.\x87?\x02I\r\x80\xc92\x98\xacB\t\x86\xa0\x14\x14\xa1\xb8\xbc\x1a\x85\
x955\b\x16\x95@\r\x86!)*\x18\xa5Y!$#\x020t\x18\x99\f2\xa9i\xd3\xe27\x93\x82\xd0\u04d8\xba2\x
8exl\x04\x93\x93W\x90\x9e\x9e\x86\xaee\xc0\r\xdd\x04^+\x03\xe7\u0724v\x84\xf8\xe8Tw6\x10\x1
3W\xb3\x8eS\xb4\x9c\xc9w\xbb5\xdd\xc4\xf1+\x067\x83J\x12#\x95\x95\x95(++CAA\xc1x]]]\xfb\xfa
\xf5\xeb\xf7\xdes\xcf\u059f\x12B:\xdc\xd7{\xf0\xc1\xaf\x93{\xef\xbdW\xd4\xd4\xd4\xfa7\xbf\x0f\xe6
~\xfc\xb1\xc5\xd3O?\x85S\xa7N\x91'\x9ex\u049d\xa9\xb3C\x87\x0e\u073as\xe7\xceM\xdd\xdd=\x
ab\x93\xc9\xe4\x82X,\x86\x81\x81ALOOA\xd7uLO'!\xc4\a2\xac\xbd\u028dLr\xe7\xd2\xffN\x16E\xd
e\xd1l\x1f\xe5\x02\xf5H\x02]_\xad\xb6y\u4302\xcb97y\u078d\xfd\x18\xce\x058\xe7\x90$\t\xaa\xaa
\xa2\xa8\xa8\b\xd1h\x14\x91H\xd9XII\xc9\u19a6\xa6C\x1b7\xb6\x1c\xfd\u0527n\xdaE\b\xf1\xf0\xd
fo\xbe\xf9+|\xfa\xd3\u007f\xe1\xdf\xec>\x98\xfb\xf1\xa7\x10\xb3\xa9\x18\x84\xe0\u05fe\xf8\xe2\x8
b\x1b\xdb\xdb\xdb\xe7\uaeb6\xe2\u0295\x89\xf2s\xe7\xceM\x85\xc3\xe1\u56a6\x05{{\xcf#\x91H
\x93I\xc308\f\xc3@&\x93\xf9\xe3\\x\xb9Y2!\x1e\xd31\xea\xea\xc1w\x17D\xed,;\xdf\x06f\u007f\x9f
R{\xec\x9a\xd5\xf9i\xa9V\x18cP\xd5\x00**\xcaQ[[\v\u01a4\xe9`0\xd0Y]]=\xac\xaa\u02ae\xdbn\xdb
rv\u035a\xb5\xbf\xca\u05d5\xb9k\xd7/q\xcb-
\x9f\xf1on\x1f\xcc\xfd\xf8S\x8c\x8b\x17\x87\x10\x8d\xd6`p\xb0\x8f\xd4\xd6~B\xe4\x1c\xfb\xab\x00
\x94\xec\u077b7\xd9\xdb\xdb3\xaf\xaf\xaf\xaf\xee\u0295\xc9\x06Bp\xfd\xc8H\x8c\xc6b#)JiEQQ\u0
46a\u02d7/c``\x10\x13\x13\x13\u0434\f4M\x83\xae\x1b0\f\x03\x9a\xa6\x9b<\xb7\xdbf\xe0Cj\x11\x
bf\x9a\x89\x96\x99\a\x93\x19\xf3D\xb3\t\xf2l\xb9\xbdG\xcbm6\xfa0f\xd3%\xc4j\xa42m\x87\u0740\
x9d}FB!\u02e6R\xc56\xb7b\x8cZ\xe0mf\u07a5\xa5%\xa8\xad\xadE0\x18\ucaac\xac\xec\f\x04\x82
GC\xa1`\xe7\xbcy\u05cc\xb6\xb4l8\x1f\x8d\xd6^$\x84$\xec\xabvv\x9e!\xf3\xe6\xcd\x17\x00\xb0c\x
c7S\u063cy3\xe6\u0319\xeb\xdf\xd0>\x98\xfb\xf1\xa7\x1c==\x9dhj\x9a\a!\x04v\xee\xfc9\x89\xc5b\
xec\xc9'\x9f\x16\u01cf\x1f7\xf2\x03\xa6\xa8\xb0\xeeK\xbd\xa3\xe3T\xb8\xaf\xaf\u007fN[[[@Q\xe4\
u0159Lf\xe5\u0673g\xb5\xe9\xe9diII\xf1\xba\x8b\x17/\x97\x0e\f\f`||\x1c\xba\xae!\x93\u0450L\x9a\x

```
fe/6\xd8Sk
B~\x80\xcf\xef\xed2\xc3\xfb\u071e=*\x04r\x8b\xb7\xb2,\x83Rj6\xf5\xe4\\U\x96e(\xb2\x9c\x05W\x0f
\xc7M\x88w,\x9b\x95=\xeb\xba\x0e\u039d\xe9M\xf6\xefp\xceA)\x81$IP\x14\x15\xb2,\x811\x06J\x1
9dYBqq\xb1\x88F\xa3\xa4\xa4\xa48\x1d\x8f\xc7\x0f\x8c\x8d\x8d\x9fkhh\b44\u05031\xe98\xe7\xbc
m\xf5\xea\x1b\x86W\xaf^;F\b\x19\xcd}\xcf\xf7\xddw\xaf\xb4b\xc5r\xb1l\xd92\xbed\xc9R\xf1\xfa\xe
b\xff\x83M\x9b6\xfb7\xb0\x0f\xe6~\xf8q\xb5\fW\xe0\u0529\xf7iOO/y\xfb\xed}(,\f\xd3D\xe2\n}\xfc\xf
1\xffH\xff\x86\xdf\v\x02\x10\xa9\u0514|\xe4\xc8\xe1\u04b6\xb6cJEE\xf9\xc6\u04e7;\x16\x8d\x8c\u
0116\x8d\x8c\x8c.\x1e\x1a\x1a\n'\x12\xe3\xd40\xcc\x06\x97t:\x8d\xa9\xa9)LM%a\xaa\r\t4M\x87\x
a6i\xf9\x93k\x80\u0232\fIbfc\x90p\x18sU\r\xa0\xa8\xa8\x10\x81@\x00\x92$\x81s\x81d2\t\x00P\x1
4\x05\xc1`\x00\x92$\x9bEXY\x02!$I\b\x9d,..DAA\x01\x14EE
\xa0BUU(\x8a\n\xc6X\xf6\x89UU\xa5\xc1`\x80^\xb80\xb4\xef\xfc\xf9\xf3\xa7\xc2\xe10K\xa7\u04d9
\xea\xea\xea\xe6`0T\xdc\xdf\xdfwZQ\x14c\xee\u0739\xa4\xbe\xbe\x1e\xc5\xc5E\b\x85B2\x80\xc1
c\xc7\xda\xf7LNN&\u05acY%777\x19\r\r\r\xf1H\xa4j\n\xf6\xdc\rB\x92\xb9\xeb~\xfb\xf6\x87\x95T*i\
xd4\xd7\u05cbU\xabV\x8a\xa5KW\xf8\xdap\x1f\xcc\xfd\xf0\xe3\u00cb\xae\xae3\u063d\xfb\xd7\u06
36\xedK\xbf\xf1\xb1\xd3\xd3\x13U\x87\x0e\x1d^\xb7\u007f\xff\x81\x8at:\xb5H\b\\\xdf\xdd\u074dX,\
x16\xe6\\\xd4%\x12\t#\x95JrEQKB\xa1\x90\xb5A8\x14\rc\fB\b$\x12\t=\x9dNO\xaa\xaaJM\xe0UH(\
x14\x82,+\xa3\x15\x15\x91X}}=\"\x91\x88$\x84\xb8\xd4\xd9\xd9\xf9v:\x9d\x89777K\xcd\xcdM(..\x8
6$1Z\\\\\"\x15\x16\x16\xbe;22z|\xf5\xea\x1b\fYV\xd9\aY\x8b\x84\x90\xe9<\x9b\x98G\xfe\xf9\xdb\xc
4\x0f\u007f\xf8\"\u05ad[\x87\xba\xba\x06\xff\xa6\xf2\xc1\xdc\x0f?>>\xd1\xde\xfe\x1e\x1e}\xf4\x9f\x
f1\uaaef}\x90\x93@\xf9\x89\x13\xc7n>r\xa4-
}\xe6\xcc\x19\xbd\xb2\xb2\xe2\x9a\xf2\xf2\xc8\n\xceE\x88s\xce3\x99\x8c\xb0\xb2c\x99\x10\x92\xe
c\xed=\xff\xf6\x85\v\x17\x06\xeb\xea\xeaX]]\x1d\xa9\xae\x8e\xb2\x86\x86\x06\xb2h\u0452vB\xc8\
u064f\xe3\xe71<<\x88\xeaj_&\xe8\x87\x1f~\xfc\x81\xc7\xd0\xd0\xc0\xc7\xf2u\xdd\u007f\xff\xdfbp\
xb0\xdf\xff\x03\xf9\xf1\xa1\xc7\xff\x03 \x99\rH\xa6\x0e6\x92\x00\x00\x00\x00IEND\xaeB`\x82")

// dashboardDockerfile is the Dockerfile required to build an dashboard container
// to aggregate various private network services under one easily accessible page.
var dashboardDockerfile = `
FROM mhart/alpine-node:latest

RUN \
npm install connect serve-static && \
\
echo 'var connect = require("connect");'                            > server.js && \
echo 'var serveStatic = require("serve-static");'                    >> server.js && \
echo 'connect().use(serveStatic("/dashboard")).listen(80, function(){' >> server.js && \
echo '    console.log("Server running on 80...");'                   >> server.js && \
echo '});'                                                          >> server.js

ADD {{.Network}}.json /dashboard/{{.Network}}.json
ADD index.html /dashboard/index.html
ADD puppeth.png /dashboard/puppeth.png
```

```
EXPOSE 80

CMD ["node", "/server.js"]
`

// dashboardComposefile is the docker-compose.yml file required to deploy and
// maintain an service aggregating dashboard.
var dashboardComposefile = `
version: '2'
services:
  dashboard:
    build: .
    image: {{.Network}}/dashboard{{if not .VHost}}
    ports:
      - "{{.Port}}:80"{{else}}
    environment:
      - VIRTUAL_HOST={{.VHost}}{{end}}
    restart: always
`

// deployDashboard deploys a new dashboard container to a remote machine via SSH,
// docker and docker-compose. If an instance with the specified network name
// already exists there, it will be overwritten!
func deployDashboard(client *sshClient, network string, port int, vhost string, services
map[string]string, conf *config, ethstats bool) ([]byte, error) {
// Generate the content to upload to the server
workdir := fmt.Sprintf("%d", rand.Int63())
files := make(map[string][]byte)

dockerfile := new(bytes.Buffer)
template.Must(template.New("").Parse(dashboardDockerfile)).Execute(dockerfile,
map[string]interface{}{
"Network": network,
})
files[filepath.Join(workdir, "Dockerfile")] = dockerfile.Bytes()

composefile := new(bytes.Buffer)
template.Must(template.New("").Parse(dashboardComposefile)).Execute(composefile,
map[string]interface{}{
"Network": network,
"Port":    port,
```

```
"VHost":   vhost,
})
files[filepath.Join(workdir, "docker-compose.yaml")] = composefile.Bytes()

statsLogin := fmt.Sprintf("yournode:%s", conf.ethstats)
if !ethstats {
statsLogin = ""
}
indexfile := new(bytes.Buffer)
template.Must(template.New("").Parse(dashboardContent)).Execute(indexfile,
map[string]interface{}{
"Network":           network,
"NetworkID":         conf.genesis.Config.ChainId,
"NetworkTitle":      strings.Title(network),
"EthstatsPage":      services["ethstats"],
"ExplorerPage":      services["explorer"],
"WalletPage":        services["wallet"],
"FaucetPage":        services["faucet"],
"GethGenesis":       network + ".json",
"BootnodesFull":     conf.bootFull,
"BootnodesLight":    conf.bootLight,
"BootnodesFullFlat":  strings.Join(conf.bootFull, ","),
"BootnodesLightFlat": strings.Join(conf.bootLight, ","),
"Ethstats":          statsLogin,
})
files[filepath.Join(workdir, "index.html")] = indexfile.Bytes()

genesis, _ := conf.genesis.MarshalJSON()
files[filepath.Join(workdir, network+".json")] = genesis

files[filepath.Join(workdir, "puppeth.png")] = dashboardMascot

// Upload the deployment files to the remote server (and clean up afterwards)
if out, err := client.Upload(files); err != nil {
return out, err
}
defer client.Run("rm -rf " + workdir)

// Build and deploy the dashboard service
return nil, client.Stream(fmt.Sprintf("cd %s && docker-compose -p %s up -d --build", workdir,
network))
}
```

```
// dashboardInfos is returned from an dashboard status check to allow reporting
// various configuration parameters.
type dashboardInfos struct {
host string
port int
}

// String implements the stringer interface.
func (info *dashboardInfos) String() string {
return fmt.Sprintf("host=%s, port=%d", info.host, info.port)
}

// checkDashboard does a health-check against a dashboard container to verify if
// it's running, and if yes, gathering a collection of useful infos about it.
func checkDashboard(client *sshClient, network string) (*dashboardInfos, error) {
// Inspect a possible ethstats container on the host
infos, err := inspectContainer(client, fmt.Sprintf("%s_dashboard_1", network))
if err != nil {
return nil, err
}
if !infos.running {
return nil, ErrServiceOffline
}
// Resolve the port from the host, or the reverse proxy
port := infos.portmap["80/tcp"]
if port == 0 {
if proxy, _ := checkNginx(client, network); proxy != nil {
port = proxy.port
}
}
if port == 0 {
return nil, ErrNotExposed
}
// Resolve the host from the reverse-proxy and configure the connection string
host := infos.envvars["VIRTUAL_HOST"]
if host == "" {
host = client.server
}
// Run a sanity check to see if the port is reachable
if err = checkPort(host, port); err != nil {
log.Warn("Dashboard service seems unreachable", "server", host, "port", port, "err", err)
```

```
}
// Container available, assemble and return the useful infos
return &dashboardInfos{
host: host,
port: port,
}, nil
}

75:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\module_ethstats.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bytes"
"fmt"
"math/rand"
"path/filepath"
"strings"
"text/template"

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

// ethstatsDockerfile is the Dockerfile required to build an ethstats backend
// and associated monitoring site.
var ethstatsDockerfile = `
FROM mhart/alpine-node:latest

RUN \
  apk add --update git                                  && \
  git clone --depth=1 https://github.com/karalabe/eth-netstats && \
apk del git && rm -rf /var/cache/apk/*                  && \
\
  cd /eth-netstats && npm install && npm install -g grunt-cli && grunt

WORKDIR /eth-netstats
EXPOSE 3000

RUN echo 'module.exports = {trusted: [{{.Trusted}}], banned: []};' > lib/utils/config.js

CMD ["npm", "start"]
```

```
`

// ethstatsComposefile is the docker-compose.yml file required to deploy and
// maintain an ethstats monitoring site.
var ethstatsComposefile = `
version: '2'
services:
  ethstats:
    build: .
    image: {{.Network}}/ethstats{{if not .VHost}}
    ports:
      - "{{.Port}}:3000"{{end}}
    environment:
      - WS_SECRET={{.Secret}}{{if .VHost}}
      - VIRTUAL_HOST={{.VHost}}{{end}}
    restart: always
`

// deployEthstats deploys a new ethstats container to a remote machine via SSH,
// docker and docker-compose. If an instance with the specified network name
// already exists there, it will be overwritten!
func deployEthstats(client *sshClient, network string, port int, secret string, vhost string, trusted
[]string) ([]byte, error) {
// Generate the content to upload to the server
workdir := fmt.Sprintf("%d", rand.Int63())
files := make(map[string][]byte)

for i, address := range trusted {
trusted[i] = fmt.Sprintf("\"%s\"", address)
}

dockerfile := new(bytes.Buffer)
template.Must(template.New("").Parse(ethstatsDockerfile)).Execute(dockerfile,
map[string]interface{}{
"Trusted": strings.Join(trusted, ", "),
})
files[filepath.Join(workdir, "Dockerfile")] = dockerfile.Bytes()

composefile := new(bytes.Buffer)
template.Must(template.New("").Parse(ethstatsComposefile)).Execute(composefile,
map[string]interface{}{
"Network": network,
```

```
"Port":    port,
"Secret":  secret,
"VHost":   vhost,
})
files[filepath.Join(workdir, "docker-compose.yaml")] = composefile.Bytes()

// Upload the deployment files to the remote server (and clean up afterwards)
if out, err := client.Upload(files); err != nil {
return out, err
}
defer client.Run("rm -rf " + workdir)

// Build and deploy the ethstats service
return nil, client.Stream(fmt.Sprintf("cd %s && docker-compose -p %s up -d --build", workdir,
network))
}

// ethstatsInfos is returned from an ethstats status check to allow reporting
// various configuration parameters.
type ethstatsInfos struct {
host   string
port   int
secret string
config string
}

// String implements the stringer interface.
func (info *ethstatsInfos) String() string {
return fmt.Sprintf("host=%s, port=%d, secret=%s", info.host, info.port, info.secret)
}

// checkEthstats does a health-check against an ethstats server to verify whether
// it's running, and if yes, gathering a collection of useful infos about it.
func checkEthstats(client *sshClient, network string) (*ethstatsInfos, error) {
// Inspect a possible ethstats container on the host
infos, err := inspectContainer(client, fmt.Sprintf("%s_ethstats_1", network))
if err != nil {
return nil, err
}
if !infos.running {
return nil, ErrServiceOffline
}
```

```
// Resolve the port from the host, or the reverse proxy
port := infos.portmap["3000/tcp"]
if port == 0 {
if proxy, _ := checkNginx(client, network); proxy != nil {
port = proxy.port
}
}
if port == 0 {
return nil, ErrNotExposed
}
// Resolve the host from the reverse-proxy and configure the connection string
host := infos.envvars["VIRTUAL_HOST"]
if host == "" {
host = client.server
}
secret := infos.envvars["WS_SECRET"]
config := fmt.Sprintf("%s@%s", secret, host)
if port != 80 && port != 443 {
config += fmt.Sprintf(":%d", port)
}
// Run a sanity check to see if the port is reachable
if err = checkPort(host, port); err != nil {
log.Warn("Ethstats service seems unreachable", "server", host, "port", port, "err", err)
}
// Container available, assemble and return the useful infos
return &ethstatsInfos{
host:   host,
port:   port,
secret: secret,
config: config,
}, nil
}

76:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\module_faucet.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bytes"
"fmt"
"html/template"
```

```
"math/rand"
"path/filepath"
"strconv"
"strings"

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

// faucetDockerfile is the Dockerfile required to build an faucet container to
// grant crypto tokens based on GitHub authentications.
var faucetDockerfile = `
FROM alpine:latest

RUN mkdir /go
ENV GOPATH /go

RUN \
  apk add --update git go make gcc musl-dev ca-certificates linux-headers                    && \
mkdir -p $GOPATH/src/github.com/ethereum                                                     && \
(cd $GOPATH/src/github.com/ethereum && git clone --depth=1 https://github.com/ethereum/go-
ethereum) && \
  go build -v github.com/ethereum/go-ethereum/cmd/faucet                                        && \
  apk del git go make gcc musl-dev linux-headers                                             && \
  rm -rf $GOPATH && rm -rf /var/cache/apk/*

ADD genesis.json /genesis.json
ADD account.json /account.json
ADD account.pass /account.pass

EXPOSE 8080

CMD [ \
"/faucet", "--genesis", "/genesis.json", "--network", "{{.NetworkID}}", "--bootnodes",
"{{.Bootnodes}}", "--ethstats", "{{.Ethstats}}", "--ethport", "{{.EthPort}}", \
"--faucet.name", "{{.FaucetName}}", "--faucet.amount", "{{.FaucetAmount}}", "--faucet.minutes",
"{{.FaucetMinutes}}", "--faucet.tiers", "{{.FaucetTiers}}",          \
"--github.user", "{{.GitHubUser}}", "--github.token", "{{.GitHubToken}}", "--account.json",
"/account.json", "--account.pass", "/account.pass"                      \
{{if .CaptchaToken}}, "--captcha.token", "{{.CaptchaToken}}", "--captcha.secret",
"{{.CaptchaSecret}}"{{end}}                                              \
]`
```

```
// faucetComposefile is the docker-compose.yml file required to deploy and maintain
// a crypto faucet.
var faucetComposefile = `
version: '2'
services:
  faucet:
    build: .
    image: {{.Network}}/faucet
    ports:
      - "{{.EthPort}}:{{.EthPort}}"{{if not .VHost}}
      - "{{.ApiPort}}:8080"{{end}}
    volumes:
      - {{.Datadir}}:/root/.faucet
    environment:
      - ETH_PORT={{.EthPort}}
      - ETH_NAME={{.EthName}}
      - FAUCET_AMOUNT={{.FaucetAmount}}
      - FAUCET_MINUTES={{.FaucetMinutes}}
      - FAUCET_TIERS={{.FaucetTiers}}
      - GITHUB_USER={{.GitHubUser}}
      - GITHUB_TOKEN={{.GitHubToken}}
      - CAPTCHA_TOKEN={{.CaptchaToken}}
      - CAPTCHA_SECRET={{.CaptchaSecret}}{{if .VHost}}
      - VIRTUAL_HOST={{.VHost}}
      - VIRTUAL_PORT=8080{{end}}
    restart: always
`

// deployFaucet deploys a new faucet container to a remote machine via SSH,
// docker and docker-compose. If an instance with the specified network name
// already exists there, it will be overwritten!
func deployFaucet(client *sshClient, network string, bootnodes []string, config *faucetInfos) ([]byte,
error) {
// Generate the content to upload to the server
workdir := fmt.Sprintf("%d", rand.Int63())
files := make(map[string][]byte)

dockerfile := new(bytes.Buffer)
template.Must(template.New("").Parse(faucetDockerfile)).Execute(dockerfile,
map[string]interface{}{
"NetworkID":     config.node.network,
"Bootnodes":     strings.Join(bootnodes, ","),
```

```
"Ethstats":      config.node.ethstats,
"EthPort":       config.node.portFull,
"GitHubUser":    config.githubUser,
"GitHubToken":   config.githubToken,
"CaptchaToken":  config.captchaToken,
"CaptchaSecret": config.captchaSecret,
"FaucetName":    strings.Title(network),
"FaucetAmount":  config.amount,
"FaucetMinutes": config.minutes,
"FaucetTiers":   config.tiers,
})
files[filepath.Join(workdir, "Dockerfile")] = dockerfile.Bytes()

composefile := new(bytes.Buffer)
template.Must(template.New("").Parse(faucetComposefile)).Execute(composefile,
map[string]interface{}{
"Network":       network,
"Datadir":       config.node.datadir,
"VHost":         config.host,
"ApiPort":       config.port,
"EthPort":       config.node.portFull,
"EthName":       config.node.ethstats[:strings.Index(config.node.ethstats, ":")],
"GitHubUser":    config.githubUser,
"GitHubToken":   config.githubToken,
"CaptchaToken":  config.captchaToken,
"CaptchaSecret": config.captchaSecret,
"FaucetAmount":  config.amount,
"FaucetMinutes": config.minutes,
"FaucetTiers":   config.tiers,
})
files[filepath.Join(workdir, "docker-compose.yaml")] = composefile.Bytes()

files[filepath.Join(workdir, "genesis.json")] = []byte(config.node.genesis)
files[filepath.Join(workdir, "account.json")] = []byte(config.node.keyJSON)
files[filepath.Join(workdir, "account.pass")] = []byte(config.node.keyPass)

// Upload the deployment files to the remote server (and clean up afterwards)
if out, err := client.Upload(files); err != nil {
return out, err
}
defer client.Run("rm -rf " + workdir)
```

```
// Build and deploy the faucet service
return nil, client.Stream(fmt.Sprintf("cd %s && docker-compose -p %s up -d --build", workdir,
network))
}

// faucetInfos is returned from an faucet status check to allow reporting various
// configuration parameters.
type faucetInfos struct {
node          *nodeInfos
host          string
port          int
amount        int
minutes       int
tiers         int
githubUser    string
githubToken   string
captchaToken  string
captchaSecret string
}

// String implements the stringer interface.
func (info *faucetInfos) String() string {
return fmt.Sprintf("host=%s, api=%d, eth=%d, amount=%d, minutes=%d, tiers=%d, github=%s,
captcha=%v, ethstats=%s", info.host, info.port, info.node.portFull, info.amount, info.minutes,
info.tiers, info.githubUser, info.captchaToken != "", info.node.ethstats)
}

// checkFaucet does a health-check against an faucet server to verify whether
// it's running, and if yes, gathering a collection of useful infos about it.
func checkFaucet(client *sshClient, network string) (*faucetInfos, error) {
// Inspect a possible faucet container on the host
infos, err := inspectContainer(client, fmt.Sprintf("%s_faucet_1", network))
if err != nil {
return nil, err
}
if !infos.running {
return nil, ErrServiceOffline
}
// Resolve the port from the host, or the reverse proxy
port := infos.portmap["8080/tcp"]
if port == 0 {
if proxy, _ := checkNginx(client, network); proxy != nil {
```

```
port = proxy.port
}
}
if port == 0 {
return nil, ErrNotExposed
}
// Resolve the host from the reverse-proxy and the config values
host := infos.envvars["VIRTUAL_HOST"]
if host == "" {
host = client.server
}
amount, _ := strconv.Atoi(infos.envvars["FAUCET_AMOUNT"])
minutes, _ := strconv.Atoi(infos.envvars["FAUCET_MINUTES"])
tiers, _ := strconv.Atoi(infos.envvars["FAUCET_TIERS"])

// Retrieve the funding account informations
var out []byte
keyJSON, keyPass := "", ""
if out, err = client.Run(fmt.Sprintf("docker exec %s_faucet_1 cat /account.json", network)); err ==
nil {
keyJSON = string(bytes.TrimSpace(out))
}
if out, err = client.Run(fmt.Sprintf("docker exec %s_faucet_1 cat /account.pass", network)); err ==
nil {
keyPass = string(bytes.TrimSpace(out))
}
// Run a sanity check to see if the port is reachable
if err = checkPort(host, port); err != nil {
log.Warn("Faucet service seems unreachable", "server", host, "port", port, "err", err)
}
// Container available, assemble and return the useful infos
return &faucetInfos{
node: &nodeInfos{
datadir:  infos.volumes["/root/.faucet"],
portFull: infos.portmap[infos.envvars["ETH_PORT"]+"/tcp"],
ethstats: infos.envvars["ETH_NAME"],
keyJSON:  keyJSON,
keyPass:  keyPass,
},
host:          host,
port:          port,
amount:        amount,
```

```
minutes:       minutes,
tiers:         tiers,
githubUser:    infos.envvars["GITHUB_USER"],
githubToken:   infos.envvars["GITHUB_TOKEN"],
captchaToken:  infos.envvars["CAPTCHA_TOKEN"],
captchaSecret: infos.envvars["CAPTCHA_SECRET"],
}, nil
}

77:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\module_nginx.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bytes"
"fmt"
"html/template"
"math/rand"
"path/filepath"

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

// nginxDockerfile is theis the Dockerfile required to build an nginx reverse-
// proxy.
var nginxDockerfile = `FROM jwilder/nginx-proxy`

// nginxComposefile is the docker-compose.yml file required to deploy and maintain
// an nginx reverse-proxy. The proxy is responsible for exposing one or more HTTP
// services running on a single host.
var nginxComposefile = `
version: '2'
services:
  nginx:
    build: .
    image: {{.Network}}/nginx
    ports:
      - "{{.Port}}:80"
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock:ro
    restart: always
```

`

// deployNginx deploys a new nginx reverse-proxy container to expose one or more
// HTTP services running on a single host. If an instance with the specified
// network name already exists there, it will be overwritten!
func deployNginx(client *sshClient, network string, port int) ([]byte, error) {
log.Info("Deploying nginx reverse-proxy", "server", client.server, "port", port)

// Generate the content to upload to the server
workdir := fmt.Sprintf("%d", rand.Int63())
files := make(map[string][]byte)

dockerfile := new(bytes.Buffer)
template.Must(template.New("").Parse(nginxDockerfile)).Execute(dockerfile, nil)
files[filepath.Join(workdir, "Dockerfile")] = dockerfile.Bytes()

composefile := new(bytes.Buffer)
template.Must(template.New("").Parse(nginxComposefile)).Execute(composefile,
map[string]interface{}{
"Network": network,
"Port":   port,
})
files[filepath.Join(workdir, "docker-compose.yaml")] = composefile.Bytes()

// Upload the deployment files to the remote server (and clean up afterwards)
if out, err := client.Upload(files); err != nil {
return out, err
}
defer client.Run("rm -rf " + workdir)

// Build and deploy the ethstats service
return nil, client.Stream(fmt.Sprintf("cd %s && docker-compose -p %s up -d --build", workdir,
network))
}

// nginxInfos is returned from an nginx reverse-proxy status check to allow
// reporting various configuration parameters.
type nginxInfos struct {
port int
}

// String implements the stringer interface.

```
func (info *nginxInfos) String() string {
return fmt.Sprintf("port=%d", info.port)
}

// checkNginx does a health-check against an nginx reverse-proxy to verify whether
// it's running, and if yes, gathering a collection of useful infos about it.
func checkNginx(client *sshClient, network string) (*nginxInfos, error) {
// Inspect a possible nginx container on the host
infos, err := inspectContainer(client, fmt.Sprintf("%s_nginx_1", network))
if err != nil {
return nil, err
}
if !infos.running {
return nil, ErrServiceOffline
}
// Container available, assemble and return the useful infos
return &nginxInfos{
port: infos.portmap["80/tcp"],
}, nil
}

78:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\module_node.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bytes"
"fmt"
"math/rand"
"path/filepath"
"strconv"
"strings"
"text/template"

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

// nodeDockerfile is the Dockerfile required to run an Ethereum node.
var nodeDockerfile = `
FROM ethereum/client-go:alpine-develop
```

```
ADD genesis.json /genesis.json
{{if .Unlock}}
ADD signer.json /signer.json
ADD signer.pass /signer.pass
{{end}}
RUN \
  echo '/geth init /genesis.json' > geth.sh && \{{if .Unlock}}
echo 'mkdir -p /root/.ethereum/keystore/ && cp /signer.json /root/.ethereum/keystore/' >> geth.sh
&& \{{end}}
echo $'/geth --networkid {{.NetworkID}} --cache 512 --port {{.Port}} --maxpeers {{.Peers}}
{{.LightFlag}} --ethstats \'{{.Ethstats}}\' {{if .BootV4}}--bootnodesv4 {{.BootV4}}{{end}} {{if .BootV5}}-
-bootnodesv5 {{.BootV5}}{{end}} {{if .Etherbase}}--etherbase {{.Etherbase}} --mine{{end}}{{if
.Unlock}}--unlock 0 --password /signer.pass --mine{{end}} --targetgaslimit {{.GasTarget}} --
gasprice {{.GasPrice}}' >> geth.sh

ENTRYPOINT ["/bin/sh", "geth.sh"]
`

// nodeComposefile is the docker-compose.yml file required to deploy and maintain
// an Ethereum node (bootnode or miner for now).
var nodeComposefile = `
version: '2'
services:
  {{.Type}}:
    build: .
    image: {{.Network}}/{{.Type}}
    ports:
      - "{{.FullPort}}:{{.FullPort}}"
      - "{{.FullPort}}:{{.FullPort}}/udp"{{if .Light}}
      - "{{.LightPort}}:{{.LightPort}}/udp"{{end}}
    volumes:
      - {{.Datadir}}:/root/.ethereum
    environment:
      - FULL_PORT={{.FullPort}}/tcp
      - LIGHT_PORT={{.LightPort}}/udp
      - TOTAL_PEERS={{.TotalPeers}}
      - LIGHT_PEERS={{.LightPeers}}
      - STATS_NAME={{.Ethstats}}
      - MINER_NAME={{.Etherbase}}
      - GAS_TARGET={{.GasTarget}}
      - GAS_PRICE={{.GasPrice}}
    restart: always
```

```

// deployNode deploys a new Ethereum node container to a remote machine via SSH,
// docker and docker-compose. If an instance with the specified network name
// already exists there, it will be overwritten!
func deployNode(client *sshClient, network string, bootv4, bootv5 []string, config *nodeInfos)
([]byte, error) {
kind := "sealnode"
if config.keyJSON == "" && config.etherbase == "" {
kind = "bootnode"
bootv4 = make([]string, 0)
bootv5 = make([]string, 0)
}
// Generate the content to upload to the server
workdir := fmt.Sprintf("%d", rand.Int63())
files := make(map[string][]byte)

lightFlag := ""
if config.peersLight > 0 {
lightFlag = fmt.Sprintf("--lightpeers=%d --lightserv=50", config.peersLight)
}
dockerfile := new(bytes.Buffer)
template.Must(template.New("").Parse(nodeDockerfile)).Execute(dockerfile, map[string]interface{}{
"NetworkID": config.network,
"Port":      config.portFull,
"Peers":     config.peersTotal,
"LightFlag": lightFlag,
"BootV4":    strings.Join(bootv4, ","),
"BootV5":    strings.Join(bootv5, ","),
"Ethstats":  config.ethstats,
"Etherbase": config.etherbase,
"GasTarget": uint64(1000000 * config.gasTarget),
"GasPrice":  uint64(1000000000 * config.gasPrice),
"Unlock":    config.keyJSON != "",
})
files[filepath.Join(workdir, "Dockerfile")] = dockerfile.Bytes()

composefile := new(bytes.Buffer)
template.Must(template.New("").Parse(nodeComposefile)).Execute(composefile,
map[string]interface{}{
"Type":       kind,
"Datadir":    config.datadir,
```

```
"Network":    network,
"FullPort":   config.portFull,
"TotalPeers": config.peersTotal,
"Light":      config.peersLight > 0,
"LightPort":  config.portFull + 1,
"LightPeers": config.peersLight,
"Ethstats":   config.ethstats[:strings.Index(config.ethstats, ":")],
"Etherbase":  config.etherbase,
"GasTarget":  config.gasTarget,
"GasPrice":   config.gasPrice,
})
files[filepath.Join(workdir, "docker-compose.yaml")] = composefile.Bytes()

//genesisfile, _ := json.MarshalIndent(config.genesis, "", "  ")
files[filepath.Join(workdir, "genesis.json")] = []byte(config.genesis)

if config.keyJSON != "" {
files[filepath.Join(workdir, "signer.json")] = []byte(config.keyJSON)
files[filepath.Join(workdir, "signer.pass")] = []byte(config.keyPass)
}
// Upload the deployment files to the remote server (and clean up afterwards)
if out, err := client.Upload(files); err != nil {
return out, err
}
defer client.Run("rm -rf " + workdir)

// Build and deploy the boot or seal node service
return nil, client.Stream(fmt.Sprintf("cd %s && docker-compose -p %s up -d --build", workdir,
network))
}

// nodeInfos is returned from a boot or seal node status check to allow reporting
// various configuration parameters.
type nodeInfos struct {
genesis    []byte
network    int64
datadir    string
ethstats   string
portFull   int
portLight  int
enodeFull  string
enodeLight string
```

```
peersTotal int
peersLight int
etherbase  string
keyJSON    string
keyPass    string
gasTarget  float64
gasPrice   float64
}

// String implements the stringer interface.
func (info *nodeInfos) String() string {
discv5 := ""
if info.peersLight > 0 {
discv5 = fmt.Sprintf(", portv5=%d", info.portLight)
}
return fmt.Sprintf("port=%d%s, datadir=%s, peers=%d, lights=%d, ethstats=%s, gastarget=%0.3f
MGas, gasprice=%0.3f GWei",
info.portFull, discv5, info.datadir, info.peersTotal, info.peersLight, info.ethstats, info.gasTarget,
info.gasPrice)
}

// checkNode does a health-check against an boot or seal node server to verify
// whether it's running, and if yes, whether it's responsive.
func checkNode(client *sshClient, network string, boot bool) (*nodeInfos, error) {
kind := "bootnode"
if !boot {
kind = "sealnode"
}
// Inspect a possible bootnode container on the host
infos, err := inspectContainer(client, fmt.Sprintf("%s_%s_1", network, kind))
if err != nil {
return nil, err
}
if !infos.running {
return nil, ErrServiceOffline
}
// Resolve a few types from the environmental variables
totalPeers, _ := strconv.Atoi(infos.envvars["TOTAL_PEERS"])
lightPeers, _ := strconv.Atoi(infos.envvars["LIGHT_PEERS"])
gasTarget, _ := strconv.ParseFloat(infos.envvars["GAS_TARGET"], 64)
gasPrice, _ := strconv.ParseFloat(infos.envvars["GAS_PRICE"], 64)
```

```
// Container available, retrieve its node ID and its genesis json
var out []byte
if out, err = client.Run(fmt.Sprintf("docker exec %s_%s_1 /geth --exec admin.nodeInfo.id attach",
network, kind)); err != nil {
return nil, ErrServiceUnreachable
}
id := bytes.Trim(bytes.TrimSpace(out), "\"")

if out, err = client.Run(fmt.Sprintf("docker exec %s_%s_1 cat /genesis.json", network, kind)); err !=
nil {
return nil, ErrServiceUnreachable
}
genesis := bytes.TrimSpace(out)

keyJSON, keyPass := "", ""
if out, err = client.Run(fmt.Sprintf("docker exec %s_%s_1 cat /signer.json", network, kind)); err ==
nil {
keyJSON = string(bytes.TrimSpace(out))
}
if out, err = client.Run(fmt.Sprintf("docker exec %s_%s_1 cat /signer.pass", network, kind)); err ==
nil {
keyPass = string(bytes.TrimSpace(out))
}
// Run a sanity check to see if the devp2p is reachable
port := infos.portmap[infos.envvars["FULL_PORT"]]
if err = checkPort(client.server, port); err != nil {
log.Warn(fmt.Sprintf("%s devp2p port seems unreachable", strings.Title(kind)), "server",
client.server, "port", port, "err", err)
}
// Assemble and return the useful infos
stats := &nodeInfos{
genesis:    genesis,
datadir:    infos.volumes["/root/.ethereum"],
portFull:   infos.portmap[infos.envvars["FULL_PORT"]],
portLight:  infos.portmap[infos.envvars["LIGHT_PORT"]],
peersTotal: totalPeers,
peersLight: lightPeers,
ethstats:   infos.envvars["STATS_NAME"],
etherbase:  infos.envvars["MINER_NAME"],
keyJSON:    keyJSON,
keyPass:    keyPass,
gasTarget:  gasTarget,
```

```
gasPrice:   gasPrice,
}
stats.enodeFull = fmt.Sprintf("enode://%s@%s:%d", id, client.address, stats.portFull)
if stats.portLight != 0 {
stats.enodeLight = fmt.Sprintf("enode://%s@%s:%d?discport=%d", id, client.address,
stats.portFull, stats.portLight)
}
return stats, nil
}

79:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\puppeth.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// puppeth is a command to assemble and maintain private networks.
package main

import (
"math/rand"
"os"
"time"

"github.com/ethereum/go-ethereum/log"
"gopkg.in/urfave/cli.v1"
)

// main is just a boring entry point to set up the CLI app.
func main() {
app := cli.NewApp()
app.Name = "puppeth"
app.Usage = "assemble and maintain private Ethereum networks"
app.Flags = []cli.Flag{
cli.StringFlag{
Name:  "network",
Usage: "name of the network to administer",
},
cli.IntFlag{
Name:  "loglevel",
Value: 4,
Usage: "log level to emit to the screen",
},
}
app.Action = func(c *cli.Context) error {
```

```
// Set up the logger to print everything and the random generator
log.Root().SetHandler(log.LvlFilterHandler(log.Lvl(c.Int("loglevel")), log.StreamHandler(os.Stdout,
log.TerminalFormat(true))))
rand.Seed(time.Now().UnixNano())

// Start the wizard and relinquish control
makeWizard(c.String("network")).run()
return nil
}
app.Run(os.Args)
}

80:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\ssh.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bufio"
"bytes"
"errors"
"fmt"
"io/ioutil"
"net"
"os"
"os/user"
"path/filepath"
"strings"
"syscall"

"github.com/ethereum/go-ethereum/log"
"golang.org/x/crypto/ssh"
"golang.org/x/crypto/ssh/terminal"
)

// sshClient is a small wrapper around Go's SSH client with a few utility methods
// implemented on top.
type sshClient struct {
server  string // Server name or IP without port number
address string // IP address of the remote server
pubkey  []byte // RSA public key to authenticate the server
client  *ssh.Client
```

```
logger  log.Logger
}

// dial establishes an SSH connection to a remote node using the current user and
// the user's configured private RSA key. If that fails, password authentication
// is fallen back to. The caller may override the login user via user@server:port.
func dial(server string, pubkey []byte) (*sshClient, error) {
// Figure out a label for the server and a logger
label := server
if strings.Contains(label, ":") {
label = label[:strings.Index(label, ":")]
}
login := ""
if strings.Contains(server, "@") {
login = label[:strings.Index(label, "@")]
label = label[strings.Index(label, "@")+1:]
server = server[strings.Index(server, "@")+1:]
}
logger := log.New("server", label)
logger.Debug("Attempting to establish SSH connection")

user, err := user.Current()
if err != nil {
return nil, err
}
if login == "" {
login = user.Username
}
// Configure the supported authentication methods (private key and password)
var auths []ssh.AuthMethod

path := filepath.Join(user.HomeDir, ".ssh", "id_rsa")
if buf, err := ioutil.ReadFile(path); err != nil {
log.Warn("No SSH key, falling back to passwords", "path", path, "err", err)
} else {
key, err := ssh.ParsePrivateKey(buf)
if err != nil {
log.Warn("Bad SSH key, falling back to passwords", "path", path, "err", err)
} else {
auths = append(auths, ssh.PublicKeys(key))
}
}
```

```
auths = append(auths, ssh.PasswordCallback(func() (string, error) {
fmt.Printf("What's the login password for %s at %s? (won't be echoed)\n> ", login, server)
blob, err := terminal.ReadPassword(int(syscall.Stdin))

fmt.Println()
return string(blob), err
}))
// Resolve the IP address of the remote server
addr, err := net.LookupHost(label)
if err != nil {
return nil, err
}
if len(addr) == 0 {
return nil, errors.New("no IPs associated with domain")
}
// Try to dial in to the remote server
logger.Trace("Dialing remote SSH server", "user", login)
if !strings.Contains(server, ":") {
server += ":22"
}
keycheck := func(hostname string, remote net.Addr, key ssh.PublicKey) error {
// If no public key is known for SSH, ask the user to confirm
if pubkey == nil {
fmt.Printf("The authenticity of host '%s (%s)' can't be established.\", hostname, remote)
fmt.Printf("SSH key fingerprint is %s [MD5]\n", ssh.FingerprintLegacyMD5(key))
fmt.Printf("Are you sure you want to continue connecting (yes/no)? ")

text, err := bufio.NewReader(os.Stdin).ReadString('\n')
switch {
case err != nil:
return err
case strings.TrimSpace(text) == "yes":
pubkey = key.Marshal()
return nil
default:
return fmt.Errorf("unknown auth choice: %v", text)
}
}
// If a public key exists for this SSH server, check that it matches
if bytes.Compare(pubkey, key.Marshal()) == 0 {
return nil
}
```

```
// We have a mismatch, forbid connecting
return errors.New("ssh key mismatch, readd the machine to update")
}
client, err := ssh.Dial("tcp", server, &ssh.ClientConfig{User: login, Auth: auths, HostKeyCallback:
keycheck})
if err != nil {
return nil, err
}
// Connection established, return our utility wrapper
c := &sshClient{
server:  label,
address: addr[0],
pubkey:  pubkey,
client:  client,
logger:  logger,
}
if err := c.init(); err != nil {
client.Close()
return nil, err
}
return c, nil
}

// init runs some initialization commands on the remote server to ensure it's
// capable of acting as puppeth target.
func (client *sshClient) init() error {
client.logger.Debug("Verifying if docker is available")
if out, err := client.Run("docker version"); err != nil {
if len(out) == 0 {
return err
}
return fmt.Errorf("docker configured incorrectly: %s", out)
}
client.logger.Debug("Verifying if docker-compose is available")
if out, err := client.Run("docker-compose version"); err != nil {
if len(out) == 0 {
return err
}
return fmt.Errorf("docker-compose configured incorrectly: %s", out)
}
return nil
}
```

```
// Close terminates the connection to an SSH server.
func (client *sshClient) Close() error {
return client.client.Close()
}

// Run executes a command on the remote server and returns the combined output
// along with any error status.
func (client *sshClient) Run(cmd string) ([]byte, error) {
// Establish a single command session
session, err := client.client.NewSession()
if err != nil {
return nil, err
}
defer session.Close()

// Execute the command and return any output
client.logger.Trace("Running command on remote server", "cmd", cmd)
return session.CombinedOutput(cmd)
}

// Stream executes a command on the remote server and streams all outputs into
// the local stdout and stderr streams.
func (client *sshClient) Stream(cmd string) error {
// Establish a single command session
session, err := client.client.NewSession()
if err != nil {
return err
}
defer session.Close()

session.Stdout = os.Stdout
session.Stderr = os.Stderr

// Execute the command and return any output
client.logger.Trace("Streaming command on remote server", "cmd", cmd)
return session.Run(cmd)
}

// Upload copied the set of files to a remote server via SCP, creating any non-
// existing folder in te mean time.
func (client *sshClient) Upload(files map[string][]byte) ([]byte, error) {
```

```
// Establish a single command session
session, err := client.client.NewSession()
if err != nil {
return nil, err
}
defer session.Close()

// Create a goroutine that streams the SCP content
go func() {
out, _ := session.StdinPipe()
defer out.Close()

for file, content := range files {
client.logger.Trace("Uploading file to server", "file", file, "bytes", len(content))

fmt.Fprintln(out, "D0755", 0, filepath.Dir(file))           // Ensure the folder exists
fmt.Fprintln(out, "C0644", len(content), filepath.Base(file)) // Create the actual file
out.Write(content)                                         // Stream the data content
fmt.Fprint(out, "\x00")                                    // Transfer end with \x00
fmt.Fprintln(out, "E")                                     // Leave directory (simpler)
}
}()
return session.CombinedOutput("/usr/bin/scp -v -tr ./")
}

81:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bufio"
"encoding/json"
"fmt"
"io/ioutil"
"math/big"
"os"
"path/filepath"
"sort"
"strconv"
"strings"
"syscall"
```

```
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/log"
"golang.org/x/crypto/ssh/terminal"
)

// config contains all the configurations needed by puppeth that should be saved
// between sessions.
type config struct {
path      string        // File containing the configuration values
genesis   *core.Genesis // Genesis block to cache for node deploys
bootFull  []string      // Bootnodes to always connect to by full nodes
bootLight []string      // Bootnodes to always connect to by light nodes
ethstats  string        // Ethstats settings to cache for node deploys

Servers map[string][]byte `json:"servers,omitempty"`
}

// servers retrieves an alphabetically sorted list of servers.
func (c config) servers() []string {
servers := make([]string, 0, len(c.Servers))
for server := range c.Servers {
servers = append(servers, server)
}
sort.Strings(servers)

return servers
}

// flush dumps the contents of config to disk.
func (c config) flush() {
os.MkdirAll(filepath.Dir(c.path), 0755)

out, _ := json.MarshalIndent(c, "", "  ")
if err := ioutil.WriteFile(c.path, out, 0644); err != nil {
log.Warn("Failed to save puppeth configs", "file", c.path, "err", err)
}
}

type wizard struct {
network string // Network name to manage
```

```
conf    config // Configurations from previous runs

servers  map[string]*sshClient // SSH connections to servers to administer
services map[string][]string   // Ethereum services known to be running on servers

in *bufio.Reader // Wrapper around stdin to allow reading user input
}

// read reads a single line from stdin, trimming if from spaces.
func (w *wizard) read() string {
fmt.Printf("> ")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
return strings.TrimSpace(text)
}

// readString reads a single line from stdin, trimming if from spaces, enforcing
// non-emptyness.
func (w *wizard) readString() string {
for {
fmt.Printf("> ")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
if text = strings.TrimSpace(text); text != "" {
return text
}
}
}

// readDefaultString reads a single line from stdin, trimming if from spaces. If
// an empty line is entered, the default value is returned.
func (w *wizard) readDefaultString(def string) string {
for {
fmt.Printf("> ")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
```

```
if text = strings.TrimSpace(text); text != "" {
return text
}
return def
}
}

// readInt reads a single line from stdin, trimming if from spaces, enforcing it
// to parse into an integer.
func (w *wizard) readInt() int {
for {
fmt.Printf("> ")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
if text = strings.TrimSpace(text); text == "" {
continue
}
val, err := strconv.Atoi(strings.TrimSpace(text))
if err != nil {
log.Error("Invalid input, expected integer", "err", err)
continue
}
return val
}
}

// readDefaultInt reads a single line from stdin, trimming if from spaces, enforcing
// it to parse into an integer. If an empty line is entered, the default value is
// returned.
func (w *wizard) readDefaultInt(def int) int {
for {
fmt.Printf("> ")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
if text = strings.TrimSpace(text); text == "" {
return def
}
val, err := strconv.Atoi(strings.TrimSpace(text))
```

```
if err != nil {
log.Error("Invalid input, expected integer", "err", err)
continue
}
return val
}
}

// readFloat reads a single line from stdin, trimming if from spaces, enforcing it
// to parse into a float.
func (w *wizard) readFloat() float64 {
for {
fmt.Printf("> ")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
if text = strings.TrimSpace(text); text == "" {
continue
}
val, err := strconv.ParseFloat(strings.TrimSpace(text), 64)
if err != nil {
log.Error("Invalid input, expected float", "err", err)
continue
}
return val
}
}

// readDefaultFloat reads a single line from stdin, trimming if from spaces, enforcing
// it to parse into a float. If an empty line is entered, the default value is returned.
func (w *wizard) readDefaultFloat(def float64) float64 {
for {
fmt.Printf("> ")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
if text = strings.TrimSpace(text); text == "" {
return def
}
val, err := strconv.ParseFloat(strings.TrimSpace(text), 64)
```

```
if err != nil {
log.Error("Invalid input, expected float", "err", err)
continue
}
return val
}
}

// readPassword reads a single line from stdin, trimming it from the trailing new
// line and returns it. The input will not be echoed.
func (w *wizard) readPassword() string {
for {
fmt.Printf("> ")
text, err := terminal.ReadPassword(int(syscall.Stdin))
if err != nil {
log.Crit("Failed to read password", "err", err)
}
fmt.Println()
return string(text)
}
}

// readAddress reads a single line from stdin, trimming if from spaces and converts
// it to an Ethereum address.
func (w *wizard) readAddress() *common.Address {
for {
// Read the address from the user
fmt.Printf("> 0x")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
if text = strings.TrimSpace(text); text == "" {
return nil
}
// Make sure it looks ok and return it if so
if len(text) != 40 {
log.Error("Invalid address length, please retry")
continue
}
bigaddr, _ := new(big.Int).SetString(text, 16)
address := common.BigToAddress(bigaddr)
```

```
return &address
}
}

// readDefaultAddress reads a single line from stdin, trimming if from spaces and
// converts it to an Ethereum address. If an empty line is entered, the default
// value is returned.
func (w *wizard) readDefaultAddress(def common.Address) common.Address {
for {
// Read the address from the user
fmt.Printf("> 0x")
text, err := w.in.ReadString('\n')
if err != nil {
log.Crit("Failed to read user input", "err", err)
}
if text = strings.TrimSpace(text); text == "" {
return def
}
// Make sure it looks ok and return it if so
if len(text) != 40 {
log.Error("Invalid address length, please retry")
continue
}
bigaddr, _ := new(big.Int).SetString(text, 16)
return common.BigToAddress(bigaddr)
}
}

// readJSON reads a raw JSON message and returns it.
func (w *wizard) readJSON() string {
var blob json.RawMessage

for {
fmt.Printf("> ")
if err := json.NewDecoder(w.in).Decode(&blob); err != nil {
log.Error("Invalid JSON, please try again", "err", err)
continue
}
return string(blob)
}
}
```

```
82:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_dashboard.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"fmt"

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

// deployDashboard queries the user for various input on deploying a web-service
// dashboard, after which is pushes the container.
func (w *wizard) deployDashboard() {
// Select the server to interact with
server := w.selectServer()
if server == "" {
return
}
client := w.servers[server]

// Retrieve any active dashboard configurations from the server
infos, err := checkDashboard(client, w.network)
if err != nil {
infos = &dashboardInfos{
port: 80,
host: client.server,
}
}
// Figure out which port to listen on
fmt.Println()
fmt.Printf("Which port should the dashboard listen on? (default = %d)\n", infos.port)
infos.port = w.readDefaultInt(infos.port)

// Figure which virtual-host to deploy the dashboard on
infos.host, err = w.ensureVirtualHost(client, infos.port, infos.host)
if err != nil {
log.Error("Failed to decide on dashboard host", "err", err)
return
}
// Port and proxy settings retrieved, figure out which services are available
available := make(map[string][]string)
```

```
for server, services := range w.services {
for _, service := range services {
available[service] = append(available[service], server)
}
}
listing := make(map[string]string)
for _, service := range []string{"ethstats", "explorer", "wallet", "faucet"} {
// Gather all the locally hosted pages of this type
var pages []string
for _, server := range available[service] {
client := w.servers[server]
if client == nil {
continue
}
// If there's a service running on the machine, retrieve it's port number
var port int
switch service {
case "ethstats":
if infos, err := checkEthstats(client, w.network); err == nil {
port = infos.port
}
case "faucet":
if infos, err := checkFaucet(client, w.network); err == nil {
port = infos.port
}
}
if page, err := resolve(client, w.network, service, port); err == nil && page != "" {
pages = append(pages, page)
}
}
// Promt the user to chose one, enter manually or simply not list this service
defLabel, defChoice := "don't list", len(pages)+2
if len(pages) > 0 {
defLabel, defChoice = pages[0], 1
}
fmt.Println()
fmt.Printf("Which %s service to list? (default = %s)\n", service, defLabel)
for i, page := range pages {
fmt.Printf(" %d. %s\n", i+1, page)
}
fmt.Printf(" %d. List external %s service\n", len(pages)+1, service)
fmt.Printf(" %d. Don't list any %s service\n", len(pages)+2, service)
```

```
choice := w.readDefaultInt(defChoice)
if choice < 0 || choice > len(pages)+2 {
log.Error("Invalid listing choice, aborting")
return
}
switch {
case choice <= len(pages):
listing[service] = pages[choice-1]
case choice == len(pages)+1:
fmt.Println()
fmt.Printf("Which address is the external %s service at?\n", service)
listing[service] = w.readString()
default:
// No service hosting for this
}
}
// If we have ethstats running, ask whether to make the secret public or not
var ethstats bool
if w.conf.ethstats != "" {
fmt.Println()
fmt.Println("Include ethstats secret on dashboard (y/n)? (default = yes)")
ethstats = w.readDefaultString("y") == "y"
}
// Try to deploy the dashboard container on the host
if out, err := deployDashboard(client, w.network, infos.port, infos.host, listing, &w.conf, ethstats);
err != nil {
log.Error("Failed to deploy dashboard container", "err", err)
if len(out) > 0 {
fmt.Printf("%s\n", out)
}
return
}
// All ok, run a network scan to pick any changes up
w.networkStats(false)
}
```

83:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_ethstats.go

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

package main
```

```
import (
"fmt"

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

// deployEthstats queries the user for various input on deploying an ethstats
// monitoring server, after which it executes it.
func (w *wizard) deployEthstats() {
// Select the server to interact with
server := w.selectServer()
if server == "" {
return
}
client := w.servers[server]

// Retrieve any active ethstats configurations from the server
infos, err := checkEthstats(client, w.network)
if err != nil {
infos = &ethstatsInfos{
port:   80,
host:   client.server,
secret: "",
}
}
// Figure out which port to listen on
fmt.Println()
fmt.Printf("Which port should ethstats listen on? (default = %d)\n", infos.port)
infos.port = w.readDefaultInt(infos.port)

// Figure which virtual-host to deploy ethstats on
if infos.host, err = w.ensureVirtualHost(client, infos.port, infos.host); err != nil {
log.Error("Failed to decide on ethstats host", "err", err)
return
}
// Port and proxy settings retrieved, figure out the secret and boot ethstats
fmt.Println()
if infos.secret == "" {
fmt.Printf("What should be the secret password for the API? (must not be empty)\n")
infos.secret = w.readString()
} else {
fmt.Printf("What should be the secret password for the API? (default = %s)\n", infos.secret)
```

```
infos.secret = w.readDefaultString(infos.secret)
}
// Try to deploy the ethstats server on the host
trusted := make([]string, 0, len(w.servers))
for _, client := range w.servers {
if client != nil {
trusted = append(trusted, client.address)
}
}
if out, err := deployEthstats(client, w.network, infos.port, infos.secret, infos.host, trusted); err != nil {
log.Error("Failed to deploy ethstats container", "err", err)
if len(out) > 0 {
fmt.Printf("%s\n", out)
}
return
}
// All ok, run a network scan to pick any changes up
w.networkStats(false)
}
```

84:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_faucet.go

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

package main

import (
"encoding/json"
"fmt"
"net/http"

"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/log"
)

// deployFaucet queries the user for various input on deploying a faucet, after
// which it executes it.
func (w *wizard) deployFaucet() {
// Select the server to interact with
server := w.selectServer()
if server == "" {
return
}
```

```
client := w.servers[server]

// Retrieve any active faucet configurations from the server
infos, err := checkFaucet(client, w.network)
if err != nil {
infos = &faucetInfos{
node:    &nodeInfos{portFull: 30303, peersTotal: 25},
port:    80,
host:    client.server,
amount:  1,
minutes: 1440,
tiers:   3,
}
}
infos.node.genesis, _ = json.MarshalIndent(w.conf.genesis, "", "  ")
infos.node.network = w.conf.genesis.Config.ChainId.Int64()

// Figure out which port to listen on
fmt.Println()
fmt.Printf("Which port should the faucet listen on? (default = %d)\n", infos.port)
infos.port = w.readDefaultInt(infos.port)

// Figure which virtual-host to deploy ethstats on
if infos.host, err = w.ensureVirtualHost(client, infos.port, infos.host); err != nil {
log.Error("Failed to decide on faucet host", "err", err)
return
}
// Port and proxy settings retrieved, figure out the funcing amount per perdion configurations
fmt.Println()
fmt.Printf("How many Ethers to release per request? (default = %d)\n", infos.amount)
infos.amount = w.readDefaultInt(infos.amount)

fmt.Println()
fmt.Printf("How many minutes to enforce between requests? (default = %d)\n", infos.minutes)
infos.minutes = w.readDefaultInt(infos.minutes)

fmt.Println()
fmt.Printf("How many funding tiers to feature (x2.5 amounts, x3 timeout)? (default = %d)\n",
infos.tiers)
infos.tiers = w.readDefaultInt(infos.tiers)
if infos.tiers == 0 {
log.Error("At least one funding tier must be set")
```

```
return
}
// Accessing GitHub gists requires API authorization, retrieve it
if infos.githubUser != "" {
fmt.Println()
fmt.Printf("Reuse previous (%s) GitHub API authorization (y/n)? (default = yes)\n",
infos.githubUser)
if w.readDefaultString("y") != "y" {
infos.githubUser, infos.githubToken = "", ""
}
}
if infos.githubUser == "" {
// No previous authorization (or new one requested)
fmt.Println()
fmt.Println("Which GitHub user to verify Gists through?")
infos.githubUser = w.readString()

fmt.Println()
fmt.Println("What is the GitHub personal access token of the user? (won't be echoed)")
infos.githubToken = w.readPassword()

// Do a sanity check query against github to ensure it's valid
req, _ := http.NewRequest("GET", "https://api.github.com/user", nil)
req.SetBasicAuth(infos.githubUser, infos.githubToken)
res, err := http.DefaultClient.Do(req)
if err != nil {
log.Error("Failed to verify GitHub authentication", "err", err)
return
}
defer res.Body.Close()

var msg struct {
Login   string `json:"login"`
Message string `json:"message"`
}
if err = json.NewDecoder(res.Body).Decode(&msg); err != nil {
log.Error("Failed to decode authorization response", "err", err)
return
}
if msg.Login != infos.githubUser {
log.Error("GitHub authorization failed", "user", infos.githubUser, "message", msg.Message)
return
```

```
}
}
// Accessing the reCaptcha service requires API authorizations, request it
if infos.captchaToken != "" {
fmt.Println()
fmt.Println("Reuse previous reCaptcha API authorization (y/n)? (default = yes)")
if w.readDefaultString("y") != "y" {
infos.captchaToken, infos.captchaSecret = "", ""
}
}
if infos.captchaToken == "" {
// No previous authorization (or old one discarded)
fmt.Println()
fmt.Println("Enable reCaptcha protection against robots (y/n)? (default = no)")
if w.readDefaultString("n") == "y" {
// Captcha protection explicitly requested, read the site and secret keys
fmt.Println()
fmt.Printf("What is the reCaptcha site key to authenticate human users?\n")
infos.captchaToken = w.readString()

fmt.Println()
fmt.Printf("What is the reCaptcha secret key to verify authentications? (won't be echoed)\n")
infos.captchaSecret = w.readPassword()
}
}
// Figure out where the user wants to store the persistent data
fmt.Println()
if infos.node.datadir == "" {
fmt.Printf("Where should data be stored on the remote machine?\n")
infos.node.datadir = w.readString()
} else {
fmt.Printf("Where should data be stored on the remote machine? (default = %s)\n",
infos.node.datadir)
infos.node.datadir = w.readDefaultString(infos.node.datadir)
}
// Figure out which port to listen on
fmt.Println()
fmt.Printf("Which TCP/UDP port should the light client listen on? (default = %d)\n",
infos.node.portFull)
infos.node.portFull = w.readDefaultInt(infos.node.portFull)

// Set a proper name to report on the stats page
```

```
fmt.Println()
if infos.node.ethstats == "" {
fmt.Printf("What should the node be called on the stats page?\n")
infos.node.ethstats = w.readString() + ":" + w.conf.ethstats
} else {
fmt.Printf("What should the node be called on the stats page? (default = %s)\n",
infos.node.ethstats)
infos.node.ethstats = w.readDefaultString(infos.node.ethstats) + ":" + w.conf.ethstats
}
// Load up the credential needed to release funds
if infos.node.keyJSON != "" {
if key, err := keystore.DecryptKey([]byte(infos.node.keyJSON), infos.node.keyPass); err != nil {
infos.node.keyJSON, infos.node.keyPass = "", ""
} else {
fmt.Println()
fmt.Printf("Reuse previous (%s) funding account (y/n)? (default = yes)\n", key.Address.Hex())
if w.readDefaultString("y") != "y" {
infos.node.keyJSON, infos.node.keyPass = "", ""
}
}
}
if infos.node.keyJSON == "" {
fmt.Println()
fmt.Println("Please paste the faucet's funding account key JSON:")
infos.node.keyJSON = w.readJSON()

fmt.Println()
fmt.Println("What's the unlock password for the account? (won't be echoed)")
infos.node.keyPass = w.readPassword()

if _, err := keystore.DecryptKey([]byte(infos.node.keyJSON), infos.node.keyPass); err != nil {
log.Error("Failed to decrypt key with given passphrase")
return
}
}
// Try to deploy the faucet server on the host
if out, err := deployFaucet(client, w.network, w.conf.bootLight, infos); err != nil {
log.Error("Failed to deploy faucet container", "err", err)
if len(out) > 0 {
fmt.Printf("%s\n", out)
}
return
```

```
}
// All ok, run a network scan to pick any changes up
w.networkStats(false)
}

85:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_genesis.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bytes"
"fmt"
"math/big"
"math/rand"
"time"

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

// makeGenesis creates a new genesis struct based on some user input.
func (w *wizard) makeGenesis() {
// Construct a default genesis block
genesis := &core.Genesis{
Timestamp:  uint64(time.Now().Unix()),
GasLimit:   4700000,
Difficulty: big.NewInt(1048576),
Alloc:      make(core.GenesisAlloc),
Config: &params.ChainConfig{
HomesteadBlock: big.NewInt(1),
EIP150Block:    big.NewInt(2),
EIP155Block:    big.NewInt(3),
EIP158Block:    big.NewInt(3),
},
}
// Figure out which consensus engine to choose
fmt.Println()
fmt.Println("Which consensus engine to use? (default = clique)")
fmt.Println(" 1. Ethash - proof-of-work")
```

```
fmt.Println(" 2. Clique - proof-of-authority")

choice := w.read()
switch {
case choice == "1":
// In case of ethash, we're pretty much done
genesis.Config.Ethash = new(params.EthashConfig)
genesis.ExtraData = make([]byte, 32)

case choice == "" || choice == "2":
// In the case of clique, configure the consensus parameters
genesis.Difficulty = big.NewInt(1)
genesis.Config.Clique = &params.CliqueConfig{
Period: 15,
Epoch:  30000,
}
fmt.Println()
fmt.Println("How many seconds should blocks take? (default = 15)")
genesis.Config.Clique.Period = uint64(w.readDefaultInt(15))

// We also need the initial list of signers
fmt.Println()
fmt.Println("Which accounts are allowed to seal? (mandatory at least one)")

var signers []common.Address
for {
if address := w.readAddress(); address != nil {
signers = append(signers, *address)
continue
}
if len(signers) > 0 {
break
}
}
// Sort the signers and embed into the extra-data section
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]
}
}
}
```

```
genesis.ExtraData = make([]byte, 32+len(signers)*common.AddressLength+65)
for i, signer := range signers {
copy(genesis.ExtraData[32+i*common.AddressLength:], signer[:])
}

default:
log.Crit("Invalid consensus engine choice", "choice", choice)
}
// Consensus all set, just ask for initial funds and go
fmt.Println()
fmt.Println("Which accounts should be pre-funded? (advisable at least one)")
for {
// Read the address of the account to fund
if address := w.readAddress(); address != nil {
genesis.Alloc[*address] = core.GenesisAccount{
Balance: new(big.Int).Lsh(big.NewInt(1), 256-7), // 2^256 / 128 (allow many pre-funds without
balance overflows)
}
continue
}
break
}
// Add a batch of precompile balances to avoid them getting deleted
for i := int64(0); i < 256; i++ {
genesis.Alloc[common.BigToAddress(big.NewInt(i))] = core.GenesisAccount{Balance:
big.NewInt(1)}
}
fmt.Println()

// Query the user for some custom extras
fmt.Println()
fmt.Println("Specify your chain/network ID if you want an explicit one (default = random)")
genesis.Config.ChainId = new(big.Int).SetUint64(uint64(w.readDefaultInt(rand.Intn(65536))))

fmt.Println()
fmt.Println("Anything fun to embed into the genesis block? (max 32 bytes)")

extra := w.read()
if len(extra) > 32 {
extra = extra[:32]
}
genesis.ExtraData = append([]byte(extra), genesis.ExtraData[len(extra):]...)
```

```
// All done, store the genesis and flush to disk
w.conf.genesis = genesis
}

86:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_intro.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"bufio"
"encoding/json"
"fmt"
"io/ioutil"
"os"
"path/filepath"
"strings"

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

// makeWizard creates and returns a new puppeth wizard.
func makeWizard(network string) *wizard {
return &wizard{
network: network,
conf: config{
Servers: make(map[string][]byte),
},
servers:  make(map[string]*sshClient),
services: make(map[string][]string),
in:       bufio.NewReader(os.Stdin),
}
}

// run displays some useful infos to the user, starting on the journey of
// setting up a new or managing an existing Ethereum private network.
func (w *wizard) run() {
fmt.Println("+------------------------------------------------------------+")
fmt.Println("| Welcome to puppeth, your Ethereum private network manager |")
fmt.Println("|                                                            |")
fmt.Println("| This tool lets you create a new Ethereum network down to  |")
```

```
fmt.Println("| the genesis block, bootnodes, miners and ethstats servers |")
fmt.Println("| without the hassle that it would normally entail.         |")
fmt.Println("|                                                           |")
fmt.Println("| Puppeth uses SSH to dial in to remote servers, and builds |")
fmt.Println("| its network components out of Docker containers using the |")
fmt.Println("| docker-compose toolset.                                   |")
fmt.Println("+-----------------------------------------------------------+")
fmt.Println()

// Make sure we have a good network name to work withfmt.Println()
if w.network == "" {
fmt.Println("Please specify a network name to administer (no spaces, please)")
for {
w.network = w.readString()
if !strings.Contains(w.network, " ") {
fmt.Printf("Sweet, you can set this via --network=%s next time!\n\n", w.network)
break
}
log.Error("I also like to live dangerously, still no spaces")
}
}
log.Info("Administering Ethereum network", "name", w.network)

// Load initial configurations and connect to all live servers
w.conf.path = filepath.Join(os.Getenv("HOME"), ".puppeth", w.network)

blob, err := ioutil.ReadFile(w.conf.path)
if err != nil {
log.Warn("No previous configurations found", "path", w.conf.path)
} else if err := json.Unmarshal(blob, &w.conf); err != nil {
log.Crit("Previous configuration corrupted", "path", w.conf.path, "err", err)
} else {
for server, pubkey := range w.conf.Servers {
log.Info("Dialing previously configured server", "server", server)
client, err := dial(server, pubkey)
if err != nil {
log.Error("Previous server unreachable", "server", server, "err", err)
}
w.servers[server] = client
}
w.networkStats(false)
}
```

```
// Basics done, loop ad infinitum about what to do
for {
fmt.Println()
fmt.Println("What would you like to do? (default = stats)")
fmt.Println(" 1. Show network stats")
if w.conf.genesis == nil {
fmt.Println(" 2. Configure new genesis")
} else {
fmt.Println(" 2. Save existing genesis")
}
if len(w.servers) == 0 {
fmt.Println(" 3. Track new remote server")
} else {
fmt.Println(" 3. Manage tracked machines")
}
if len(w.services) == 0 {
fmt.Println(" 4. Deploy network components")
} else {
fmt.Println(" 4. Manage network components")
}
//fmt.Println(" 5. ProTips for common usecases")

choice := w.read()
switch {
case choice == "" || choice == "1":
w.networkStats(false)

case choice == "2":
// If we don't have a genesis, make one
if w.conf.genesis == nil {
w.makeGenesis()
} else {
// Otherwise just save whatever we currently have
fmt.Println()
fmt.Printf("Which file to save the genesis into? (default = %s.json)\n", w.network)
out, _ := json.MarshalIndent(w.conf.genesis, "", "  ")
if err := ioutil.WriteFile(w.readDefaultString(fmt.Sprintf("%s.json", w.network)), out, 0644); err != nil
{
log.Error("Failed to save genesis file", "err", err)
}
log.Info("Exported existing genesis block")
}
```

```
case choice == "3":
if len(w.servers) == 0 {
if w.makeServer() != "" {
w.networkStats(false)
}
} else {
w.manageServers()
}
case choice == "4":
if len(w.services) == 0 {
w.deployComponent()
} else {
w.manageComponents()
}

case choice == "5":
w.networkStats(true)

default:
log.Error("That's not something I can do")
}
}
}

87:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_netstats.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"encoding/json"
"fmt"
"os"
"strings"

"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/log"
"github.com/olekukonko/tablewriter"
)

// networkStats verifies the status of network components and generates a protip
// configuration set to give users hints on how to do various tasks.
```

```
func (w *wizard) networkStats(tips bool) {
if len(w.servers) == 0 {
log.Error("No remote machines to gather stats from")
return
}
protips := new(protips)

// Iterate over all the specified hosts and check their status
stats := tablewriter.NewWriter(os.Stdout)
stats.SetHeader([]string{"Server", "IP", "Status", "Service", "Details"})
stats.SetColWidth(100)

for server, pubkey := range w.conf.Servers {
client := w.servers[server]
logger := log.New("server", server)
logger.Info("Starting remote server health-check")

// If the server is not connected, try to connect again
if client == nil {
conn, err := dial(server, pubkey)
if err != nil {
logger.Error("Failed to establish remote connection", "err", err)
stats.Append([]string{server, "", err.Error(), "", ""})
continue
}
client = conn
}
// Client connected one way or another, run health-checks
services := make(map[string]string)
logger.Debug("Checking for nginx availability")
if infos, err := checkNginx(client, w.network); err != nil {
if err != ErrServiceUnknown {
services["nginx"] = err.Error()
}
} else {
services["nginx"] = infos.String()
}
logger.Debug("Checking for ethstats availability")
if infos, err := checkEthstats(client, w.network); err != nil {
if err != ErrServiceUnknown {
services["ethstats"] = err.Error()
}
```

```
} else {
services["ethstats"] = infos.String()
protips.ethstats = infos.config
}
logger.Debug("Checking for bootnode availability")
if infos, err := checkNode(client, w.network, true); err != nil {
if err != ErrServiceUnknown {
services["bootnode"] = err.Error()
}
} else {
services["bootnode"] = infos.String()

protips.genesis = string(infos.genesis)
protips.bootFull = append(protips.bootFull, infos.enodeFull)
if infos.enodeLight != "" {
protips.bootLight = append(protips.bootLight, infos.enodeLight)
}
}
logger.Debug("Checking for sealnode availability")
if infos, err := checkNode(client, w.network, false); err != nil {
if err != ErrServiceUnknown {
services["sealnode"] = err.Error()
}
} else {
services["sealnode"] = infos.String()
protips.genesis = string(infos.genesis)
}
logger.Debug("Checking for faucet availability")
if infos, err := checkFaucet(client, w.network); err != nil {
if err != ErrServiceUnknown {
services["faucet"] = err.Error()
}
} else {
services["faucet"] = infos.String()
}
logger.Debug("Checking for dashboard availability")
if infos, err := checkDashboard(client, w.network); err != nil {
if err != ErrServiceUnknown {
services["dashboard"] = err.Error()
}
} else {
services["dashboard"] = infos.String()
```

```
}
// All status checks complete, report and check next server
delete(w.services, server)
for service := range services {
w.services[server] = append(w.services[server], service)
}
server, address := client.server, client.address
for service, status := range services {
stats.Append([]string{server, address, "online", service, status})
server, address = "", ""
}
if len(services) == 0 {
stats.Append([]string{server, address, "online", "", ""})
}
}
// If a genesis block was found, load it into our configs
if protips.genesis != "" {
genesis := new(core.Genesis)
if err := json.Unmarshal([]byte(protips.genesis), genesis); err != nil {
log.Error("Failed to parse remote genesis", "err", err)
} else {
w.conf.genesis = genesis
protips.network = genesis.Config.ChainId.Int64()
}
}
if protips.ethstats != "" {
w.conf.ethstats = protips.ethstats
}
w.conf.bootFull = protips.bootFull
w.conf.bootLight = protips.bootLight

// Print any collected stats and return
if !tips {
stats.Render()
} else {
protips.print(w.network)
}
}

// protips contains a collection of network infos to report pro-tips
// based on.
type protips struct {
```

```
genesis   string
network   int64
bootFull  []string
bootLight []string
ethstats  string
}

// print analyzes the network information available and prints a collection of
// pro tips for the user's consideration.
func (p *protips) print(network string) {
// If a known genesis block is available, display it and prepend an init command
fullinit, lightinit := "", ""
if p.genesis != "" {
fullinit = fmt.Sprintf("geth --datadir=$HOME/.%s init %s.json && ", network, network)
lightinit = fmt.Sprintf("geth --datadir=$HOME/.%s --light init %s.json && ", network, network)
}
// If an ethstats server is available, add the ethstats flag
statsflag := ""
if p.ethstats != "" {
if strings.Contains(p.ethstats, " ") {
statsflag = fmt.Sprintf(` --ethstats="yournode:%s"`, p.ethstats)
} else {
statsflag = fmt.Sprintf(` --ethstats=yournode:%s`, p.ethstats)
}
}
// If bootnodes have been specified, add the bootnode flag
bootflagFull := ""
if len(p.bootFull) > 0 {
bootflagFull = fmt.Sprintf(` --bootnodes %s`, strings.Join(p.bootFull, ","))
}
bootflagLight := ""
if len(p.bootLight) > 0 {
bootflagLight = fmt.Sprintf(` --bootnodes %s`, strings.Join(p.bootLight, ","))
}
// Assemble all the known pro-tips
var tasks, tips []string

tasks = append(tasks, "Run an archive node with historical data")
tips = append(tips, fmt.Sprintf("%sgeth --networkid=%d --datadir=$HOME/.%s --
cache=1024%s%s", fullinit, p.network, network, statsflag, bootflagFull))

tasks = append(tasks, "Run a full node with recent data only")
```

```
tips = append(tips, fmt.Sprintf("%sgeth --networkid=%d --datadir=$HOME/.%s --cache=512 --fast%s%s", fullinit, p.network, network, statsflag, bootflagFull))

tasks = append(tasks, "Run a light node with on demand retrievals")
tips = append(tips, fmt.Sprintf("%sgeth --networkid=%d --datadir=$HOME/.%s --light%s%s", lightinit, p.network, network, statsflag, bootflagLight))

tasks = append(tasks, "Run an embedded node with constrained memory")
tips = append(tips, fmt.Sprintf("%sgeth --networkid=%d --datadir=$HOME/.%s --cache=32 --light%s%s", lightinit, p.network, network, statsflag, bootflagLight))

// If the tips are short, display in a table
short := true
for _, tip := range tips {
if len(tip) > 100 {
short = false
break
}
}
fmt.Println()
if short {
howto := tablewriter.NewWriter(os.Stdout)
howto.SetHeader([]string{"Fun tasks for you", "Tips on how to"})
howto.SetColWidth(100)

for i := 0; i < len(tasks); i++ {
howto.Append([]string{tasks[i], tips[i]})
}
howto.Render()
return
}
// Meh, tips got ugly, split into many lines
for i := 0; i < len(tasks); i++ {
fmt.Println(tasks[i])
fmt.Println(strings.Repeat("-", len(tasks[i])))
fmt.Println(tips[i])
fmt.Println()
fmt.Println()
}
}
```

88:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_network.go

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

package main

import (
"fmt"
"strings"

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

// manageServers displays a list of servers the user can disconnect from, and an
// option to connect to new servers.
func (w *wizard) manageServers() {
// List all the servers we can disconnect, along with an entry to connect a new one
fmt.Println()

servers := w.conf.servers()
for i, server := range servers {
fmt.Printf(" %d. Disconnect %s\n", i+1, server)
}
fmt.Printf(" %d. Connect another server\n", len(w.conf.Servers)+1)

choice := w.readInt()
if choice < 0 || choice > len(w.conf.Servers)+1 {
log.Error("Invalid server choice, aborting")
return
}
// If the user selected an existing server, drop it
if choice <= len(w.conf.Servers) {
server := servers[choice-1]
client := w.servers[server]

delete(w.servers, server)
if client != nil {
client.Close()
}
delete(w.conf.Servers, server)
w.conf.flush()

log.Info("Disconnected existing server", "server", server)
w.networkStats(false)
```

```
return
}
// If the user requested connecting a new server, do it
if w.makeServer() != "" {
w.networkStats(false)
}
}

// makeServer reads a single line from stdin and interprets it as a hostname to
// connect to. It tries to establish a new SSH session and also executing some
// baseline validations.
//
// If connection succeeds, the server is added to the wizards configs!
func (w *wizard) makeServer() string {
fmt.Println()
fmt.Println("Please enter remote server's address:")

for {
// Read and fial the server to ensure docker is present
input := w.readString()

client, err := dial(input, nil)
if err != nil {
log.Error("Server not ready for puppeth", "err", err)
return ""
}
// All checks passed, start tracking the server
w.servers[input] = client
w.conf.Servers[input] = client.pubkey
w.conf.flush()

return input
}
}

// selectServer lists the user all the currnetly known servers to choose from,
// also granting the option to add a new one.
func (w *wizard) selectServer() string {
// List the available server to the user and wait for a choice
fmt.Println()
fmt.Println("Which server do you want to interact with?")
```

```
servers := w.conf.servers()
for i, server := range servers {
fmt.Printf(" %d. %s\n", i+1, server)
}
fmt.Printf(" %d. Connect another server\n", len(w.conf.Servers)+1)

choice := w.readInt()
if choice < 0 || choice > len(w.conf.Servers)+1 {
log.Error("Invalid server choice, aborting")
return ""
}
// If the user requested connecting to a new server, go for it
if choice <= len(w.conf.Servers) {
return servers[choice-1]
}
return w.makeServer()
}

// manageComponents displays a list of network components the user can tear down
// and an option
func (w *wizard) manageComponents() {
// List all the componens we can tear down, along with an entry to deploy a new one
fmt.Println()

var serviceHosts, serviceNames []string
for server, services := range w.services {
for _, service := range services {
serviceHosts = append(serviceHosts, server)
serviceNames = append(serviceNames, service)

fmt.Printf(" %d. Tear down %s on %s\n", len(serviceHosts), strings.Title(service), server)
}
}
fmt.Printf(" %d. Deploy new network component\n", len(serviceHosts)+1)

choice := w.readInt()
if choice < 0 || choice > len(serviceHosts)+1 {
log.Error("Invalid component choice, aborting")
return
}
// If the user selected an existing service, destroy it
if choice <= len(serviceHosts) {
```

```
// Figure out the service to destroy and execute it
service := serviceNames[choice-1]
server := serviceHosts[choice-1]
client := w.servers[server]

if out, err := tearDown(client, w.network, service, true); err != nil {
log.Error("Failed to tear down component", "err", err)
if len(out) > 0 {
fmt.Printf("%s\n", out)
}
return
}
// Clean up any references to it from out state
services := w.services[server]
for i, name := range services {
if name == service {
w.services[server] = append(services[:i], services[i+1:]...)
if len(w.services[server]) == 0 {
delete(w.services, server)
}
}
}
log.Info("Torn down existing component", "server", server, "service", service)
return
}
// If the user requested deploying a new component, do it
w.deployComponent()
}

// deployComponent displays a list of network components the user can deploy and
// guides through the process.
func (w *wizard) deployComponent() {
// Print all the things we can deploy and wait or user choice
fmt.Println()
fmt.Println("What would you like to deploy? (recommended order)")
fmt.Println(" 1. Ethstats  - Network monitoring tool")
fmt.Println(" 2. Bootnode  - Entry point of the network")
fmt.Println(" 3. Sealer    - Full node minting new blocks")
fmt.Println(" 4. Wallet    - Browser wallet for quick sends (todo)")
fmt.Println(" 5. Faucet    - Crypto faucet to give away funds")
fmt.Println(" 6. Dashboard - Website listing above web-services")
```

```
switch w.read() {
case "1":
w.deployEthstats()
case "2":
w.deployNode(true)
case "3":
w.deployNode(false)
case "4":
case "5":
w.deployFaucet()
case "6":
w.deployDashboard()
default:
log.Error("That's not something I can do")
}
}
```

89:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_nginx.go

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

package main

import (
"fmt"

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

// ensureVirtualHost checks whether a reverse-proxy is running on the specified
// host machine, and if yes requests a virtual host from the user to host a
// specific web service on. If no proxy exists, the method will offer to deploy
// one.
//
// If the user elects not to use a reverse proxy, an empty hostname is returned!
func (w *wizard) ensureVirtualHost(client *sshClient, port int, def string) (string, error) {
if proxy, _ := checkNginx(client, w.network); proxy != nil {
// Reverse proxy is running, if ports match, we need a virtual host
if proxy.port == port {
fmt.Println()
fmt.Printf("Shared port, which domain to assign? (default = %s)\n", def)
return w.readDefaultString(def), nil
}
```

```
}
// Reverse proxy is not running, offer to deploy a new one
fmt.Println()
fmt.Println("Allow sharing the port with other services (y/n)? (default = yes)")
if w.readDefaultString("y") == "y" {
if out, err := deployNginx(client, w.network, port); err != nil {
log.Error("Failed to deploy reverse-proxy", "err", err)
if len(out) > 0 {
fmt.Printf("%s\n", out)
}
return "", err
}
// Reverse proxy deployed, ask again for the virtual-host
fmt.Println()
fmt.Printf("Proxy deployed, which domain to assign? (default = %s)\n", def)
return w.readDefaultString(def), nil
}
// Reverse proxy not requested, deploy as a standalone service
return "", nil
}

90:F:\git\coin\ethereum\go-ethereum\cmd\puppeth\wizard_node.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"encoding/json"
"fmt"
"time"

"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/log"
)

// deployNode creates a new node configuration based on some user input.
func (w *wizard) deployNode(boot bool) {
// Do some sanity check before the user wastes time on input
if w.conf.genesis == nil {
log.Error("No genesis block configured")
return
```

```
}
if w.conf.ethstats == "" {
log.Error("No ethstats server configured")
return
}
// Select the server to interact with
server := w.selectServer()
if server == "" {
return
}
client := w.servers[server]

// Retrieve any active ethstats configurations from the server
infos, err := checkNode(client, w.network, boot)
if err != nil {
if boot {
infos = &nodeInfos{portFull: 30303, peersTotal: 512, peersLight: 256}
} else {
infos = &nodeInfos{portFull: 30303, peersTotal: 50, peersLight: 0, gasTarget: 4.7, gasPrice: 18}
}
}
infos.genesis, _ = json.MarshalIndent(w.conf.genesis, "", "  ")
infos.network = w.conf.genesis.Config.ChainId.Int64()

// Figure out where the user wants to store the persistent data
fmt.Println()
if infos.datadir == "" {
fmt.Printf("Where should data be stored on the remote machine?\n")
infos.datadir = w.readString()
} else {
fmt.Printf("Where should data be stored on the remote machine? (default = %s)\n", infos.datadir)
infos.datadir = w.readDefaultString(infos.datadir)
}
// Figure out which port to listen on
fmt.Println()
fmt.Printf("Which TCP/UDP port to listen on? (default = %d)\n", infos.portFull)
infos.portFull = w.readDefaultInt(infos.portFull)

// Figure out how many peers to allow (different based on node type)
fmt.Println()
fmt.Printf("How many peers to allow connecting? (default = %d)\n", infos.peersTotal)
infos.peersTotal = w.readDefaultInt(infos.peersTotal)
```

```
// Figure out how many light peers to allow (different based on node type)
fmt.Println()
fmt.Printf("How many light peers to allow connecting? (default = %d)\n", infos.peersLight)
infos.peersLight = w.readDefaultInt(infos.peersLight)

// Set a proper name to report on the stats page
fmt.Println()
if infos.ethstats == "" {
fmt.Printf("What should the node be called on the stats page?\n")
infos.ethstats = w.readString() + ":" + w.conf.ethstats
} else {
fmt.Printf("What should the node be called on the stats page? (default = %s)\n", infos.ethstats)
infos.ethstats = w.readDefaultString(infos.ethstats) + ":" + w.conf.ethstats
}
// If the node is a miner/signer, load up needed credentials
if !boot {
if w.conf.genesis.Config.Ethash != nil {
// Ethash based miners only need an etherbase to mine against
fmt.Println()
if infos.etherbase == "" {
fmt.Printf("What address should the miner user?\n")
for {
if address := w.readAddress(); address != nil {
infos.etherbase = address.Hex()
break
}
}
} else {
fmt.Printf("What address should the miner user? (default = %s)\n", infos.etherbase)
infos.etherbase = w.readDefaultAddress(common.HexToAddress(infos.etherbase)).Hex()
}
} else if w.conf.genesis.Config.Clique != nil {
// If a previous signer was already set, offer to reuse it
if infos.keyJSON != "" {
if key, err := keystore.DecryptKey([]byte(infos.keyJSON), infos.keyPass); err != nil {
infos.keyJSON, infos.keyPass = "", ""
} else {
fmt.Println()
fmt.Printf("Reuse previous (%s) signing account (y/n)? (default = yes)\n", key.Address.Hex())
if w.readDefaultString("y") != "y" {
infos.keyJSON, infos.keyPass = "", ""
```

```
}
}
}
// Clique based signers need a keyfile and unlock password, ask if unavailable
if infos.keyJSON == "" {
fmt.Println()
fmt.Println("Please paste the signer's key JSON:")
infos.keyJSON = w.readJSON()

fmt.Println()
fmt.Println("What's the unlock password for the account? (won't be echoed)")
infos.keyPass = w.readPassword()

if _, err := keystore.DecryptKey([]byte(infos.keyJSON), infos.keyPass); err != nil {
log.Error("Failed to decrypt key with given passphrase")
return
}
}
}
// Establish the gas dynamics to be enforced by the signer
fmt.Println()
fmt.Printf("What gas limit should empty blocks target (MGas)? (default = %0.3f)\n",
infos.gasTarget)
infos.gasTarget = w.readDefaultFloat(infos.gasTarget)

fmt.Println()
fmt.Printf("What gas price should the signer require (GWei)? (default = %0.3f)\n", infos.gasPrice)
infos.gasPrice = w.readDefaultFloat(infos.gasPrice)
}
// Try to deploy the full node on the host
if out, err := deployNode(client, w.network, w.conf.bootFull, w.conf.bootLight, infos); err != nil {
log.Error("Failed to deploy Ethereum node container", "err", err)
if len(out) > 0 {
fmt.Printf("%s\n", out)
}
return
}
// All ok, run a network scan to pick any changes up
log.Info("Waiting for node to finish booting")
time.Sleep(3 * time.Second)

w.networkStats(false)
```

```
}

91:F:\git\coin\ethereum\go-ethereum\cmd\rlpdump\main.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// rlpdump is a pretty-printer for RLP data.
package main

import (
"bytes"
"encoding/hex"
"flag"
"fmt"
"io"
"os"
"strings"

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

var (
hexMode = flag.String("hex", "", "dump given hex data")
noASCII = flag.Bool("noascii", false, "don't print ASCII strings readably")
single  = flag.Bool("single", false, "print only the first element, discard the rest")
)

func init() {
flag.Usage = func() {
fmt.Fprintln(os.Stderr, "Usage:", os.Args[0], "[-noascii] [-hex <data>] [filename]")
flag.PrintDefaults()
fmt.Fprintln(os.Stderr, `
Dumps RLP data from the given file in readable form.
If the filename is omitted, data is read from stdin.`)
}
}

func main() {
flag.Parse()

var r io.Reader
switch {
case *hexMode != "":
```

```
data, err := hex.DecodeString(*hexMode)
if err != nil {
die(err)
}
r = bytes.NewReader(data)

case flag.NArg() == 0:
r = os.Stdin

case flag.NArg() == 1:
fd, err := os.Open(flag.Arg(0))
if err != nil {
die(err)
}
defer fd.Close()
r = fd

default:
fmt.Fprintln(os.Stderr, "Error: too many arguments")
flag.Usage()
os.Exit(2)
}

s := rlp.NewStream(r, 0)
for {
if err := dump(s, 0); err != nil {
if err != io.EOF {
die(err)
}
break
}
fmt.Println()
if *single {
break
}
}
}

func dump(s *rlp.Stream, depth int) error {
kind, size, err := s.Kind()
if err != nil {
return err
```

```
}
switch kind {
case rlp.Byte, rlp.String:
str, err := s.Bytes()
if err != nil {
return err
}
if len(str) == 0 || !*noASCII && isASCII(str) {
fmt.Printf("%s%q", ws(depth), str)
} else {
fmt.Printf("%s%x", ws(depth), str)
}
case rlp.List:
s.List()
defer s.ListEnd()
if size == 0 {
fmt.Print(ws(depth) + "[]")
} else {
fmt.Println(ws(depth) + "[")
for i := 0; ; i++ {
if i > 0 {
fmt.Print(",\n")
}
if err := dump(s, depth+1); err == rlp.EOL {
break
} else if err != nil {
return err
}
}
fmt.Print(ws(depth) + "]")
}
}
return nil
}

func isASCII(b []byte) bool {
for _, c := range b {
if c < 32 || c > 126 {
return false
}
}
return true
```

```
}

func ws(n int) string {
return strings.Repeat("  ", n)
}

func die(args ...interface{}) {
fmt.Fprintln(os.Stderr, args...)
os.Exit(1)
}

92:F:\git\coin\ethereum\go-ethereum\cmd\swarm\cleandb.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/swarm/storage"
"gopkg.in/urfave/cli.v1"
)

func cleandb(ctx *cli.Context) {
args := ctx.Args()
if len(args) != 1 {
utils.Fatalf("Need path to chunks database as the first and only argument")
}

chunkDbPath := args[0]
hash := storage.MakeHashFunc("SHA3")
dbStore, err := storage.NewDbStore(chunkDbPath, hash, 10000000, 0)
if err != nil {
utils.Fatalf("Cannot initialise dbstore: %v", err)
}
dbStore.Cleanup()
}

93:F:\git\coin\ethereum\go-ethereum\cmd\swarm\hash.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// Command bzzhash computes a swarm tree hash.
package main
```

```
import (
"fmt"
"os"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/swarm/storage"
"gopkg.in/urfave/cli.v1"
)

func hash(ctx *cli.Context) {
args := ctx.Args()
if len(args) < 1 {
utils.Fatalf("Usage: swarm hash <file name>")
}
f, err := os.Open(args[0])
if err != nil {
utils.Fatalf("Error opening file " + args[1])
}
defer f.Close()

stat, _ := f.Stat()
chunker := storage.NewTreeChunker(storage.NewChunkerParams())
key, err := chunker.Split(f, stat.Size(), nil, nil, nil)
if err != nil {
utils.Fatalf("%v\n", err)
} else {
fmt.Printf("%v\n", key)
}
}

94:F:\git\coin\ethereum\go-ethereum\cmd\swarm\list.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"fmt"
"os"
"strings"
"text/tabwriter"
```

```
"github.com/ethereum/go-ethereum/cmd/utils"
swarm "github.com/ethereum/go-ethereum/swarm/api/client"
"gopkg.in/urfave/cli.v1"
)

func list(ctx *cli.Context) {
args := ctx.Args()

if len(args) < 1 {
utils.Fatalf("Please supply a manifest reference as the first argument")
} else if len(args) > 2 {
utils.Fatalf("Too many arguments - usage 'swarm ls manifest [prefix]'")
}
manifest := args[0]

var prefix string
if len(args) == 2 {
prefix = args[1]
}

bzzapi := strings.TrimRight(ctx.GlobalString(SwarmApiFlag.Name), "/")
client := swarm.NewClient(bzzapi)
list, err := client.List(manifest, prefix)
if err != nil {
utils.Fatalf("Failed to generate file and directory list: %s", err)
}

w := tabwriter.NewWriter(os.Stdout, 1, 2, 2, ' ', 0)
defer w.Flush()
fmt.Fprintln(w, "HASH\tCONTENT TYPE\tPATH")
for _, prefix := range list.CommonPrefixes {
fmt.Fprintf(w, "%s\t%s\t%s\n", "", "DIR", prefix)
}
for _, entry := range list.Entries {
fmt.Fprintf(w, "%s\t%s\t%s\n", entry.Hash, entry.ContentType, entry.Path)
}
}

95:F:\git\coin\ethereum\go-ethereum\cmd\swarm\main.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main
```

```
import (
"context"
"crypto/ecdsa"
"fmt"
"io/ioutil"
"math/big"
"os"
"os/signal"
"runtime"
"strconv"
"strings"
"syscall"
"time"

"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/console"
"github.com/ethereum/go-ethereum/contracts/ens"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/ethclient"
"github.com/ethereum/go-ethereum/internal/debug"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/p2p"
"github.com/ethereum/go-ethereum/p2p/discover"
"github.com/ethereum/go-ethereum/params"
"github.com/ethereum/go-ethereum/rpc"
"github.com/ethereum/go-ethereum/swarm"
bzzapi "github.com/ethereum/go-ethereum/swarm/api"
"gopkg.in/urfave/cli.v1"
)

const clientIdentifier = "swarm"

var (
gitCommit       string // Git SHA1 commit hash of the release (set via linker flags)
testbetBootNodes = []string{
"enode://ec8ae764f7cb0417bdfb009b9d0f18ab3818a3a4e8e7c67dd5f18971a93510a2e6f43cd0b6
9a27e439a9629457ea804104f37c85e41eed057d3faabbf7744cdf@13.74.157.139:30429",
```

"enode://c2e1fceb3bf3be19dff71eec6cccf19f2dbf7567ee017d130240c670be8594bc9163353ca55dd8df7a4f161dd94b36d0615c17418b5a3cdcbb4e9d99dfa4de37@13.74.157.139:30430",
"enode://fe29b82319b734ce1ec68b84657d57145fee237387e63273989d354486731e59f78858e452ef800a020559da22dcca759536e6aa5517c53930d29ce0b1029286@13.74.157.139:30431",
"enode://1d7187e7bde45cf0bee489ce9852dd6d1a0d9aa67a33a6b8e6db8a4fbc6fcfa6f0f1a5419343671521b863b187d1c73bad3603bae66421d157ffef357669ddb8@13.74.157.139:30432",
"enode://0e4cba800f7b1ee73673afa6a4acead4018f0149d2e3216be3f133318fd165b324cd71b81fbe1e80deac8dbf56e57a49db7be67f8b9bc81bd2b7ee496434fb5d@13.74.157.139:30433",
}
)

var (
ChequebookAddrFlag = cli.StringFlag{
Name: "chequebook",
Usage: "chequebook contract address",
}
SwarmAccountFlag = cli.StringFlag{
Name: "bzzaccount",
Usage: "Swarm account key file",
}
SwarmListenAddrFlag = cli.StringFlag{
Name: "httpaddr",
Usage: "Swarm HTTP API listening interface",
}
SwarmPortFlag = cli.StringFlag{
Name: "bzzport",
Usage: "Swarm local http api port",
}
SwarmNetworkIdFlag = cli.IntFlag{
Name: "bzznetworkid",
Usage: "Network identifier (integer, default 3=swarm testnet)",
}
SwarmConfigPathFlag = cli.StringFlag{
Name: "bzzconfig",
Usage: "Swarm config file path (datadir/bzz)",
}
SwarmSwapEnabledFlag = cli.BoolFlag{
Name: "swap",
Usage: "Swarm SWAP enabled (default false)",
}
SwarmSwapAPIFlag = cli.StringFlag{
Name: "swap-api",

```
Usage: "URL of the Ethereum API provider to use to settle SWAP payments",
}
SwarmSyncEnabledFlag = cli.BoolTFlag{
Name:  "sync",
Usage: "Swarm Syncing enabled (default true)",
}
EnsAPIFlag = cli.StringFlag{
Name:  "ens-api",
Usage: "URL of the Ethereum API provider to use for ENS record lookups",
Value: node.DefaultIPCEndpoint("geth"),
}
EnsAddrFlag = cli.StringFlag{
Name:  "ens-addr",
Usage: "ENS contract address (default is detected as testnet or mainnet using --ens-api)",
}
SwarmApiFlag = cli.StringFlag{
Name:  "bzzapi",
Usage: "Swarm HTTP endpoint",
Value: "http://127.0.0.1:8500",
}
SwarmRecursiveUploadFlag = cli.BoolFlag{
Name:  "recursive",
Usage: "Upload directories recursively",
}
SwarmWantManifestFlag = cli.BoolTFlag{
Name:  "manifest",
Usage: "Automatic manifest upload",
}
SwarmUploadDefaultPath = cli.StringFlag{
Name:  "defaultpath",
Usage: "path to file served for empty url path (none)",
}
SwarmUpFromStdinFlag = cli.BoolFlag{
Name:  "stdin",
Usage: "reads data to be uploaded from stdin",
}
SwarmUploadMimeType = cli.StringFlag{
Name:  "mime",
Usage: "force mime type",
}
CorsStringFlag = cli.StringFlag{
Name:  "corsdomain",
```

```
Usage: "Domain on which to send Access-Control-Allow-Origin header (multiple domains can be supplied separated by a ',')",
}

// the following flags are deprecated and should be removed in the future
DeprecatedEthAPIFlag = cli.StringFlag{
Name:  "ethapi",
Usage: "DEPRECATED: please use --ens-api and --swap-api",
}
)

var defaultNodeConfig = node.DefaultConfig

// This init function sets defaults so cmd/swarm can run alongside geth.
func init() {
defaultNodeConfig.Name = clientIdentifier
defaultNodeConfig.Version = params.VersionWithCommit(gitCommit)
defaultNodeConfig.P2P.ListenAddr = ":30399"
defaultNodeConfig.IPCPath = "bzzd.ipc"
// Set flag defaults for --help display.
utils.ListenPortFlag.Value = 30399
}

var app = utils.NewApp(gitCommit, "Ethereum Swarm")

// This init function creates the cli.App.
func init() {
app.Action = bzzd
app.HideVersion = true // we have a command to print the version
app.Copyright = "Copyright 2013-2016 The go-ethereum Authors"
app.Commands = []cli.Command{
{
Action:    version,
Name:      "version",
Usage:     "Print version numbers",
ArgsUsage: " ",
Description: `
The output of this command is supposed to be machine-readable.
`,
},
{
Action:    upload,
```

```
Name:      "up",
Usage:     "upload a file or directory to swarm using the HTTP API",
ArgsUsage: " <file>",
Description: `
"upload a file or directory to swarm using the HTTP API and prints the root hash",
`,
},
{
Action:    list,
Name:      "ls",
Usage:     "list files and directories contained in a manifest",
ArgsUsage: " <manifest> [<prefix>]",
Description: `
Lists files and directories contained in a manifest.
`,
},
{
Action:    hash,
Name:      "hash",
Usage:     "print the swarm hash of a file or directory",
ArgsUsage: " <file>",
Description: `
Prints the swarm hash of file or directory.
`,
},
{
Name:      "manifest",
Usage:     "update a MANIFEST",
ArgsUsage: "manifest COMMAND",
Description: `
Updates a MANIFEST by adding/removing/updating the hash of a path.
`,
Subcommands: []cli.Command{
{
Action:    add,
Name:      "add",
Usage:     "add a new path to the manifest",
ArgsUsage: "<MANIFEST> <path> <hash> [<content-type>]",
Description: `
Adds a new path to the manifest
`,
},
```

```
{
Action:    update,
Name:      "update",
Usage:     "update the hash for an already existing path in the manifest",
ArgsUsage: "<MANIFEST> <path> <newhash> [<newcontent-type>]",
Description: `
Update the hash for an already existing path in the manifest
`,
},
{
Action:    remove,
Name:      "remove",
Usage:     "removes a path from the manifest",
ArgsUsage: "<MANIFEST> <path>",
Description: `
Removes a path from the manifest
`,
},
},
},
{
Action:    cleandb,
Name:      "cleandb",
Usage:     "Cleans database of corrupted entries",
ArgsUsage: " ",
Description: `
Cleans database of corrupted entries.
`,
},
}

app.Flags = []cli.Flag{
utils.IdentityFlag,
utils.DataDirFlag,
utils.BootnodesFlag,
utils.KeyStoreDirFlag,
utils.ListenPortFlag,
utils.NoDiscoverFlag,
utils.DiscoveryV5Flag,
utils.NetrestrictFlag,
utils.NodeKeyFileFlag,
utils.NodeKeyHexFlag,
```

```
utils.MaxPeersFlag,
utils.NATFlag,
utils.IPCDisabledFlag,
utils.IPCPathFlag,
utils.PasswordFileFlag,
// bzzd-specific flags
CorsStringFlag,
EnsAPIFlag,
EnsAddrFlag,
SwarmConfigPathFlag,
SwarmSwapEnabledFlag,
SwarmSwapAPIFlag,
SwarmSyncEnabledFlag,
SwarmListenAddrFlag,
SwarmPortFlag,
SwarmAccountFlag,
SwarmNetworkIdFlag,
ChequebookAddrFlag,
// upload flags
SwarmApiFlag,
SwarmRecursiveUploadFlag,
SwarmWantManifestFlag,
SwarmUploadDefaultPath,
SwarmUpFromStdinFlag,
SwarmUploadMimeType,
//deprecated flags
DeprecatedEthAPIFlag,
}
app.Flags = append(app.Flags, debug.Flags...)
app.Before = func(ctx *cli.Context) error {
runtime.GOMAXPROCS(runtime.NumCPU())
return debug.Setup(ctx)
}
app.After = func(ctx *cli.Context) error {
debug.Exit()
return nil
}
}

func main() {
if err := app.Run(os.Args); err != nil {
fmt.Fprintln(os.Stderr, err)
```

```
os.Exit(1)
}
}

func version(ctx *cli.Context) error {
fmt.Println(strings.Title(clientIdentifier))
fmt.Println("Version:", params.Version)
if gitCommit != "" {
fmt.Println("Git Commit:", gitCommit)
}
fmt.Println("Network Id:", ctx.GlobalInt(utils.NetworkIdFlag.Name))
fmt.Println("Go Version:", runtime.Version())
fmt.Println("OS:", runtime.GOOS)
fmt.Printf("GOPATH=%s\n", os.Getenv("GOPATH"))
fmt.Printf("GOROOT=%s\n", runtime.GOROOT())
return nil
}

func bzzd(ctx *cli.Context) error {
// exit if the deprecated --ethapi flag is set
if ctx.GlobalString(DeprecatedEthAPIFlag.Name) != "" {
utils.Fatalf("--ethapi is no longer a valid command line flag, please use --ens-api and/or --swap-api.")
}

cfg := defaultNodeConfig
utils.SetNodeConfig(ctx, &cfg)
stack, err := node.New(&cfg)
if err != nil {
utils.Fatalf("can't create node: %v", err)
}

registerBzzService(ctx, stack)
utils.StartNode(stack)

go func() {
sigc := make(chan os.Signal, 1)
signal.Notify(sigc, syscall.SIGTERM)
defer signal.Stop(sigc)
<-sigc
log.Info("Got sigterm, shutting swarm down...")
stack.Stop()
```

```
}()

networkId := ctx.GlobalUint64(SwarmNetworkIdFlag.Name)
// Add bootnodes as initial peers.
if ctx.GlobalIsSet(utils.BootnodesFlag.Name) {
bootnodes := strings.Split(ctx.GlobalString(utils.BootnodesFlag.Name), ",")
injectBootnodes(stack.Server(), bootnodes)
} else {
if networkId == 3 {
injectBootnodes(stack.Server(), testbetBootNodes)
}
}

stack.Wait()
return nil
}

// detectEnsAddr determines the ENS contract address by getting both the
// version and genesis hash using the client and matching them to either
// mainnet or testnet addresses
func detectEnsAddr(client *rpc.Client) (common.Address, error) {
ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
defer cancel()

var version string
if err := client.CallContext(ctx, &version, "net_version"); err != nil {
return common.Address{}, err
}

block, err := ethclient.NewClient(client).BlockByNumber(ctx, big.NewInt(0))
if err != nil {
return common.Address{}, err
}

switch {

case version == "1" && block.Hash() == params.MainnetGenesisHash:
log.Info("using Mainnet ENS contract address", "addr", ens.MainNetAddress)
return ens.MainNetAddress, nil

case version == "3" && block.Hash() == params.TestnetGenesisHash:
log.Info("using Testnet ENS contract address", "addr", ens.TestNetAddress)
```

```
return ens.TestNetAddress, nil

default:
return common.Address{}, fmt.Errorf("unknown version and genesis hash: %s %s", version,
block.Hash())
}
}

func registerBzzService(ctx *cli.Context, stack *node.Node) {
prvkey := getAccount(ctx, stack)

chbookaddr := common.HexToAddress(ctx.GlobalString(ChequebookAddrFlag.Name))
bzzdir := ctx.GlobalString(SwarmConfigPathFlag.Name)
if bzzdir == "" {
bzzdir = stack.InstanceDir()
}

bzzconfig, err := bzzapi.NewConfig(bzzdir, chbookaddr, prvkey,
ctx.GlobalUint64(SwarmNetworkIdFlag.Name))
if err != nil {
utils.Fatalf("unable to configure swarm: %v", err)
}
bzzport := ctx.GlobalString(SwarmPortFlag.Name)
if len(bzzport) > 0 {
bzzconfig.Port = bzzport
}
if bzzaddr := ctx.GlobalString(SwarmListenAddrFlag.Name); bzzaddr != "" {
bzzconfig.ListenAddr = bzzaddr
}
swapEnabled := ctx.GlobalBool(SwarmSwapEnabledFlag.Name)
syncEnabled := ctx.GlobalBoolT(SwarmSyncEnabledFlag.Name)

swapapi := ctx.GlobalString(SwarmSwapAPIFlag.Name)
if swapEnabled && swapapi == "" {
utils.Fatalf("SWAP is enabled but --swap-api is not set")
}

ensapi := ctx.GlobalString(EnsAPIFlag.Name)
ensAddr := ctx.GlobalString(EnsAddrFlag.Name)

cors := ctx.GlobalString(CorsStringFlag.Name)
```

```
boot := func(ctx *node.ServiceContext) (node.Service, error) {
var swapClient *ethclient.Client
if swapapi != "" {
log.Info("connecting to SWAP API", "url", swapapi)
swapClient, err = ethclient.Dial(swapapi)
if err != nil {
return nil, fmt.Errorf("error connecting to SWAP API %s: %s", swapapi, err)
}
}

var ensClient *ethclient.Client
if ensapi != "" {
log.Info("connecting to ENS API", "url", ensapi)
client, err := rpc.Dial(ensapi)
if err != nil {
return nil, fmt.Errorf("error connecting to ENS API %s: %s", ensapi, err)
}
ensClient = ethclient.NewClient(client)

if ensAddr != "" {
bzzconfig.EnsRoot = common.HexToAddress(ensAddr)
} else {
ensAddr, err := detectEnsAddr(client)
if err == nil {
bzzconfig.EnsRoot = ensAddr
} else {
log.Warn(fmt.Sprintf("could not determine ENS contract address, using default %s",
bzzconfig.EnsRoot), "err", err)
}
}
}

return swarm.NewSwarm(ctx, swapClient, ensClient, bzzconfig, swapEnabled, syncEnabled, cors)
}
if err := stack.Register(boot); err != nil {
utils.Fatalf("Failed to register the Swarm service: %v", err)
}
}

func getAccount(ctx *cli.Context, stack *node.Node) *ecdsa.PrivateKey {
keyid := ctx.GlobalString(SwarmAccountFlag.Name)
```

```
if keyid == "" {
utils.Fatalf("Option %q is required", SwarmAccountFlag.Name)
}
// Try to load the arg as a hex key file.
if key, err := crypto.LoadECDSA(keyid); err == nil {
log.Info("Swarm account key loaded", "address", crypto.PubkeyToAddress(key.PublicKey))
return key
}
// Otherwise try getting it from the keystore.
am := stack.AccountManager()
ks := am.Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore)

return decryptStoreAccount(ks, keyid, utils.MakePasswordList(ctx))
}

func decryptStoreAccount(ks *keystore.KeyStore, account string, passwords []string)
*ecdsa.PrivateKey {
var a accounts.Account
var err error
if common.IsHexAddress(account) {
a, err = ks.Find(accounts.Account{Address: common.HexToAddress(account)})
} else if ix, ixerr := strconv.Atoi(account); ixerr == nil && ix > 0 {
if accounts := ks.Accounts(); len(accounts) > ix {
a = accounts[ix]
} else {
err = fmt.Errorf("index %d higher than number of accounts %d", ix, len(accounts))
}
} else {
utils.Fatalf("Can't find swarm account key %s", account)
}
if err != nil {
utils.Fatalf("Can't find swarm account key: %v", err)
}
keyjson, err := ioutil.ReadFile(a.URL.Path)
if err != nil {
utils.Fatalf("Can't load swarm account key: %v", err)
}
for i := 0; i < 3; i++ {
password := getPassPhrase(fmt.Sprintf("Unlocking swarm account %s [%d/3]", a.Address.Hex(),
i+1), i, passwords)
key, err := keystore.DecryptKey(keyjson, password)
if err == nil {
```

```
return key.PrivateKey
}
}
utils.Fatalf("Can't decrypt swarm account key")
return nil
}

// getPassPhrase retrieves the password associated with bzz account, either by fetching
// from a list of pre-loaded passwords, or by requesting it interactively from user.
func getPassPhrase(prompt string, i int, passwords []string) string {
// non-interactive
if len(passwords) > 0 {
if i < len(passwords) {
return passwords[i]
}
return passwords[len(passwords)-1]
}

// fallback to interactive mode
if prompt != "" {
fmt.Println(prompt)
}
password, err := console.Stdin.PromptPassword("Passphrase: ")
if err != nil {
utils.Fatalf("Failed to read passphrase: %v", err)
}
return password
}

func injectBootnodes(srv *p2p.Server, nodes []string) {
for _, url := range nodes {
n, err := discover.ParseNode(url)
if err != nil {
log.Error("Invalid swarm bootnode", "err", err)
continue
}
srv.AddPeer(n)
}
}

96:F:\git\coin\ethereum\go-ethereum\cmd\swarm\manifest.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.
```

```
// Command  MANIFEST update
package main

import (
"encoding/json"
"fmt"
"mime"
"path/filepath"
"strings"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/swarm/api"
swarm "github.com/ethereum/go-ethereum/swarm/api/client"
"gopkg.in/urfave/cli.v1"
)

func add(ctx *cli.Context) {
args := ctx.Args()
if len(args) < 3 {
utils.Fatalf("Need atleast three arguments <MHASH> <path> <HASH> [<content-type>]")
}

var (
mhash = args[0]
path  = args[1]
hash  = args[2]

ctype        string
wantManifest = ctx.GlobalBoolT(SwarmWantManifestFlag.Name)
mroot        api.Manifest
)

if len(args) > 3 {
ctype = args[3]
} else {
ctype = mime.TypeByExtension(filepath.Ext(path))
}

newManifest := addEntryToManifest(ctx, mhash, path, hash, ctype)
fmt.Println(newManifest)
```

```
if !wantManifest {
// Print the manifest. This is the only output to stdout.
mrootJSON, _ := json.MarshalIndent(mroot, "", "  ")
fmt.Println(string(mrootJSON))
return
}
}

func update(ctx *cli.Context) {

args := ctx.Args()
if len(args) < 3 {
utils.Fatalf("Need atleast three arguments <MHASH> <path> <HASH>")
}

var (
mhash = args[0]
path  = args[1]
hash  = args[2]

ctype        string
wantManifest = ctx.GlobalBoolT(SwarmWantManifestFlag.Name)
mroot        api.Manifest
)
if len(args) > 3 {
ctype = args[3]
} else {
ctype = mime.TypeByExtension(filepath.Ext(path))
}

newManifest := updateEntryInManifest(ctx, mhash, path, hash, ctype)
fmt.Println(newManifest)

if !wantManifest {
// Print the manifest. This is the only output to stdout.
mrootJSON, _ := json.MarshalIndent(mroot, "", "  ")
fmt.Println(string(mrootJSON))
return
}
}

func remove(ctx *cli.Context) {
```

```
args := ctx.Args()
if len(args) < 2 {
utils.Fatalf("Need atleast two arguments <MHASH> <path>")
}

var (
mhash = args[0]
path  = args[1]

wantManifest = ctx.GlobalBoolT(SwarmWantManifestFlag.Name)
mroot        api.Manifest
)

newManifest := removeEntryFromManifest(ctx, mhash, path)
fmt.Println(newManifest)

if !wantManifest {
// Print the manifest. This is the only output to stdout.
mrootJSON, _ := json.MarshalIndent(mroot, "", "  ")
fmt.Println(string(mrootJSON))
return
}
}

func addEntryToManifest(ctx *cli.Context, mhash, path, hash, ctype string) string {

var (
bzzapi           = strings.TrimRight(ctx.GlobalString(SwarmApiFlag.Name), "/")
client           = swarm.NewClient(bzzapi)
longestPathEntry = api.ManifestEntry{}
)

mroot, err := client.DownloadManifest(mhash)
if err != nil {
utils.Fatalf("Manifest download failed: %v", err)
}

//TODO: check if the "hash" to add is valid and present in swarm
_, err = client.DownloadManifest(hash)
if err != nil {
utils.Fatalf("Hash to add is not present: %v", err)
}
```

```
// See if we path is in this Manifest or do we have to dig deeper
for _, entry := range mroot.Entries {
if path == entry.Path {
utils.Fatalf("Path %s already present, not adding anything", path)
} else {
if entry.ContentType == "application/bzz-manifest+json" {
prfxlen := strings.HasPrefix(path, entry.Path)
if prfxlen && len(path) > len(longestPathEntry.Path) {
longestPathEntry = entry
}
}
}
}

if longestPathEntry.Path != "" {
// Load the child Manifest add the entry there
newPath := path[len(longestPathEntry.Path):]
newHash := addEntryToManifest(ctx, longestPathEntry.Hash, newPath, hash, ctype)

// Replace the hash for parent Manifests
newMRoot := &api.Manifest{}
for _, entry := range mroot.Entries {
if longestPathEntry.Path == entry.Path {
entry.Hash = newHash
}
newMRoot.Entries = append(newMRoot.Entries, entry)
}
mroot = newMRoot
} else {
// Add the entry in the leaf Manifest
newEntry := api.ManifestEntry{
Hash:        hash,
Path:        path,
ContentType: ctype,
}
mroot.Entries = append(mroot.Entries, newEntry)
}

newManifestHash, err := client.UploadManifest(mroot)
if err != nil {
utils.Fatalf("Manifest upload failed: %v", err)
```

```
}
return newManifestHash

}

func updateEntryInManifest(ctx *cli.Context, mhash, path, hash, ctype string) string {

var (
bzzapi          = strings.TrimRight(ctx.GlobalString(SwarmApiFlag.Name), "/")
client          = swarm.NewClient(bzzapi)
newEntry        = api.ManifestEntry{}
longestPathEntry = api.ManifestEntry{}
)

mroot, err := client.DownloadManifest(mhash)
if err != nil {
utils.Fatalf("Manifest download failed: %v", err)
}

//TODO: check if the "hash" with which to update is valid and present in swarm

// See if we path is in this Manifest or do we have to dig deeper
for _, entry := range mroot.Entries {
if path == entry.Path {
newEntry = entry
} else {
if entry.ContentType == "application/bzz-manifest+json" {
prfxlen := strings.HasPrefix(path, entry.Path)
if prfxlen && len(path) > len(longestPathEntry.Path) {
longestPathEntry = entry
}
}
}
}

if longestPathEntry.Path == "" && newEntry.Path == "" {
utils.Fatalf("Path %s not present in the Manifest, not setting anything", path)
}

if longestPathEntry.Path != "" {
// Load the child Manifest add the entry there
newPath := path[len(longestPathEntry.Path):]
```

```
newHash := updateEntryInManifest(ctx, longestPathEntry.Hash, newPath, hash, ctype)

// Replace the hash for parent Manifests
newMRoot := &api.Manifest{}
for _, entry := range mroot.Entries {
if longestPathEntry.Path == entry.Path {
entry.Hash = newHash
}
newMRoot.Entries = append(newMRoot.Entries, entry)

}
mroot = newMRoot
}

if newEntry.Path != "" {
// Replace the hash for leaf Manifest
newMRoot := &api.Manifest{}
for _, entry := range mroot.Entries {
if newEntry.Path == entry.Path {
myEntry := api.ManifestEntry{
Hash:        hash,
Path:        entry.Path,
ContentType: ctype,
}
newMRoot.Entries = append(newMRoot.Entries, myEntry)
} else {
newMRoot.Entries = append(newMRoot.Entries, entry)
}
}
mroot = newMRoot
}

newManifestHash, err := client.UploadManifest(mroot)
if err != nil {
utils.Fatalf("Manifest upload failed: %v", err)
}
return newManifestHash
}

func removeEntryFromManifest(ctx *cli.Context, mhash, path string) string {

var (
```

```
bzzapi          = strings.TrimRight(ctx.GlobalString(SwarmApiFlag.Name), "/")
client          = swarm.NewClient(bzzapi)
entryToRemove   = api.ManifestEntry{}
longestPathEntry = api.ManifestEntry{}
)

mroot, err := client.DownloadManifest(mhash)
if err != nil {
utils.Fatalf("Manifest download failed: %v", err)
}

// See if we path is in this Manifest or do we have to dig deeper
for _, entry := range mroot.Entries {
if path == entry.Path {
entryToRemove = entry
} else {
if entry.ContentType == "application/bzz-manifest+json" {
prfxlen := strings.HasPrefix(path, entry.Path)
if prfxlen && len(path) > len(longestPathEntry.Path) {
longestPathEntry = entry
}
}
}
}

if longestPathEntry.Path == "" && entryToRemove.Path == "" {
utils.Fatalf("Path %s not present in the Manifest, not removing anything", path)
}

if longestPathEntry.Path != "" {
// Load the child Manifest remove the entry there
newPath := path[len(longestPathEntry.Path):]
newHash := removeEntryFromManifest(ctx, longestPathEntry.Hash, newPath)

// Replace the hash for parent Manifests
newMRoot := &api.Manifest{}
for _, entry := range mroot.Entries {
if longestPathEntry.Path == entry.Path {
entry.Hash = newHash
}
newMRoot.Entries = append(newMRoot.Entries, entry)
}
```

```
mroot = newMRoot
}

if entryToRemove.Path != "" {
// remove the entry in this Manifest
newMRoot := &api.Manifest{}
for _, entry := range mroot.Entries {
if entryToRemove.Path != entry.Path {
newMRoot.Entries = append(newMRoot.Entries, entry)
}
}
mroot = newMRoot
}

newManifestHash, err := client.UploadManifest(mroot)
if err != nil {
utils.Fatalf("Manifest upload failed: %v", err)
}
return newManifestHash
}

97:F:\git\coin\ethereum\go-ethereum\cmd\swarm\run_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main

import (
"fmt"
"io/ioutil"
"net"
"os"
"path/filepath"
"runtime"
"testing"
"time"

"github.com/docker/docker/pkg/reexec"
"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/internal/cmdtest"
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/p2p"
"github.com/ethereum/go-ethereum/rpc"
```

```
"github.com/ethereum/go-ethereum/swarm"
)

func init() {
// Run the app if we've been exec'd as "swarm-test" in runSwarm.
reexec.Register("swarm-test", func() {
if err := app.Run(os.Args); err != nil {
fmt.Fprintln(os.Stderr, err)
os.Exit(1)
}
os.Exit(0)
})
}

func TestMain(m *testing.M) {
// check if we have been reexec'd
if reexec.Init() {
return
}
os.Exit(m.Run())
}

func runSwarm(t *testing.T, args ...string) *cmdtest.TestCmd {
tt := cmdtest.NewTestCmd(t, nil)

// Boot "swarm". This actually runs the test binary but the TestMain
// function will prevent any tests from running.
tt.Run("swarm-test", args...)

return tt
}

type testCluster struct {
Nodes  []*testNode
TmpDir string
}

// newTestCluster starts a test swarm cluster of the given size.
//
// A temporary directory is created and each node gets a data directory inside
// it.
//
```

```
// Each node listens on 127.0.0.1 with random ports for both the HTTP and p2p
// ports (assigned by first listening on 127.0.0.1:0 and then passing the ports
// as flags).
//
// When starting more than one node, they are connected together using the
// admin SetPeer RPC method.
func newTestCluster(t *testing.T, size int) *testCluster {
cluster := &testCluster{}
defer func() {
if t.Failed() {
cluster.Shutdown()
}
}()

tmpdir, err := ioutil.TempDir("", "swarm-test")
if err != nil {
t.Fatal(err)
}
cluster.TmpDir = tmpdir

// start the nodes
cluster.Nodes = make([]*testNode, 0, size)
for i := 0; i < size; i++ {
dir := filepath.Join(cluster.TmpDir, fmt.Sprintf("swarm%02d", i))
if err := os.Mkdir(dir, 0700); err != nil {
t.Fatal(err)
}

node := newTestNode(t, dir)
node.Name = fmt.Sprintf("swarm%02d", i)

cluster.Nodes = append(cluster.Nodes, node)
}

if size == 1 {
return cluster
}

// connect the nodes together
for _, node := range cluster.Nodes {
if err := node.Client.Call(nil, "admin_addPeer", cluster.Nodes[0].Enode); err != nil {
t.Fatal(err)
```

```
}
}

// wait until all nodes have the correct number of peers
outer:
for _, node := range cluster.Nodes {
var peers []*p2p.PeerInfo
for start := time.Now(); time.Since(start) < time.Minute; time.Sleep(50 * time.Millisecond) {
if err := node.Client.Call(&peers, "admin_peers"); err != nil {
t.Fatal(err)
}
if len(peers) == len(cluster.Nodes)-1 {
continue outer
}
}
t.Fatalf("%s only has %d / %d peers", node.Name, len(peers), len(cluster.Nodes)-1)
}

return cluster
}

func (c *testCluster) Shutdown() {
for _, node := range c.Nodes {
node.Shutdown()
}
os.RemoveAll(c.TmpDir)
}

type testNode struct {
Name   string
Addr   string
URL    string
Enode  string
Dir    string
Client *rpc.Client
Cmd    *cmdtest.TestCmd
}

const testPassphrase = "swarm-test-passphrase"

func newTestNode(t *testing.T, dir string) *testNode {
// create key
```

```
conf := &node.Config{
DataDir: dir,
IPCPath: "bzzd.ipc",
}
n, err := node.New(conf)
if err != nil {
t.Fatal(err)
}
account, err :=
n.AccountManager().Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore).NewAccount(test
Passphrase)
if err != nil {
t.Fatal(err)
}

node := &testNode{Dir: dir}

// use a unique IPCPath when running tests on Windows
if runtime.GOOS == "windows" {
conf.IPCPath = fmt.Sprintf("bzzd-%s.ipc", account.Address.String())
}

// assign ports
httpPort, err := assignTCPPort()
if err != nil {
t.Fatal(err)
}
p2pPort, err := assignTCPPort()
if err != nil {
t.Fatal(err)
}

// start the node
node.Cmd = runSwarm(t,
"--port", p2pPort,
"--nodiscover",
"--datadir", dir,
"--ipcpath", conf.IPCPath,
"--ens-api", "",
"--bzzaccount", account.Address.String(),
"--bzznetworkid", "321",
"--bzzport", httpPort,
```

```
"--verbosity", "6",
)
node.Cmd.InputLine(testPassphrase)
defer func() {
if t.Failed() {
node.Shutdown()
}
}()

// wait for the node to start
for start := time.Now(); time.Since(start) < 10*time.Second; time.Sleep(50 * time.Millisecond) {
node.Client, err = rpc.Dial(conf.IPCEndpoint())
if err == nil {
break
}
}
if node.Client == nil {
t.Fatal(err)
}

// load info
var info swarm.Info
if err := node.Client.Call(&info, "bzz_info"); err != nil {
t.Fatal(err)
}
node.Addr = net.JoinHostPort("127.0.0.1", info.Port)
node.URL = "http://" + node.Addr

var nodeInfo p2p.NodeInfo
if err := node.Client.Call(&nodeInfo, "admin_nodeInfo"); err != nil {
t.Fatal(err)
}
node.Enode = fmt.Sprintf("enode://%s@127.0.0.1:%s", nodeInfo.ID, p2pPort)

return node
}

func (n *testNode) Shutdown() {
if n.Cmd != nil {
n.Cmd.Kill()
}
}
```

```
func assignTCPPort() (string, error) {
l, err := net.Listen("tcp", "127.0.0.1:0")
if err != nil {
return "", err
}
l.Close()
_, port, err := net.SplitHostPort(l.Addr().String())
if err != nil {
return "", err
}
return port, nil
}

98:F:\git\coin\ethereum\go-ethereum\cmd\swarm\upload.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// Command bzzup uploads files to the swarm HTTP API.
package main

import (
"errors"
"fmt"
"io"
"io/ioutil"
"mime"
"net/http"
"os"
"os/user"
"path"
"path/filepath"
"strings"

"github.com/ethereum/go-ethereum/cmd/utils"
swarm "github.com/ethereum/go-ethereum/swarm/api/client"
"gopkg.in/urfave/cli.v1"
)

func upload(ctx *cli.Context) {

args := ctx.Args()
var (
```

```
bzzapi       = strings.TrimRight(ctx.GlobalString(SwarmApiFlag.Name), "/")
recursive    = ctx.GlobalBool(SwarmRecursiveUploadFlag.Name)
wantManifest = ctx.GlobalBoolT(SwarmWantManifestFlag.Name)
defaultPath  = ctx.GlobalString(SwarmUploadDefaultPath.Name)
fromStdin    = ctx.GlobalBool(SwarmUpFromStdinFlag.Name)
mimeType     = ctx.GlobalString(SwarmUploadMimeType.Name)
client       = swarm.NewClient(bzzapi)
file         string
)

if len(args) != 1 {
if fromStdin {
tmp, err := ioutil.TempFile("", "swarm-stdin")
if err != nil {
utils.Fatalf("error create tempfile: %s", err)
}
defer os.Remove(tmp.Name())
n, err := io.Copy(tmp, os.Stdin)
if err != nil {
utils.Fatalf("error copying stdin to tempfile: %s", err)
} else if n == 0 {
utils.Fatalf("error reading from stdin: zero length")
}
file = tmp.Name()
} else {
utils.Fatalf("Need filename as the first and only argument")
}
} else {
file = expandPath(args[0])
}

if !wantManifest {
f, err := swarm.Open(file)
if err != nil {
utils.Fatalf("Error opening file: %s", err)
}
defer f.Close()
hash, err := client.UploadRaw(f, f.Size)
if err != nil {
utils.Fatalf("Upload failed: %s", err)
}
fmt.Println(hash)
```

```
return
}

stat, err := os.Stat(file)
if err != nil {
utils.Fatalf("Error opening file: %s", err)
}

// define a function which either uploads a directory or single file
// based on the type of the file being uploaded
var doUpload func() (hash string, err error)
if stat.IsDir() {
doUpload = func() (string, error) {
if !recursive {
return "", errors.New("Argument is a directory and recursive upload is disabled")
}
return client.UploadDirectory(file, defaultPath, "")
}
} else {
doUpload = func() (string, error) {
f, err := swarm.Open(file)
if err != nil {
return "", fmt.Errorf("error opening file: %s", err)
}
defer f.Close()
if mimeType == "" {
mimeType = detectMimeType(file)
}
f.ContentType = mimeType
return client.Upload(f, "")
}
}
hash, err := doUpload()
if err != nil {
utils.Fatalf("Upload failed: %s", err)
}
fmt.Println(hash)
}

// Expands a file path
// 1. replace tilde with users home dir
// 2. expands embedded environment variables
```

```
// 3. cleans the path, e.g. /a/b/../c -> /a/c
// Note, it has limitations, e.g. ~someuser/tmp will not be expanded
func expandPath(p string) string {
if strings.HasPrefix(p, "~/") || strings.HasPrefix(p, "~\\") {
if home := homeDir(); home != "" {
p = home + p[1:]
}
}
return path.Clean(os.ExpandEnv(p))
}

func homeDir() string {
if home := os.Getenv("HOME"); home != "" {
return home
}
if usr, err := user.Current(); err == nil {
return usr.HomeDir
}
return ""
}

func detectMimeType(file string) string {
if ext := filepath.Ext(file); ext != "" {
return mime.TypeByExtension(ext)
}
f, err := os.Open(file)
if err != nil {
return ""
}
defer f.Close()
buf := make([]byte, 512)
if n, _ := f.Read(buf); n > 0 {
return http.DetectContentType(buf)
}
return ""
}

99:F:\git\coin\ethereum\go-ethereum\cmd\swarm\upload_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package main
```

```
import (
"io"
"io/ioutil"
"net/http"
"os"
"testing"
)

// TestCLISwarmUp tests that running 'swarm up' makes the resulting file
// available from all nodes via the HTTP API
func TestCLISwarmUp(t *testing.T) {
t.Skip("flaky test")

// start 3 node cluster
t.Log("starting 3 node cluster")
cluster := newTestCluster(t, 3)
defer cluster.Shutdown()

// create a tmp file
tmp, err := ioutil.TempFile("", "swarm-test")
assertNil(t, err)
defer tmp.Close()
defer os.Remove(tmp.Name())
_, err = io.WriteString(tmp, "data")
assertNil(t, err)

// upload the file with 'swarm up' and expect a hash
t.Log("uploading file with 'swarm up'")
up := runSwarm(t, "--bzzapi", cluster.Nodes[0].URL, "up", tmp.Name())
_, matches := up.ExpectRegexp(`[a-f\d]{64}`)
up.ExpectExit()
hash := matches[0]
t.Logf("file uploaded with hash %s", hash)

// get the file from the HTTP API of each node
for _, node := range cluster.Nodes {
t.Logf("getting file from %s", node.Name)
res, err := http.Get(node.URL + "/bzz:/" + hash)
assertNil(t, err)
assertHTTPResponse(t, res, http.StatusOK, "data")
}
}
```

```
func assertNil(t *testing.T, err error) {
if err != nil {
t.Fatal(err)
}
}

func assertHTTPResponse(t *testing.T, res *http.Response, expectedStatus int, expectedBody
string) {
defer res.Body.Close()
if res.StatusCode != expectedStatus {
t.Fatalf("expected HTTP status %d, got %s", expectedStatus, res.Status)
}
data, err := ioutil.ReadAll(res.Body)
assertNil(t, err)
if string(data) != expectedBody {
t.Fatalf("expected HTTP body %q, got %q", expectedBody, data)
}
}

100:F:\git\coin\ethereum\go-ethereum\cmd\utils\cmd.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// Package utils contains internal helper functions for go-ethereum commands.
package utils

import (
"compress/gzip"
"fmt"
"io"
"os"
"os/signal"
"runtime"
"strings"

"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/core/types"
"github.com/ethereum/go-ethereum/internal/debug"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/rlp"
)
```

```
const (
importBatchSize = 2500
)

// Fatalf formats a message to standard error and exits the program.
// The message is also printed to standard output if standard error
// is redirected to a different file.
func Fatalf(format string, args ...interface{}) {
w := io.MultiWriter(os.Stdout, os.Stderr)
if runtime.GOOS == "windows" {
// The SameFile check below doesn't work on Windows.
// stdout is unlikely to get redirected though, so just print there.
w = os.Stdout
} else {
outf, _ := os.Stdout.Stat()
errf, _ := os.Stderr.Stat()
if outf != nil && errf != nil && os.SameFile(outf, errf) {
w = os.Stderr
}
}
fmt.Fprintf(w, "Fatal: "+format+"\n", args...)
os.Exit(1)
}

func StartNode(stack *node.Node) {
if err := stack.Start(); err != nil {
Fatalf("Error starting protocol stack: %v", err)
}
go func() {
sigc := make(chan os.Signal, 1)
signal.Notify(sigc, os.Interrupt)
defer signal.Stop(sigc)
<-sigc
log.Info("Got interrupt, shutting down...")
go stack.Stop()
for i := 10; i > 0; i-- {
<-sigc
if i > 1 {
log.Warn("Already shutting down, interrupt more to panic.", "times", i-1)
}
}
```

```
debug.Exit() // ensure trace and CPU profile data is flushed.
debug.LoudPanic("boom")
}()
}

func ImportChain(chain *core.BlockChain, fn string) error {
// Watch for Ctrl-C while the import is running.
// If a signal is received, the import will stop at the next batch.
interrupt := make(chan os.Signal, 1)
stop := make(chan struct{})
signal.Notify(interrupt, os.Interrupt)
defer signal.Stop(interrupt)
defer close(interrupt)
go func() {
if _, ok := <-interrupt; ok {
log.Info("Interrupted during import, stopping at next batch")
}
close(stop)
}()
checkInterrupt := func() bool {
select {
case <-stop:
return true
default:
return false
}
}

log.Info("Importing blockchain", "file", fn)
fh, err := os.Open(fn)
if err != nil {
return err
}
defer fh.Close()

var reader io.Reader = fh
if strings.HasSuffix(fn, ".gz") {
if reader, err = gzip.NewReader(reader); err != nil {
return err
}
}
```

```
stream := rlp.NewStream(reader, 0)

// Run actual the import.
blocks := make(types.Blocks, importBatchSize)
n := 0
for batch := 0; ; batch++ {
// Load a batch of RLP blocks.
if checkInterrupt() {
return fmt.Errorf("interrupted")
}
i := 0
for ; i < importBatchSize; i++ {
var b types.Block
if err := stream.Decode(&b); err == io.EOF {
break
} else if err != nil {
return fmt.Errorf("at block %d: %v", n, err)
}
// don't import first block
if b.NumberU64() == 0 {
i--
continue
}
blocks[i] = &b
n++
}
if i == 0 {
break
}
// Import the batch.
if checkInterrupt() {
return fmt.Errorf("interrupted")
}
if hasAllBlocks(chain, blocks[:i]) {
log.Info("Skipping batch as all blocks present", "batch", batch, "first", blocks[0].Hash(), "last",
blocks[i-1].Hash())
continue
}

if _, err := chain.InsertChain(blocks[:i]); err != nil {
return fmt.Errorf("invalid block %d: %v", n, err)
}
```

```
}
return nil
}

func hasAllBlocks(chain *core.BlockChain, bs []*types.Block) bool {
for _, b := range bs {
if !chain.HasBlock(b.Hash()) {
return false
}
}
return true
}

func ExportChain(blockchain *core.BlockChain, fn string) error {
log.Info("Exporting blockchain", "file", fn)
fh, err := os.OpenFile(fn, os.O_CREATE|os.O_WRONLY|os.O_TRUNC, os.ModePerm)
if err != nil {
return err
}
defer fh.Close()

var writer io.Writer = fh
if strings.HasSuffix(fn, ".gz") {
writer = gzip.NewWriter(writer)
defer writer.(*gzip.Writer).Close()
}

if err := blockchain.Export(writer); err != nil {
return err
}
log.Info("Exported blockchain", "file", fn)

return nil
}

func ExportAppendChain(blockchain *core.BlockChain, fn string, first uint64, last uint64) error {
log.Info("Exporting blockchain", "file", fn)
// TODO verify mode perms
fh, err := os.OpenFile(fn, os.O_CREATE|os.O_APPEND|os.O_WRONLY, os.ModePerm)
if err != nil {
return err
}
```

```
defer fh.Close()

var writer io.Writer = fh
if strings.HasSuffix(fn, ".gz") {
writer = gzip.NewWriter(writer)
defer writer.(*gzip.Writer).Close()
}

if err := blockchain.ExportN(writer, first, last); err != nil {
return err
}
log.Info("Exported blockchain to", "file", fn)
return nil
}

101:F:\git\coin\ethereum\go-ethereum\cmd\utils\customflags.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package utils

import (
"encoding"
"errors"
"flag"
"fmt"
"math/big"
"os"
"os/user"
"path"
"strings"

"github.com/ethereum/go-ethereum/common/math"
"gopkg.in/urfave/cli.v1"
)

// Custom type which is registered in the flags library which cli uses for
// argument parsing. This allows us to expand Value to an absolute path when
// the argument is parsed
type DirectoryString struct {
Value string
}
```

```
func (self *DirectoryString) String() string {
return self.Value
}

func (self *DirectoryString) Set(value string) error {
self.Value = expandPath(value)
return nil
}

// Custom cli.Flag type which expand the received string to an absolute path.
// e.g. ~/.ethereum -> /home/username/.ethereum
type DirectoryFlag struct {
Name  string
Value DirectoryString
Usage string
}

func (self DirectoryFlag) String() string {
fmtString := "%s %v\t%v"
if len(self.Value.Value) > 0 {
fmtString = "%s \"%v\"\t%v"
}
return fmt.Sprintf(fmtString, prefixedNames(self.Name), self.Value.Value, self.Usage)
}

func eachName(longName string, fn func(string)) {
parts := strings.Split(longName, ",")
for _, name := range parts {
name = strings.Trim(name, " ")
fn(name)
}
}

// called by cli library, grabs variable from environment (if in env)
// and adds variable to flag set for parsing.
func (self DirectoryFlag) Apply(set *flag.FlagSet) {
eachName(self.Name, func(name string) {
set.Var(&self.Value, self.Name, self.Usage)
})
}

type TextMarshaler interface {
```

```
encoding.TextMarshaler
encoding.TextUnmarshaler
}

// textMarshalerVal turns a TextMarshaler into a flag.Value
type textMarshalerVal struct {
v TextMarshaler
}

func (v textMarshalerVal) String() string {
if v.v == nil {
return ""
}
text, _ := v.v.MarshalText()
return string(text)
}

func (v textMarshalerVal) Set(s string) error {
return v.v.UnmarshalText([]byte(s))
}

// TextMarshalerFlag wraps a TextMarshaler value.
type TextMarshalerFlag struct {
Name  string
Value TextMarshaler
Usage string
}

func (f TextMarshalerFlag) GetName() string {
return f.Name
}

func (f TextMarshalerFlag) String() string {
return fmt.Sprintf("%s \"%v\"\t%v", prefixedNames(f.Name), f.Value, f.Usage)
}

func (f TextMarshalerFlag) Apply(set *flag.FlagSet) {
eachName(f.Name, func(name string) {
set.Var(textMarshalerVal{f.Value}, f.Name, f.Usage)
})
}
```

```
// GlobalTextMarshaler returns the value of a TextMarshalerFlag from the global flag set.
func GlobalTextMarshaler(ctx *cli.Context, name string) TextMarshaler {
val := ctx.GlobalGeneric(name)
if val == nil {
return nil
}
return val.(textMarshalerVal).v
}

// BigFlag is a command line flag that accepts 256 bit big integers in decimal or
// hexadecimal syntax.
type BigFlag struct {
Name  string
Value *big.Int
Usage string
}

// bigValue turns *big.Int into a flag.Value
type bigValue big.Int

func (b *bigValue) String() string {
if b == nil {
return ""
}
return (*big.Int)(b).String()
}

func (b *bigValue) Set(s string) error {
int, ok := math.ParseBig256(s)
if !ok {
return errors.New("invalid integer syntax")
}
*b = (bigValue)(*int)
return nil
}

func (f BigFlag) GetName() string {
return f.Name
}

func (f BigFlag) String() string {
fmtString := "%s %v\t%v"
```

```
if f.Value != nil {
fmtString = "%s \"%v\"\t%v"
}
return fmt.Sprintf(fmtString, prefixedNames(f.Name), f.Value, f.Usage)
}

func (f BigFlag) Apply(set *flag.FlagSet) {
eachName(f.Name, func(name string) {
set.Var((*bigValue)(f.Value), f.Name, f.Usage)
})
}

// GlobalBig returns the value of a BigFlag from the global flag set.
func GlobalBig(ctx *cli.Context, name string) *big.Int {
val := ctx.GlobalGeneric(name)
if val == nil {
return nil
}
return (*big.Int)(val.(*bigValue))
}

func prefixFor(name string) (prefix string) {
if len(name) == 1 {
prefix = "-"
} else {
prefix = "--"
}

return
}

func prefixedNames(fullName string) (prefixed string) {
parts := strings.Split(fullName, ",")
for i, name := range parts {
name = strings.Trim(name, " ")
prefixed += prefixFor(name) + name
if i < len(parts)-1 {
prefixed += ", "
}
}
return
}
```

```
func (self DirectoryFlag) GetName() string {
return self.Name
}

func (self *DirectoryFlag) Set(value string) {
self.Value.Value = value
}

// Expands a file path
// 1. replace tilde with users home dir
// 2. expands embedded environment variables
// 3. cleans the path, e.g. /a/b/../c -> /a/c
// Note, it has limitations, e.g. ~someuser/tmp will not be expanded
func expandPath(p string) string {
if strings.HasPrefix(p, "~/") || strings.HasPrefix(p, "~\\") {
if home := homeDir(); home != "" {
p = home + p[1:]
}
}
return path.Clean(os.ExpandEnv(p))
}

func homeDir() string {
if home := os.Getenv("HOME"); home != "" {
return home
}
if usr, err := user.Current(); err == nil {
return usr.HomeDir
}
return ""
}

102:F:\git\coin\ethereum\go-ethereum\cmd\utils\customflags_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package utils

import (
"os"
"os/user"
"testing"
```

```
)

func TestPathExpansion(t *testing.T) {
user, _ := user.Current()
tests := map[string]string{
"/home/someuser/tmp": "/home/someuser/tmp",
"~/tmp":              user.HomeDir + "/tmp",
"~thisOtherUser/b/":  "~thisOtherUser/b",
"$DDDXXX/a/b":        "/tmp/a/b",
"/a/b/":              "/a/b",
}
os.Setenv("DDDXXX", "/tmp")
for test, expected := range tests {
got := expandPath(test)
if got != expected {
t.Errorf("test %s, got %s, expected %s\n", test, got, expected)
}
}
}

103:F:\git\coin\ethereum\go-ethereum\cmd\utils\fdlimit_freebsd.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// +build freebsd

package utils

import "syscall"

// This file is largely identical to fdlimit_unix.go,
// but Rlimit fields have type int64 on FreeBSD so it needs
// an extra conversion.

// raiseFdLimit tries to maximize the file descriptor allowance of this process
// to the maximum hard-limit allowed by the OS.
func raiseFdLimit(max uint64) error {
// Get the current limit
var limit syscall.Rlimit
if err := syscall.Getrlimit(syscall.RLIMIT_NOFILE, &limit); err != nil {
return err
}
// Try to update the limit to the max allowance
```

```
limit.Cur = limit.Max
if limit.Cur > int64(max) {
limit.Cur = int64(max)
}
if err := syscall.Setrlimit(syscall.RLIMIT_NOFILE, &limit); err != nil {
return err
}
return nil
}

// getFdLimit retrieves the number of file descriptors allowed to be opened by this
// process.
func getFdLimit() (int, error) {
var limit syscall.Rlimit
if err := syscall.Getrlimit(syscall.RLIMIT_NOFILE, &limit); err != nil {
return 0, err
}
return int(limit.Cur), nil
}

104:F:\git\coin\ethereum\go-ethereum\cmd\utils\fdlimit_test.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

package utils

import "testing"

// TestFileDescriptorLimits simply tests whether the file descriptor allowance
// per this process can be retrieved.
func TestFileDescriptorLimits(t *testing.T) {
target := 4096

if limit, err := getFdLimit(); err != nil || limit <= 0 {
t.Fatalf("failed to retrieve file descriptor limit (%d): %v", limit, err)
}
if err := raiseFdLimit(uint64(target)); err != nil {
t.Fatalf("failed to raise file allowance")
}
if limit, err := getFdLimit(); err != nil || limit < target {
t.Fatalf("failed to retrieve raised descriptor limit (have %v, want %v): %v", limit, target, err)
}
}
```

105:F:\git\coin\ethereum\go-ethereum\cmd\utils\fdlimit_unix.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// +build linux darwin netbsd openbsd solaris

package utils

import "syscall"

// raiseFdLimit tries to maximize the file descriptor allowance of this process
// to the maximum hard-limit allowed by the OS.
func raiseFdLimit(max uint64) error {
// Get the current limit
var limit syscall.Rlimit
if err := syscall.Getrlimit(syscall.RLIMIT_NOFILE, &limit); err != nil {
return err
}
// Try to update the limit to the max allowance
limit.Cur = limit.Max
if limit.Cur > max {
limit.Cur = max
}
if err := syscall.Setrlimit(syscall.RLIMIT_NOFILE, &limit); err != nil {
return err
}
return nil
}

// getFdLimit retrieves the number of file descriptors allowed to be opened by this
// process.
func getFdLimit() (int, error) {
var limit syscall.Rlimit
if err := syscall.Getrlimit(syscall.RLIMIT_NOFILE, &limit); err != nil {
return 0, err
}
return int(limit.Cur), nil
}

106:F:\git\coin\ethereum\go-ethereum\cmd\utils\fdlimit_windows.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

```
package utils

import "errors"

// raiseFdLimit tries to maximize the file descriptor allowance of this process
// to the maximum hard-limit allowed by the OS.
func raiseFdLimit(max uint64) error {
// This method is NOP by design:
//  * Linux/Darwin counterparts need to manually increase per process limits
//  * On Windows Go uses the CreateFile API, which is limited to 16K files, non
//    changeable from within a running process
// This way we can always "request" raising the limits, which will either have
// or not have effect based on the platform we're running on.
if max > 16384 {
return errors.New("file descriptor limit (16384) reached")
}
return nil
}

// getFdLimit retrieves the number of file descriptors allowed to be opened by this
// process.
func getFdLimit() (int, error) {
// Please see raiseFdLimit for the reason why we use hard coded 16K as the limit
return 16384, nil
}

107:F:\git\coin\ethereum\go-ethereum\cmd\utils\flags.go
// along with go-ethereum. If not, see <http://www.gnu.org/licenses/>.

// Package utils contains internal helper functions for go-ethereum commands.
package utils

import (
"crypto/ecdsa"
"fmt"
"io/ioutil"
"math/big"
"os"
"path/filepath"
"runtime"
"strconv"
"strings"
```

```
"github.com/ethereum/go-ethereum/accounts"
"github.com/ethereum/go-ethereum/accounts/keystore"
"github.com/ethereum/go-ethereum/common"
"github.com/ethereum/go-ethereum/consensus/ethash"
"github.com/ethereum/go-ethereum/core"
"github.com/ethereum/go-ethereum/core/state"
"github.com/ethereum/go-ethereum/core/vm"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/eth"
"github.com/ethereum/go-ethereum/eth/downloader"
"github.com/ethereum/go-ethereum/eth/gasprice"
"github.com/ethereum/go-ethereum/ethdb"
"github.com/ethereum/go-ethereum/ethstats"
"github.com/ethereum/go-ethereum/event"
"github.com/ethereum/go-ethereum/les"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/metrics"
"github.com/ethereum/go-ethereum/node"
"github.com/ethereum/go-ethereum/p2p"
"github.com/ethereum/go-ethereum/p2p/discover"
"github.com/ethereum/go-ethereum/p2p/discv5"
"github.com/ethereum/go-ethereum/p2p/nat"
"github.com/ethereum/go-ethereum/p2p/netutil"
"github.com/ethereum/go-ethereum/params"
whisper "github.com/ethereum/go-ethereum/whisper/whisperv5"
"gopkg.in/urfave/cli.v1"
)

var (
CommandHelpTemplate = `{{.cmd.Name}}{{if .cmd.Subcommands}} command{{end}}{{if
.cmd.Flags}} [command options]{{end}} [arguments...]
{{if .cmd.Description}}{{.cmd.Description}}
{{end}}{{if .cmd.Subcommands}}
SUBCOMMANDS:
{{range .cmd.Subcommands}}{{.cmd.Name}}{{with .cmd.ShortName}}, {{.cmd}}{{end}}{{ "\t"
}}{{.cmd.Usage}}
{{end}}{{end}}{{if .categorizedFlags}}
{{range $idx, $categorized := .categorizedFlags}}{{$categorized.Name}} OPTIONS:
{{range $categorized.Flags}}{{"\t"}}{{.}}
{{end}}
{{end}}{{end}}`
```

```
)

func init() {
cli.AppHelpTemplate = `{{.Name}} {{if .Flags}}[global options] {{end}}command{{if .Flags}}
[command options]{{end}} [arguments...]

VERSION:
   {{.Version}}

COMMANDS:
   {{range .Commands}}{{.Name}}{{with .ShortName}}, {{.}}{{end}}{{ "\t" }}{{.Usage}}
   {{end}}{{if .Flags}}
GLOBAL OPTIONS:
   {{range .Flags}}{{.}}
   {{end}}{{end}}
`

cli.CommandHelpTemplate = CommandHelpTemplate
}

// NewApp creates an app with sane defaults.
func NewApp(gitCommit, usage string) *cli.App {
app := cli.NewApp()
app.Name = filepath.Base(os.Args[0])
app.Author = ""
//app.Authors = nil
app.Email = ""
app.Version = params.Version
if gitCommit != "" {
app.Version += "-" + gitCommit[:8]
}
app.Usage = usage
return app
}

// These are all the command line flags we support.
// If you add to this list, please remember to include the
// flag in the appropriate command definition.
//
// The flags are defined here so their names and help texts
// are the same for all commands.
```

```
var (
// General settings
DataDirFlag = DirectoryFlag{
Name:  "datadir",
Usage: "Data directory for the databases and keystore",
Value: DirectoryString{node.DefaultDataDir()},
}
KeyStoreDirFlag = DirectoryFlag{
Name:  "keystore",
Usage: "Directory for the keystore (default = inside the datadir)",
}
NoUSBFlag = cli.BoolFlag{
Name:  "nousb",
Usage: "Disables monitoring for and managine USB hardware wallets",
}
NetworkIdFlag = cli.Uint64Flag{
Name:  "networkid",
Usage: "Network identifier (integer, 1=Frontier, 2=Morden (disused), 3=Ropsten, 4=Rinkeby)",
Value: eth.DefaultConfig.NetworkId,
}
TestnetFlag = cli.BoolFlag{
Name:  "testnet",
Usage: "Ropsten network: pre-configured proof-of-work test network",
}
RinkebyFlag = cli.BoolFlag{
Name:  "rinkeby",
Usage: "Rinkeby network: pre-configured proof-of-authority test network",
}
DevModeFlag = cli.BoolFlag{
Name:  "dev",
Usage: "Developer mode: pre-configured private network with several debugging flags",
}
IdentityFlag = cli.StringFlag{
Name:  "identity",
Usage: "Custom node name",
}
DocRootFlag = DirectoryFlag{
Name:  "docroot",
Usage: "Document Root for HTTPClient file scheme",
Value: DirectoryString{homeDir()},
}
FastSyncFlag = cli.BoolFlag{
```

```
Name:  "fast",
Usage: "Enable fast syncing through state downloads",
}
LightModeFlag = cli.BoolFlag{
Name:  "light",
Usage: "Enable light client mode",
}
defaultSyncMode = eth.DefaultConfig.SyncMode
SyncModeFlag    = TextMarshalerFlag{
Name:  "syncmode",
Usage: `Blockchain sync mode ("fast", "full", or "light")`,
Value: &defaultSyncMode,
}

LightServFlag = cli.IntFlag{
Name:  "lightserv",
Usage: "Maximum percentage of time allowed for serving LES requests (0-90)",
Value: 0,
}
LightPeersFlag = cli.IntFlag{
Name:  "lightpeers",
Usage: "Maximum number of LES client peers",
Value: 20,
}
LightKDFFlag = cli.BoolFlag{
Name:  "lightkdf",
Usage: "Reduce key-derivation RAM & CPU usage at some expense of KDF strength",
}
// Ethash settings
EthashCacheDirFlag = DirectoryFlag{
Name:  "ethash.cachedir",
Usage: "Directory to store the ethash verification caches (default = inside the datadir)",
}
EthashCachesInMemoryFlag = cli.IntFlag{
Name:  "ethash.cachesinmem",
Usage: "Number of recent ethash caches to keep in memory (16MB each)",
Value: eth.DefaultConfig.EthashCachesInMem,
}
EthashCachesOnDiskFlag = cli.IntFlag{
Name:  "ethash.cachesondisk",
Usage: "Number of recent ethash caches to keep on disk (16MB each)",
Value: eth.DefaultConfig.EthashCachesOnDisk,
```

```
}
EthashDatasetDirFlag = DirectoryFlag{
Name:  "ethash.dagdir",
Usage: "Directory to store the ethash mining DAGs (default = inside home folder)",
Value: DirectoryString{eth.DefaultConfig.EthashDatasetDir},
}
EthashDatasetsInMemoryFlag = cli.IntFlag{
Name:  "ethash.dagsinmem",
Usage: "Number of recent ethash mining DAGs to keep in memory (1+GB each)",
Value: eth.DefaultConfig.EthashDatasetsInMem,
}
EthashDatasetsOnDiskFlag = cli.IntFlag{
Name:  "ethash.dagsondisk",
Usage: "Number of recent ethash mining DAGs to keep on disk (1+GB each)",
Value: eth.DefaultConfig.EthashDatasetsOnDisk,
}
// Transaction pool settings
TxPoolPriceLimitFlag = cli.Uint64Flag{
Name:  "txpool.pricelimit",
Usage: "Minimum gas price limit to enforce for acceptance into the pool",
Value: eth.DefaultConfig.TxPool.PriceLimit,
}
TxPoolPriceBumpFlag = cli.Uint64Flag{
Name:  "txpool.pricebump",
Usage: "Price bump percentage to replace an already existing transaction",
Value: eth.DefaultConfig.TxPool.PriceBump,
}
TxPoolAccountSlotsFlag = cli.Uint64Flag{
Name:  "txpool.accountslots",
Usage: "Minimum number of executable transaction slots guaranteed per account",
Value: eth.DefaultConfig.TxPool.AccountSlots,
}
TxPoolGlobalSlotsFlag = cli.Uint64Flag{
Name:  "txpool.globalslots",
Usage: "Maximum number of executable transaction slots for all accounts",
Value: eth.DefaultConfig.TxPool.GlobalSlots,
}
TxPoolAccountQueueFlag = cli.Uint64Flag{
Name:  "txpool.accountqueue",
Usage: "Maximum number of non-executable transaction slots permitted per account",
Value: eth.DefaultConfig.TxPool.AccountQueue,
}
```

```
TxPoolGlobalQueueFlag = cli.Uint64Flag{
Name:  "txpool.globalqueue",
Usage: "Maximum number of non-executable transaction slots for all accounts",
Value: eth.DefaultConfig.TxPool.GlobalQueue,
}
TxPoolLifetimeFlag = cli.DurationFlag{
Name:  "txpool.lifetime",
Usage: "Maximum amount of time non-executable transaction are queued",
Value: eth.DefaultConfig.TxPool.Lifetime,
}
// Performance tuning settings
CacheFlag = cli.IntFlag{
Name:  "cache",
Usage: "Megabytes of memory allocated to internal caching (min 16MB / database forced)",
Value: 128,
}
TrieCacheGenFlag = cli.IntFlag{
Name:  "trie-cache-gens",
Usage: "Number of trie node generations to keep in memory",
Value: int(state.MaxTrieCacheGen),
}
// Miner settings
MiningEnabledFlag = cli.BoolFlag{
Name:  "mine",
Usage: "Enable mining",
}
MinerThreadsFlag = cli.IntFlag{
Name:  "minerthreads",
Usage: "Number of CPU threads to use for mining",
Value: runtime.NumCPU(),
}
TargetGasLimitFlag = cli.Uint64Flag{
Name:  "targetgaslimit",
Usage: "Target gas limit sets the artificial target gas floor for the blocks to mine",
Value: params.GenesisGasLimit.Uint64(),
}
EtherbaseFlag = cli.StringFlag{
Name:  "etherbase",
Usage: "Public address for block mining rewards (default = first account created)",
Value: "0",
}
GasPriceFlag = BigFlag{
```

```
Name:  "gasprice",
Usage: "Minimal gas price to accept for mining a transactions",
Value: eth.DefaultConfig.GasPrice,
}
ExtraDataFlag = cli.StringFlag{
Name:  "extradata",
Usage: "Block extra data set by the miner (default = client version)",
}
// Account settings
UnlockedAccountFlag = cli.StringFlag{
Name:  "unlock",
Usage: "Comma separated list of accounts to unlock",
Value: "",
}
PasswordFileFlag = cli.StringFlag{
Name:  "password",
Usage: "Password file to use for non-inteactive password input",
Value: "",
}

VMEnableDebugFlag = cli.BoolFlag{
Name:  "vmdebug",
Usage: "Record information useful for VM and contract debugging",
}
// Logging and debug settings
EthStatsURLFlag = cli.StringFlag{
Name:  "ethstats",
Usage: "Reporting URL of a ethstats service (nodename:secret@host:port)",
}
MetricsEnabledFlag = cli.BoolFlag{
Name:  metrics.MetricsEnabledFlag,
Usage: "Enable metrics collection and reporting",
}
FakePoWFlag = cli.BoolFlag{
Name:  "fakepow",
Usage: "Disables proof-of-work verification",
}
NoCompactionFlag = cli.BoolFlag{
Name:  "nocompaction",
Usage: "Disables db compaction after import",
}
// RPC settings
```

```
RPCEnabledFlag = cli.BoolFlag{
Name:  "rpc",
Usage: "Enable the HTTP-RPC server",
}
RPCListenAddrFlag = cli.StringFlag{
Name:  "rpcaddr",
Usage: "HTTP-RPC server listening interface",
Value: node.DefaultHTTPHost,
}
RPCPortFlag = cli.IntFlag{
Name:  "rpcport",
Usage: "HTTP-RPC server listening port",
Value: node.DefaultHTTPPort,
}
RPCCORSDomainFlag = cli.StringFlag{
Name:  "rpccorsdomain",
Usage: "Comma separated list of domains from which to accept cross origin requests (browser
enforced)",
Value: "",
}
RPCApiFlag = cli.StringFlag{
Name:  "rpcapi",
Usage: "API's offered over the HTTP-RPC interface",
Value: "",
}
IPCDisabledFlag = cli.BoolFlag{
Name:  "ipcdisable",
Usage: "Disable the IPC-RPC server",
}
IPCPathFlag = DirectoryFlag{
Name:  "ipcpath",
Usage: "Filename for IPC socket/pipe within the datadir (explicit paths escape it)",
}
WSEnabledFlag = cli.BoolFlag{
Name:  "ws",
Usage: "Enable the WS-RPC server",
}
WSListenAddrFlag = cli.StringFlag{
Name:  "wsaddr",
Usage: "WS-RPC server listening interface",
Value: node.DefaultWSHost,
}
```

```
WSPortFlag = cli.IntFlag{
Name:  "wsport",
Usage: "WS-RPC server listening port",
Value: node.DefaultWSPort,
}
WSApiFlag = cli.StringFlag{
Name:  "wsapi",
Usage: "API's offered over the WS-RPC interface",
Value: "",
}
WSAllowedOriginsFlag = cli.StringFlag{
Name:  "wsorigins",
Usage: "Origins from which to accept websockets requests",
Value: "",
}
ExecFlag = cli.StringFlag{
Name:  "exec",
Usage: "Execute JavaScript statement",
}
PreloadJSFlag = cli.StringFlag{
Name:  "preload",
Usage: "Comma separated list of JavaScript files to preload into the console",
}

// Network Settings
MaxPeersFlag = cli.IntFlag{
Name:  "maxpeers",
Usage: "Maximum number of network peers (network disabled if set to 0)",
Value: 25,
}
MaxPendingPeersFlag = cli.IntFlag{
Name:  "maxpendpeers",
Usage: "Maximum number of pending connection attempts (defaults used if set to 0)",
Value: 0,
}
ListenPortFlag = cli.IntFlag{
Name:  "port",
Usage: "Network listening port",
Value: 30303,
}
BootnodesFlag = cli.StringFlag{
Name:  "bootnodes",
```

```
Usage: "Comma separated enode URLs for P2P discovery bootstrap (set v4+v5 instead for light servers)",
Value: "",
}
BootnodesV4Flag = cli.StringFlag{
Name:  "bootnodesv4",
Usage: "Comma separated enode URLs for P2P v4 discovery bootstrap (light server, full nodes)",
Value: "",
}
BootnodesV5Flag = cli.StringFlag{
Name:  "bootnodesv5",
Usage: "Comma separated enode URLs for P2P v5 discovery bootstrap (light server, light nodes)",
Value: "",
}
NodeKeyFileFlag = cli.StringFlag{
Name:  "nodekey",
Usage: "P2P node key file",
}
NodeKeyHexFlag = cli.StringFlag{
Name:  "nodekeyhex",
Usage: "P2P node key as hex (for testing)",
}
NATFlag = cli.StringFlag{
Name:  "nat",
Usage: "NAT port mapping mechanism (any|none|upnp|pmp|extip:<IP>)",
Value: "any",
}
NoDiscoverFlag = cli.BoolFlag{
Name:  "nodiscover",
Usage: "Disables the peer discovery mechanism (manual peer addition)",
}
DiscoveryV5Flag = cli.BoolFlag{
Name:  "v5disc",
Usage: "Enables the experimental RLPx V5 (Topic Discovery) mechanism",
}
NetrestrictFlag = cli.StringFlag{
Name:  "netrestrict",
Usage: "Restricts network communication to the given IP networks (CIDR masks)",
}

// ATM the url is left to the user and deployment to
```

```
JSpathFlag = cli.StringFlag{
Name:  "jspath",
Usage: "JavaScript root path for `loadScript`",
Value: ".",
}

// Gas price oracle settings
GpoBlocksFlag = cli.IntFlag{
Name:  "gpoblocks",
Usage: "Number of recent blocks to check for gas prices",
Value: eth.DefaultConfig.GPO.Blocks,
}
GpoPercentileFlag = cli.IntFlag{
Name:  "gpopercentile",
Usage: "Suggested gas price is the given percentile of a set of recent transaction gas prices",
Value: eth.DefaultConfig.GPO.Percentile,
}
WhisperEnabledFlag = cli.BoolFlag{
Name:  "shh",
Usage: "Enable Whisper",
}
WhisperMaxMessageSizeFlag = cli.IntFlag{
Name:  "shh.maxmessagesize",
Usage: "Max message size accepted",
Value: int(whisper.DefaultMaxMessageSize),
}
WhisperMinPOWFlag = cli.Float64Flag{
Name:  "shh.pow",
Usage: "Minimum POW accepted",
Value: whisper.DefaultMinimumPoW,
}
)

// MakeDataDir retrieves the currently requested data directory, terminating
// if none (or the empty string) is specified. If the node is starting a testnet,
// the a subdirectory of the specified datadir will be used.
func MakeDataDir(ctx *cli.Context) string {
if path := ctx.GlobalString(DataDirFlag.Name); path != "" {
if ctx.GlobalBool(TestnetFlag.Name) {
return filepath.Join(path, "testnet")
}
if ctx.GlobalBool(RinkebyFlag.Name) {
```

```
return filepath.Join(path, "rinkeby")
}
return path
}
Fatalf("Cannot determine default data directory, please set manually (--datadir)")
return ""
}

// setNodeKey creates a node key from set command line flags, either loading it
// from a file or as a specified hex value. If neither flags were provided, this
// method returns nil and an emphemeral key is to be generated.
func setNodeKey(ctx *cli.Context, cfg *p2p.Config) {
var (
hex  = ctx.GlobalString(NodeKeyHexFlag.Name)
file = ctx.GlobalString(NodeKeyFileFlag.Name)
key  *ecdsa.PrivateKey
err  error
)
switch {
case file != "" && hex != "":
Fatalf("Options %q and %q are mutually exclusive", NodeKeyFileFlag.Name,
NodeKeyHexFlag.Name)
case file != "":
if key, err = crypto.LoadECDSA(file); err != nil {
Fatalf("Option %q: %v", NodeKeyFileFlag.Name, err)
}
cfg.PrivateKey = key
case hex != "":
if key, err = crypto.HexToECDSA(hex); err != nil {
Fatalf("Option %q: %v", NodeKeyHexFlag.Name, err)
}
cfg.PrivateKey = key
}
}

// setNodeUserIdent creates the user identifier from CLI flags.
func setNodeUserIdent(ctx *cli.Context, cfg *node.Config) {
if identity := ctx.GlobalString(IdentityFlag.Name); len(identity) > 0 {
cfg.UserIdent = identity
}
}
```

```
// setBootstrapNodes creates a list of bootstrap nodes from the command line
// flags, reverting to pre-configured ones if none have been specified.
func setBootstrapNodes(ctx *cli.Context, cfg *p2p.Config) {
urls := params.MainnetBootnodes
switch {
case ctx.GlobalIsSet(BootnodesFlag.Name) || ctx.GlobalIsSet(BootnodesV4Flag.Name):
if ctx.GlobalIsSet(BootnodesV4Flag.Name) {
urls = strings.Split(ctx.GlobalString(BootnodesV4Flag.Name), ",")
} else {
urls = strings.Split(ctx.GlobalString(BootnodesFlag.Name), ",")
}
case ctx.GlobalBool(TestnetFlag.Name):
urls = params.TestnetBootnodes
case ctx.GlobalBool(RinkebyFlag.Name):
urls = params.RinkebyBootnodes
}

cfg.BootstrapNodes = make([]*discover.Node, 0, len(urls))
for _, url := range urls {
node, err := discover.ParseNode(url)
if err != nil {
log.Error("Bootstrap URL invalid", "enode", url, "err", err)
continue
}
cfg.BootstrapNodes = append(cfg.BootstrapNodes, node)
}
}

// setBootstrapNodesV5 creates a list of bootstrap nodes from the command line
// flags, reverting to pre-configured ones if none have been specified.
func setBootstrapNodesV5(ctx *cli.Context, cfg *p2p.Config) {
urls := params.DiscoveryV5Bootnodes
switch {
case ctx.GlobalIsSet(BootnodesFlag.Name) || ctx.GlobalIsSet(BootnodesV5Flag.Name):
if ctx.GlobalIsSet(BootnodesV5Flag.Name) {
urls = strings.Split(ctx.GlobalString(BootnodesV5Flag.Name), ",")
} else {
urls = strings.Split(ctx.GlobalString(BootnodesFlag.Name), ",")
}
case ctx.GlobalBool(RinkebyFlag.Name):
urls = params.RinkebyV5Bootnodes
case cfg.BootstrapNodesV5 != nil:
```

```
return // already set, don't apply defaults.
}

cfg.BootstrapNodesV5 = make([]*discv5.Node, 0, len(urls))
for _, url := range urls {
node, err := discv5.ParseNode(url)
if err != nil {
log.Error("Bootstrap URL invalid", "enode", url, "err", err)
continue
}
cfg.BootstrapNodesV5 = append(cfg.BootstrapNodesV5, node)
}
}

// setListenAddress creates a TCP listening address string from set command
// line flags.
func setListenAddress(ctx *cli.Context, cfg *p2p.Config) {
if ctx.GlobalIsSet(ListenPortFlag.Name) {
cfg.ListenAddr = fmt.Sprintf(":%d", ctx.GlobalInt(ListenPortFlag.Name))
}
}

// setDiscoveryV5Address creates a UDP listening address string from set command
// line flags for the V5 discovery protocol.
func setDiscoveryV5Address(ctx *cli.Context, cfg *p2p.Config) {
if ctx.GlobalIsSet(ListenPortFlag.Name) {
cfg.DiscoveryV5Addr = fmt.Sprintf(":%d", ctx.GlobalInt(ListenPortFlag.Name)+1)
}
}

// setNAT creates a port mapper from command line flags.
func setNAT(ctx *cli.Context, cfg *p2p.Config) {
if ctx.GlobalIsSet(NATFlag.Name) {
natif, err := nat.Parse(ctx.GlobalString(NATFlag.Name))
if err != nil {
Fatalf("Option %s: %v", NATFlag.Name, err)
}
cfg.NAT = natif
}
}

// splitAndTrim splits input separated by a comma
```

```
// and trims excessive white space from the substrings.
func splitAndTrim(input string) []string {
result := strings.Split(input, ",")
for i, r := range result {
result[i] = strings.TrimSpace(r)
}
return result
}

// setHTTP creates the HTTP RPC listener interface string from the set
// command line flags, returning empty if the HTTP endpoint is disabled.
func setHTTP(ctx *cli.Context, cfg *node.Config) {
if ctx.GlobalBool(RPCEnabledFlag.Name) && cfg.HTTPHost == "" {
cfg.HTTPHost = "127.0.0.1"
if ctx.GlobalIsSet(RPCListenAddrFlag.Name) {
cfg.HTTPHost = ctx.GlobalString(RPCListenAddrFlag.Name)
}
}

if ctx.GlobalIsSet(RPCPortFlag.Name) {
cfg.HTTPPort = ctx.GlobalInt(RPCPortFlag.Name)
}
if ctx.GlobalIsSet(RPCCORSDomainFlag.Name) {
cfg.HTTPCors = splitAndTrim(ctx.GlobalString(RPCCORSDomainFlag.Name))
}
if ctx.GlobalIsSet(RPCApiFlag.Name) {
cfg.HTTPModules = splitAndTrim(ctx.GlobalString(RPCApiFlag.Name))
}
}

// setWS creates the WebSocket RPC listener interface string from the set
// command line flags, returning empty if the HTTP endpoint is disabled.
func setWS(ctx *cli.Context, cfg *node.Config) {
if ctx.GlobalBool(WSEnabledFlag.Name) && cfg.WSHost == "" {
cfg.WSHost = "127.0.0.1"
if ctx.GlobalIsSet(WSListenAddrFlag.Name) {
cfg.WSHost = ctx.GlobalString(WSListenAddrFlag.Name)
}
}

if ctx.GlobalIsSet(WSPortFlag.Name) {
cfg.WSPort = ctx.GlobalInt(WSPortFlag.Name)
```

```
}
if ctx.GlobalIsSet(WSAllowedOriginsFlag.Name) {
cfg.WSOrigins = splitAndTrim(ctx.GlobalString(WSAllowedOriginsFlag.Name))
}
if ctx.GlobalIsSet(WSApiFlag.Name) {
cfg.WSModules = splitAndTrim(ctx.GlobalString(WSApiFlag.Name))
}
}

// setIPC creates an IPC path configuration from the set command line flags,
// returning an empty string if IPC was explicitly disabled, or the set path.
func setIPC(ctx *cli.Context, cfg *node.Config) {
checkExclusive(ctx, IPCDisabledFlag, IPCPathFlag)
switch {
case ctx.GlobalBool(IPCDisabledFlag.Name):
cfg.IPCPath = ""
case ctx.GlobalIsSet(IPCPathFlag.Name):
cfg.IPCPath = ctx.GlobalString(IPCPathFlag.Name)
}
}

// makeDatabaseHandles raises out the number of allowed file handles per process
// for Geth and returns half of the allowance to assign to the database.
func makeDatabaseHandles() int {
if err := raiseFdLimit(2048); err != nil {
Fatalf("Failed to raise file descriptor allowance: %v", err)
}
limit, err := getFdLimit()
if err != nil {
Fatalf("Failed to retrieve file descriptor allowance: %v", err)
}
if limit > 2048 { // cap database file descriptors even if more is available
limit = 2048
}
return limit / 2 // Leave half for networking and other stuff
}

// MakeAddress converts an account specified directly as a hex encoded string or
// a key index in the key store to an internal account representation.
func MakeAddress(ks *keystore.KeyStore, account string) (accounts.Account, error) {
// If the specified account is a valid address, return it
if common.IsHexAddress(account) {
```

```
return accounts.Account{Address: common.HexToAddress(account)}, nil
}
// Otherwise try to interpret the account as a keystore index
index, err := strconv.Atoi(account)
if err != nil || index < 0 {
return accounts.Account{}, fmt.Errorf("invalid account address or index %q", account)
}
accs := ks.Accounts()
if len(accs) <= index {
return accounts.Account{}, fmt.Errorf("index %d higher than number of accounts %d", index,
len(accs))
}
return accs[index], nil
}

// setEtherbase retrieves the etherbase either from the directly specified
// command line flags or from the keystore if CLI indexed.
func setEtherbase(ctx *cli.Context, ks *keystore.KeyStore, cfg *eth.Config) {
if ctx.GlobalIsSet(EtherbaseFlag.Name) {
account, err := MakeAddress(ks, ctx.GlobalString(EtherbaseFlag.Name))
if err != nil {
Fatalf("Option %q: %v", EtherbaseFlag.Name, err)
}
cfg.Etherbase = account.Address
return
}
accounts := ks.Accounts()
if (cfg.Etherbase == common.Address{}) {
if len(accounts) > 0 {
cfg.Etherbase = accounts[0].Address
} else {
log.Warn("No etherbase set and no accounts found as default")
}
}
}

// MakePasswordList reads password lines from the file specified by the global --password flag.
func MakePasswordList(ctx *cli.Context) []string {
path := ctx.GlobalString(PasswordFileFlag.Name)
if path == "" {
return nil
}
```

```
text, err := ioutil.ReadFile(path)
if err != nil {
Fatalf("Failed to read password file: %v", err)
}
lines := strings.Split(string(text), "\n")
// Sanitise DOS line endings.
for i := range lines {
lines[i] = strings.TrimRight(lines[i], "\r")
}
return lines
}

func SetP2PConfig(ctx *cli.Context, cfg *p2p.Config) {
setNodeKey(ctx, cfg)
setNAT(ctx, cfg)
setListenAddress(ctx, cfg)
setDiscoveryV5Address(ctx, cfg)
setBootstrapNodes(ctx, cfg)
setBootstrapNodesV5(ctx, cfg)

if ctx.GlobalIsSet(MaxPeersFlag.Name) {
cfg.MaxPeers = ctx.GlobalInt(MaxPeersFlag.Name)
}
if ctx.GlobalIsSet(MaxPendingPeersFlag.Name) {
cfg.MaxPendingPeers = ctx.GlobalInt(MaxPendingPeersFlag.Name)
}
if ctx.GlobalIsSet(NoDiscoverFlag.Name) || ctx.GlobalBool(LightModeFlag.Name) {
cfg.NoDiscovery = true
}

// if we're running a light client or server, force enable the v5 peer discovery
// unless it is explicitly disabled with --nodiscover note that explicitly specifying
// --v5disc overrides --nodiscover, in which case the later only disables v4 discovery
forceV5Discovery := (ctx.GlobalBool(LightModeFlag.Name) || ctx.GlobalInt(LightServFlag.Name) >
0) && !ctx.GlobalBool(NoDiscoverFlag.Name)
if ctx.GlobalIsSet(DiscoveryV5Flag.Name) {
cfg.DiscoveryV5 = ctx.GlobalBool(DiscoveryV5Flag.Name)
} else if forceV5Discovery {
cfg.DiscoveryV5 = true
}

if netrestrict := ctx.GlobalString(NetrestrictFlag.Name); netrestrict != "" {
```

```
list, err := netutil.ParseNetlist(netrestrict)
if err != nil {
Fatalf("Option %q: %v", NetrestrictFlag.Name, err)
}
cfg.NetRestrict = list
}

if ctx.GlobalBool(DevModeFlag.Name) {
// --dev mode can't use p2p networking.
cfg.MaxPeers = 0
cfg.ListenAddr = ":0"
cfg.DiscoveryV5Addr = ":0"
cfg.NoDiscovery = true
cfg.DiscoveryV5 = false
}
}

// SetNodeConfig applies node-related command line flags to the config.
func SetNodeConfig(ctx *cli.Context, cfg *node.Config) {
SetP2PConfig(ctx, &cfg.P2P)
setIPC(ctx, cfg)
setHTTP(ctx, cfg)
setWS(ctx, cfg)
setNodeUserIdent(ctx, cfg)

switch {
case ctx.GlobalIsSet(DataDirFlag.Name):
cfg.DataDir = ctx.GlobalString(DataDirFlag.Name)
case ctx.GlobalBool(DevModeFlag.Name):
cfg.DataDir = filepath.Join(os.TempDir(), "ethereum_dev_mode")
case ctx.GlobalBool(TestnetFlag.Name):
cfg.DataDir = filepath.Join(node.DefaultDataDir(), "testnet")
case ctx.GlobalBool(RinkebyFlag.Name):
cfg.DataDir = filepath.Join(node.DefaultDataDir(), "rinkeby")
}

if ctx.GlobalIsSet(KeyStoreDirFlag.Name) {
cfg.KeyStoreDir = ctx.GlobalString(KeyStoreDirFlag.Name)
}
if ctx.GlobalIsSet(LightKDFFlag.Name) {
cfg.UseLightweightKDF = ctx.GlobalBool(LightKDFFlag.Name)
}
```

```
if ctx.GlobalIsSet(NoUSBFlag.Name) {
cfg.NoUSB = ctx.GlobalBool(NoUSBFlag.Name)
}
}

func setGPO(ctx *cli.Context, cfg *gasprice.Config) {
if ctx.GlobalIsSet(GpoBlocksFlag.Name) {
cfg.Blocks = ctx.GlobalInt(GpoBlocksFlag.Name)
}
if ctx.GlobalIsSet(GpoPercentileFlag.Name) {
cfg.Percentile = ctx.GlobalInt(GpoPercentileFlag.Name)
}
}

func setTxPool(ctx *cli.Context, cfg *core.TxPoolConfig) {
if ctx.GlobalIsSet(TxPoolPriceLimitFlag.Name) {
cfg.PriceLimit = ctx.GlobalUint64(TxPoolPriceLimitFlag.Name)
}
if ctx.GlobalIsSet(TxPoolPriceBumpFlag.Name) {
cfg.PriceBump = ctx.GlobalUint64(TxPoolPriceBumpFlag.Name)
}
if ctx.GlobalIsSet(TxPoolAccountSlotsFlag.Name) {
cfg.AccountSlots = ctx.GlobalUint64(TxPoolAccountSlotsFlag.Name)
}
if ctx.GlobalIsSet(TxPoolGlobalSlotsFlag.Name) {
cfg.GlobalSlots = ctx.GlobalUint64(TxPoolGlobalSlotsFlag.Name)
}
if ctx.GlobalIsSet(TxPoolAccountQueueFlag.Name) {
cfg.AccountQueue = ctx.GlobalUint64(TxPoolAccountQueueFlag.Name)
}
if ctx.GlobalIsSet(TxPoolGlobalQueueFlag.Name) {
cfg.GlobalQueue = ctx.GlobalUint64(TxPoolGlobalQueueFlag.Name)
}
if ctx.GlobalIsSet(TxPoolLifetimeFlag.Name) {
cfg.Lifetime = ctx.GlobalDuration(TxPoolLifetimeFlag.Name)
}
}

func setEthash(ctx *cli.Context, cfg *eth.Config) {
if ctx.GlobalIsSet(EthashCacheDirFlag.Name) {
cfg.EthashCacheDir = ctx.GlobalString(EthashCacheDirFlag.Name)
}
```

```
if ctx.GlobalIsSet(EthashDatasetDirFlag.Name) {
cfg.EthashDatasetDir = ctx.GlobalString(EthashDatasetDirFlag.Name)
}
if ctx.GlobalIsSet(EthashCachesInMemoryFlag.Name) {
cfg.EthashCachesInMem = ctx.GlobalInt(EthashCachesInMemoryFlag.Name)
}
if ctx.GlobalIsSet(EthashCachesOnDiskFlag.Name) {
cfg.EthashCachesOnDisk = ctx.GlobalInt(EthashCachesOnDiskFlag.Name)
}
if ctx.GlobalIsSet(EthashDatasetsInMemoryFlag.Name) {
cfg.EthashDatasetsInMem = ctx.GlobalInt(EthashDatasetsInMemoryFlag.Name)
}
if ctx.GlobalIsSet(EthashDatasetsOnDiskFlag.Name) {
cfg.EthashDatasetsOnDisk = ctx.GlobalInt(EthashDatasetsOnDiskFlag.Name)
}
}

func checkExclusive(ctx *cli.Context, flags ...cli.Flag) {
set := make([]string, 0, 1)
for _, flag := range flags {
if ctx.GlobalIsSet(flag.GetName()) {
set = append(set, "--"+flag.GetName())
}
}
if len(set) > 1 {
Fatalf("flags %v can't be used at the same time", strings.Join(set, ", "))
}
}

// SetShhConfig applies shh-related command line flags to the config.
func SetShhConfig(ctx *cli.Context, stack *node.Node, cfg *whisper.Config) {
if ctx.GlobalIsSet(WhisperMaxMessageSizeFlag.Name) {
cfg.MaxMessageSize = uint32(ctx.GlobalUint(WhisperMaxMessageSizeFlag.Name))
}
if ctx.GlobalIsSet(WhisperMinPOWFlag.Name) {
cfg.MinimumAcceptedPOW = ctx.GlobalFloat64(WhisperMinPOWFlag.Name)
}
}

// SetEthConfig applies eth-related command line flags to the config.
func SetEthConfig(ctx *cli.Context, stack *node.Node, cfg *eth.Config) {
// Avoid conflicting network flags
```

```
checkExclusive(ctx, DevModeFlag, TestnetFlag, RinkebyFlag)
checkExclusive(ctx, FastSyncFlag, LightModeFlag, SyncModeFlag)

ks := stack.AccountManager().Backends(keystore.KeyStoreType)[0].(*keystore.KeyStore)
setEtherbase(ctx, ks, cfg)
setGPO(ctx, &cfg.GPO)
setTxPool(ctx, &cfg.TxPool)
setEthash(ctx, cfg)

switch {
case ctx.GlobalIsSet(SyncModeFlag.Name):
cfg.SyncMode = *GlobalTextMarshaler(ctx, SyncModeFlag.Name).(*downloader.SyncMode)
case ctx.GlobalBool(FastSyncFlag.Name):
cfg.SyncMode = downloader.FastSync
case ctx.GlobalBool(LightModeFlag.Name):
cfg.SyncMode = downloader.LightSync
}
if ctx.GlobalIsSet(LightServFlag.Name) {
cfg.LightServ = ctx.GlobalInt(LightServFlag.Name)
}
if ctx.GlobalIsSet(LightPeersFlag.Name) {
cfg.LightPeers = ctx.GlobalInt(LightPeersFlag.Name)
}
if ctx.GlobalIsSet(NetworkIdFlag.Name) {
cfg.NetworkId = ctx.GlobalUint64(NetworkIdFlag.Name)
}

// Ethereum needs to know maxPeers to calculate the light server peer ratio.
// TODO(fjl): ensure Ethereum can get MaxPeers from node.
cfg.MaxPeers = ctx.GlobalInt(MaxPeersFlag.Name)

if ctx.GlobalIsSet(CacheFlag.Name) {
cfg.DatabaseCache = ctx.GlobalInt(CacheFlag.Name)
}
cfg.DatabaseHandles = makeDatabaseHandles()

if ctx.GlobalIsSet(MinerThreadsFlag.Name) {
cfg.MinerThreads = ctx.GlobalInt(MinerThreadsFlag.Name)
}
if ctx.GlobalIsSet(DocRootFlag.Name) {
cfg.DocRoot = ctx.GlobalString(DocRootFlag.Name)
}
```

```
if ctx.GlobalIsSet(ExtraDataFlag.Name) {
cfg.ExtraData = []byte(ctx.GlobalString(ExtraDataFlag.Name))
}
if ctx.GlobalIsSet(GasPriceFlag.Name) {
cfg.GasPrice = GlobalBig(ctx, GasPriceFlag.Name)
}
if ctx.GlobalIsSet(VMEnableDebugFlag.Name) {
// TODO(fjl): force-enable this in --dev mode
cfg.EnablePreimageRecording = ctx.GlobalBool(VMEnableDebugFlag.Name)
}

// Override any default configs for hard coded networks.
switch {
case ctx.GlobalBool(TestnetFlag.Name):
if !ctx.GlobalIsSet(NetworkIdFlag.Name) {
cfg.NetworkId = 3
}
cfg.Genesis = core.DefaultTestnetGenesisBlock()
case ctx.GlobalBool(RinkebyFlag.Name):
if !ctx.GlobalIsSet(NetworkIdFlag.Name) {
cfg.NetworkId = 4
}
cfg.Genesis = core.DefaultRinkebyGenesisBlock()
case ctx.GlobalBool(DevModeFlag.Name):
cfg.Genesis = core.DevGenesisBlock()
if !ctx.GlobalIsSet(GasPriceFlag.Name) {
cfg.GasPrice = new(big.Int)
}
cfg.PowTest = true
}

// TODO(fjl): move trie cache generations into config
if gen := ctx.GlobalInt(TrieCacheGenFlag.Name); gen > 0 {
state.MaxTrieCacheGen = uint16(gen)
}
}

// RegisterEthService adds an Ethereum client to the stack.
func RegisterEthService(stack *node.Node, cfg *eth.Config) {
var err error
if cfg.SyncMode == downloader.LightSync {
err = stack.Register(func(ctx *node.ServiceContext) (node.Service, error) {
```

```
return les.New(ctx, cfg)
})
} else {
err = stack.Register(func(ctx *node.ServiceContext) (node.Service, error) {
fullNode, err := eth.New(ctx, cfg)
if fullNode != nil && cfg.LightServ > 0 {
ls, _ := les.NewLesServer(fullNode, cfg)
fullNode.AddLesServer(ls)
}
return fullNode, err
})
}
if err != nil {
Fatalf("Failed to register the Ethereum service: %v", err)
}
}

// RegisterShhService configures Whisper and adds it to the given node.
func RegisterShhService(stack *node.Node, cfg *whisper.Config) {
if err := stack.Register(func(n *node.ServiceContext) (node.Service, error) {
return whisper.New(cfg), nil
}); err != nil {
Fatalf("Failed to register the Whisper service: %v", err)
}
}

// RegisterEthStatsService configures the Ethereum Stats daemon and adds it to
// th egiven node.
func RegisterEthStatsService(stack *node.Node, url string) {
if err := stack.Register(func(ctx *node.ServiceContext) (node.Service, error) {
// Retrieve both eth and les services
var ethServ *eth.Ethereum
ctx.Service(&ethServ)

var lesServ *les.LightEthereum
ctx.Service(&lesServ)

return ethstats.New(url, ethServ, lesServ)
}); err != nil {
Fatalf("Failed to register the Ethereum Stats service: %v", err)
}
}
```

```
// SetupNetwork configures the system for either the main net or some test network.
func SetupNetwork(ctx *cli.Context) {
// TODO(fjl): move target gas limit into config
params.TargetGasLimit = new(big.Int).SetUint64(ctx.GlobalUint64(TargetGasLimitFlag.Name))
}

// MakeChainDatabase open an LevelDB using the flags passed to the client and will hard crash if it fails.
func MakeChainDatabase(ctx *cli.Context, stack *node.Node) ethdb.Database {
var (
cache   = ctx.GlobalInt(CacheFlag.Name)
handles = makeDatabaseHandles()
)
name := "chaindata"
if ctx.GlobalBool(LightModeFlag.Name) {
name = "lightchaindata"
}
chainDb, err := stack.OpenDatabase(name, cache, handles)
if err != nil {
Fatalf("Could not open database: %v", err)
}
return chainDb
}

func MakeGenesis(ctx *cli.Context) *core.Genesis {
var genesis *core.Genesis
switch {
case ctx.GlobalBool(TestnetFlag.Name):
genesis = core.DefaultTestnetGenesisBlock()
case ctx.GlobalBool(RinkebyFlag.Name):
genesis = core.DefaultRinkebyGenesisBlock()
case ctx.GlobalBool(DevModeFlag.Name):
genesis = core.DevGenesisBlock()
}
return genesis
}

// MakeChain creates a chain manager from set command line flags.
func MakeChain(ctx *cli.Context, stack *node.Node) (chain *core.BlockChain, chainDb ethdb.Database) {
var err error
```

```
chainDb = MakeChainDatabase(ctx, stack)

engine := ethash.NewFaker()
if !ctx.GlobalBool(FakePoWFlag.Name) {
engine = ethash.New("", 1, 0, "", 1, 0)
}
config, _, err := core.SetupGenesisBlock(chainDb, MakeGenesis(ctx))
if err != nil {
Fatalf("%v", err)
}
vmcfg := vm.Config{EnablePreimageRecording: ctx.GlobalBool(VMEnableDebugFlag.Name)}
chain, err = core.NewBlockChain(chainDb, config, engine, new(event.TypeMux), vmcfg)
if err != nil {
Fatalf("Can't create BlockChain: %v", err)
}
return chain, chainDb
}

// MakeConsolePreloads retrieves the absolute paths for the console JavaScript
// scripts to preload before starting.
func MakeConsolePreloads(ctx *cli.Context) []string {
// Skip preloading if there's nothing to preload
if ctx.GlobalString(PreloadJSFlag.Name) == "" {
return nil
}
// Otherwise resolve absolute paths and return them
preloads := []string{}

assets := ctx.GlobalString(JSpathFlag.Name)
for _, file := range strings.Split(ctx.GlobalString(PreloadJSFlag.Name), ",") {
preloads = append(preloads, common.AbsolutePath(assets, strings.TrimSpace(file)))
}
return preloads
}

// MigrateFlags sets the global flag from a local flag when it's set.
// This is a temporary function used for migrating old command/flags to the
// new format.
//
// e.g. geth account new --keystore /tmp/mykeystore --lightkdf
//
// is equivalent after calling this method with:
```

```
//
// geth --keystore /tmp/mykeystore --lightkdf account new
//
// This allows the use of the existing configuration functionality.
// When all flags are migrated this function can be removed and the existing
// configuration functionality must be changed that is uses local flags
func MigrateFlags(action func(ctx *cli.Context) error) func(*cli.Context) error {
return func(ctx *cli.Context) error {
for _, name := range ctx.FlagNames() {
if ctx.IsSet(name) {
ctx.GlobalSet(name, ctx.String(name))
}
}
return action(ctx)
}
}
```

108:F:\git\coin\ethereum\go-ethereum\cmd\wnode\main.go

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

// This is a simple Whisper node. It could be used as a stand-alone bootstrap node.
// Also, could be used for different test and diagnostics purposes.

package main

import (
"bufio"
"crypto/ecdsa"
"crypto/sha512"
"encoding/binary"
"encoding/hex"
"flag"
"fmt"
"io/ioutil"
"os"
"path/filepath"
"strconv"
"strings"
"time"

"github.com/ethereum/go-ethereum/cmd/utils"
"github.com/ethereum/go-ethereum/common"
```

```
"github.com/ethereum/go-ethereum/console"
"github.com/ethereum/go-ethereum/crypto"
"github.com/ethereum/go-ethereum/log"
"github.com/ethereum/go-ethereum/p2p"
"github.com/ethereum/go-ethereum/p2p/discover"
"github.com/ethereum/go-ethereum/p2p/nat"
"github.com/ethereum/go-ethereum/whisper/mailserver"
whisper "github.com/ethereum/go-ethereum/whisper/whisperv5"
"golang.org/x/crypto/pbkdf2"
)

const quitCommand = "~Q"

// singletons
var (
server     *p2p.Server
shh        *whisper.Whisper
done       chan struct{}
mailServer mailserver.WMailServer

input = bufio.NewReader(os.Stdin)
)

// encryption
var (
symKey     []byte
pub        *ecdsa.PublicKey
asymKey    *ecdsa.PrivateKey
nodeid     *ecdsa.PrivateKey
topic      whisper.TopicType
asymKeyID  string
filterID   string
symPass    string
msPassword string
)

// cmd arguments
var (
bootstrapMode  = flag.Bool("standalone", false, "boostrap node: don't actively connect to peers, wait for incoming connections")
forwarderMode  = flag.Bool("forwarder", false, "forwarder mode: only forward messages, neither send nor decrypt messages")
```

```
mailServerMode = flag.Bool("mailserver", false, "mail server mode: delivers expired messages on demand")
requestMail    = flag.Bool("mailclient", false, "request expired messages from the bootstrap server")
asymmetricMode = flag.Bool("asym", false, "use asymmetric encryption")
generateKey    = flag.Bool("generatekey", false, "generate and show the private key")
fileExMode     = flag.Bool("fileexchange", false, "file exchange mode")
testMode       = flag.Bool("test", false, "use of predefined parameters for diagnostics")
echoMode       = flag.Bool("echo", false, "echo mode: prints some arguments for diagnostics")

argVerbosity = flag.Int("verbosity", int(log.LvlError), "log verbosity level")
argTTL       = flag.Uint("ttl", 30, "time-to-live for messages in seconds")
argWorkTime  = flag.Uint("work", 5, "work time in seconds")
argMaxSize   = flag.Uint("maxsize", uint(whisper.DefaultMaxMessageSize), "max size of message")
argPoW       = flag.Float64("pow", whisper.DefaultMinimumPoW, "PoW for normal messages in float format (e.g. 2.7)")
argServerPoW = flag.Float64("mspow", whisper.DefaultMinimumPoW, "PoW requirement for Mail Server request")

argIP      = flag.String("ip", "", "IP address and port of this node (e.g. 127.0.0.1:30303)")
argPub     = flag.String("pub", "", "public key for asymmetric encryption")
argDBPath  = flag.String("dbpath", "", "path to the server's DB directory")
argIDFile  = flag.String("idfile", "", "file name with node id (private key)")
argEnode   = flag.String("boot", "", "bootstrap node you want to connect to (e.g. enode://e454.....08d50@52.176.211.200:16428)")
argTopic   = flag.String("topic", "", "topic in hexadecimal format (e.g. 70a4beef)")
argSaveDir = flag.String("savedir", "", "directory where incoming messages will be saved as files")
)

func main() {
processArgs()
initialize()
run()
}

func processArgs() {
flag.Parse()

if len(*argIDFile) > 0 {
var err error
nodeid, err = crypto.LoadECDSA(*argIDFile)
```

```
if err != nil {
utils.Fatalf("Failed to load file [%s]: %s.", *argIDFile, err)
}
}

const enodePrefix = "enode://"
if len(*argEnode) > 0 {
if (*argEnode)[:len(enodePrefix)] != enodePrefix {
*argEnode = enodePrefix + *argEnode
}
}

if len(*argTopic) > 0 {
x, err := hex.DecodeString(*argTopic)
if err != nil {
utils.Fatalf("Failed to parse the topic: %s", err)
}
topic = whisper.BytesToTopic(x)
}

if *asymmetricMode && len(*argPub) > 0 {
pub = crypto.ToECDSAPub(common.FromHex(*argPub))
if !isKeyValid(pub) {
utils.Fatalf("invalid public key")
}
}

if len(*argSaveDir) > 0 {
if _, err := os.Stat(*argSaveDir); os.IsNotExist(err) {
utils.Fatalf("Download directory '%s' does not exist", *argSaveDir)
}
} else if *fileExMode {
utils.Fatalf("Parameter 'savedir' is mandatory for file exchange mode")
}

if *echoMode {
echo()
}
}

func echo() {
fmt.Printf("ttl = %d \n", *argTTL)
```

```
fmt.Printf("workTime = %d \n", *argWorkTime)
fmt.Printf("pow = %f \n", *argPoW)
fmt.Printf("mspow = %f \n", *argServerPoW)
fmt.Printf("ip = %s \n", *argIP)
fmt.Printf("pub = %s \n", common.ToHex(crypto.FromECDSAPub(pub)))
fmt.Printf("idfile = %s \n", *argIDFile)
fmt.Printf("dbpath = %s \n", *argDBPath)
fmt.Printf("boot = %s \n", *argEnode)
}

func initialize() {
log.Root().SetHandler(log.LvlFilterHandler(log.Lvl(*argVerbosity), log.StreamHandler(os.Stderr,
log.TerminalFormat(false))))

done = make(chan struct{})
var peers []*discover.Node
var err error

if *generateKey {
key, err := crypto.GenerateKey()
if err != nil {
utils.Fatalf("Failed to generate private key: %s", err)
}
k := hex.EncodeToString(crypto.FromECDSA(key))
fmt.Printf("Random private key: %s \n", k)
os.Exit(0)
}

if *testMode {
symPass = "wwww" // ascii code: 0x77777777
msPassword = "wwww"
}

if *bootstrapMode {
if len(*argIP) == 0 {
argIP = scanLineA("Please enter your IP and port (e.g. 127.0.0.1:30348): ")
}
} else {
if len(*argEnode) == 0 {
argEnode = scanLineA("Please enter the peer's enode: ")
}
peer := discover.MustParseNode(*argEnode)
```

```
peers = append(peers, peer)
}

cfg := &whisper.Config{
MaxMessageSize:     uint32(*argMaxSize),
MinimumAcceptedPOW: *argPoW,
}

if *mailServerMode {
if len(msPassword) == 0 {
msPassword, err = console.Stdin.PromptPassword("Please enter the Mail Server password: ")
if err != nil {
utils.Fatalf("Failed to read Mail Server password: %s", err)
}
}

shh = whisper.New(cfg)
shh.RegisterServer(&mailServer)
mailServer.Init(shh, *argDBPath, msPassword, *argServerPoW)
} else {
shh = whisper.New(cfg)
}

if *argPoW != whisper.DefaultMinimumPoW {
err := shh.SetMinimumPoW(*argPoW)
if err != nil {
utils.Fatalf("Failed to set PoW: %s", err)
}
}

if uint32(*argMaxSize) != whisper.DefaultMaxMessageSize {
err := shh.SetMaxMessageSize(uint32(*argMaxSize))
if err != nil {
utils.Fatalf("Failed to set max message size: %s", err)
}
}

asymKeyID, err = shh.NewKeyPair()
if err != nil {
utils.Fatalf("Failed to generate a new key pair: %s", err)
}
```

```
asymKey, err = shh.GetPrivateKey(asymKeyID)
if err != nil {
utils.Fatalf("Failed to retrieve a new key pair: %s", err)
}

if nodeid == nil {
tmpID, err := shh.NewKeyPair()
if err != nil {
utils.Fatalf("Failed to generate a new key pair: %s", err)
}

nodeid, err = shh.GetPrivateKey(tmpID)
if err != nil {
utils.Fatalf("Failed to retrieve a new key pair: %s", err)
}
}

maxPeers := 80
if *bootstrapMode {
maxPeers = 800
}

server = &p2p.Server{
Config: p2p.Config{
PrivateKey:     nodeid,
MaxPeers:       maxPeers,
Name:           common.MakeName("wnode", "5.0"),
Protocols:      shh.Protocols(),
ListenAddr:     *argIP,
NAT:            nat.Any(),
BootstrapNodes: peers,
StaticNodes:    peers,
TrustedNodes:   peers,
},
}
}

func startServer() {
err := server.Start()
if err != nil {
utils.Fatalf("Failed to start Whisper peer: %s.", err)
}
```

```
fmt.Printf("my public key: %s \n", common.ToHex(crypto.FromECDSAPub(&asymKey.PublicKey)))
fmt.Println(server.NodeInfo().Enode)

if *bootstrapMode {
configureNode()
fmt.Println("Bootstrap Whisper node started")
} else {
fmt.Println("Whisper node started")
// first see if we can establish connection, then ask for user input
waitForConnection(true)
configureNode()
}

if !*forwarderMode {
fmt.Printf("Please type the message. To quit type: '%s'\n", quitCommand)
}
}

func isKeyValid(k *ecdsa.PublicKey) bool {
return k.X != nil && k.Y != nil
}

func configureNode() {
var err error
var p2pAccept bool

if *forwarderMode {
return
}

if *asymmetricMode {
if len(*argPub) == 0 {
s := scanLine("Please enter the peer's public key: ")
b := common.FromHex(s)
if b == nil {
utils.Fatalf("Error: can not convert hexadecimal string")
}
pub = crypto.ToECDSAPub(b)
if !isKeyValid(pub) {
utils.Fatalf("Error: invalid public key")
}
```

```
}
}

if *requestMail {
p2pAccept = true
if len(msPassword) == 0 {
msPassword, err = console.Stdin.PromptPassword("Please enter the Mail Server password: ")
if err != nil {
utils.Fatalf("Failed to read Mail Server password: %s", err)
}
}
}

if !*asymmetricMode && !*forwarderMode {
if len(symPass) == 0 {
symPass, err = console.Stdin.PromptPassword("Please enter the password for symmetric
encryption: ")
if err != nil {
utils.Fatalf("Failed to read passphrase: %v", err)
}
}

symKeyID, err := shh.AddSymKeyFromPassword(symPass)
if err != nil {
utils.Fatalf("Failed to create symmetric key: %s", err)
}
symKey, err = shh.GetSymKey(symKeyID)
if err != nil {
utils.Fatalf("Failed to save symmetric key: %s", err)
}
if len(*argTopic) == 0 {
generateTopic([]byte(symPass))
}

fmt.Printf("Filter is configured for the topic: %x \n", topic)
}

if *mailServerMode {
if len(*argDBPath) == 0 {
argDBPath = scanLineA("Please enter the path to DB file: ")
}
}
```

```
filter := whisper.Filter{
KeySym:   symKey,
KeyAsym:  asymKey,
Topics:   [][]byte{topic[:]},
AllowP2P: p2pAccept,
}
filterID, err = shh.Subscribe(&filter)
if err != nil {
utils.Fatalf("Failed to install filter: %s", err)
}
}

func generateTopic(password []byte) {
x := pbkdf2.Key(password, password, 4096, 128, sha512.New)
for i := 0; i < len(x); i++ {
topic[i%whisper.TopicLength] ^= x[i]
}
}

func waitForConnection(timeout bool) {
var cnt int
var connected bool
for !connected {
time.Sleep(time.Millisecond * 50)
connected = server.PeerCount() > 0
if timeout {
cnt++
if cnt > 1000 {
utils.Fatalf("Timeout expired, failed to connect")
}
}
}

fmt.Println("Connected to peer.")
}

func run() {
defer mailServer.Close()
startServer()
defer server.Stop()
shh.Start(nil)
```

```
defer shh.Stop()

if !*forwarderMode {
go messageLoop()
}

if *requestMail {
requestExpiredMessagesLoop()
} else if *fileExMode {
sendFilesLoop()
} else {
sendLoop()
}
}

func sendLoop() {
for {
s := scanLine("")
if s == quitCommand {
fmt.Println("Quit command received")
close(done)
break
}
sendMsg([]byte(s))

if *asymmetricMode {
// print your own message for convenience,
// because in asymmetric mode it is impossible to decrypt it
timestamp := time.Now().Unix()
from := crypto.PubkeyToAddress(asymKey.PublicKey)
fmt.Printf("\n%d <%x>: %s\n", timestamp, from, s)
}
}
}

func sendFilesLoop() {
for {
s := scanLine("")
if s == quitCommand {
fmt.Println("Quit command received")
close(done)
break
```

```
}
b, err := ioutil.ReadFile(s)
if err != nil {
fmt.Printf(">>> Error: %s \n", err)
continue
} else {
h := sendMsg(b)
if (h == common.Hash{}) {
fmt.Printf(">>> Error: message was not sent \n")
} else {
timestamp := time.Now().Unix()
from := crypto.PubkeyToAddress(asymKey.PublicKey)
fmt.Printf("\n%d <%x>: sent message with hash %x\n", timestamp, from, h)
}
}
}
}

func scanLine(prompt string) string {
if len(prompt) > 0 {
fmt.Print(prompt)
}
txt, err := input.ReadString('\n')
if err != nil {
utils.Fatalf("input error: %s", err)
}
txt = strings.TrimRight(txt, "\n\r")
return txt
}

func scanLineA(prompt string) *string {
s := scanLine(prompt)
return &s
}

func scanUint(prompt string) uint32 {
s := scanLine(prompt)
i, err := strconv.Atoi(s)
if err != nil {
utils.Fatalf("Fail to parse the lower time limit: %s", err)
}
return uint32(i)
```

```
}

func sendMsg(payload []byte) common.Hash {
params := whisper.MessageParams{
Src:      asymKey,
Dst:      pub,
KeySym:   symKey,
Payload:  payload,
Topic:    topic,
TTL:      uint32(*argTTL),
PoW:      *argPoW,
WorkTime: uint32(*argWorkTime),
}

msg, err := whisper.NewSentMessage(&params)
if err != nil {
utils.Fatalf("failed to create new message: %s", err)
}
envelope, err := msg.Wrap(&params)
if err != nil {
fmt.Printf("failed to seal message: %v \n", err)
return common.Hash{}
}

err = shh.Send(envelope)
if err != nil {
fmt.Printf("failed to send message: %v \n", err)
return common.Hash{}
}

return envelope.Hash()
}

func messageLoop() {
f := shh.GetFilter(filterID)
if f == nil {
utils.Fatalf("filter is not installed")
}

ticker := time.NewTicker(time.Millisecond * 50)

for {
```

```
select {
case <-ticker.C:
messages := f.Retrieve()
for _, msg := range messages {
if *fileExMode || len(msg.Payload) > 2048 {
writeMessageToFile(*argSaveDir, msg)
} else {
printMessageInfo(msg)
}
}
case <-done:
return
}
}
}

func printMessageInfo(msg *whisper.ReceivedMessage) {
timestamp := fmt.Sprintf("%d", msg.Sent) // unix timestamp for diagnostics
text := string(msg.Payload)

var address common.Address
if msg.Src != nil {
address = crypto.PubkeyToAddress(*msg.Src)
}

if whisper.IsPubKeyEqual(msg.Src, &asymKey.PublicKey) {
fmt.Printf("\n%s <%x>: %s\n", timestamp, address, text) // message from myself
} else {
fmt.Printf("\n%s [%x]: %s\n", timestamp, address, text) // message from a peer
}
}

func writeMessageToFile(dir string, msg *whisper.ReceivedMessage) {
timestamp := fmt.Sprintf("%d", msg.Sent)
name := fmt.Sprintf("%x", msg.EnvelopeHash)

var address common.Address
if msg.Src != nil {
address = crypto.PubkeyToAddress(*msg.Src)
}

if whisper.IsPubKeyEqual(msg.Src, &asymKey.PublicKey) {
```

```
// message from myself: don't save, only report
fmt.Printf("\n%s <%x>: message received: '%s'\n", timestamp, address, name)
} else if len(dir) > 0 {
fullpath := filepath.Join(dir, name)
err := ioutil.WriteFile(fullpath, msg.Payload, 0644)
if err != nil {
fmt.Printf("\n%s {%x}: message received but not saved: %s\n", timestamp, address, err)
} else {
fmt.Printf("\n%s {%x}: message received and saved as '%s' (%d bytes)\n", timestamp, address,
name, len(msg.Payload))
}
} else {
fmt.Printf("\n%s {%x}: big message received (%d bytes), but not saved: %s\n", timestamp,
address, len(msg.Payload), name)
}
}

func requestExpiredMessagesLoop() {
var key, peerID []byte
var timeLow, timeUpp uint32
var t string
var xt, empty whisper.TopicType

keyID, err := shh.AddSymKeyFromPassword(msPassword)
if err != nil {
utils.Fatalf("Failed to create symmetric key for mail request: %s", err)
}
key, err = shh.GetSymKey(keyID)
if err != nil {
utils.Fatalf("Failed to save symmetric key for mail request: %s", err)
}
peerID = extractIdFromEnode(*argEnode)
shh.AllowP2PMessagesFromPeer(peerID)

for {
timeLow = scanUint("Please enter the lower limit of the time range (unix timestamp): ")
timeUpp = scanUint("Please enter the upper limit of the time range (unix timestamp): ")
t = scanLine("Please enter the topic (hexadecimal): ")
if len(t) >= whisper.TopicLength*2 {
x, err := hex.DecodeString(t)
if err != nil {
utils.Fatalf("Failed to parse the topic: %s", err)
```

```
}
xt = whisper.BytesToTopic(x)
}
if timeUpp == 0 {
timeUpp = 0xFFFFFFFF
}

data := make([]byte, 8+whisper.TopicLength)
binary.BigEndian.PutUint32(data, timeLow)
binary.BigEndian.PutUint32(data[4:], timeUpp)
copy(data[8:], xt[:])
if xt == empty {
data = data[:8]
}

var params whisper.MessageParams
params.PoW = *argServerPoW
params.Payload = data
params.KeySym = key
params.Src = nodeid
params.WorkTime = 5

msg, err := whisper.NewSentMessage(&params)
if err != nil {
utils.Fatalf("failed to create new message: %s", err)
}
env, err := msg.Wrap(&params)
if err != nil {
utils.Fatalf("Wrap failed: %s", err)
}

err = shh.RequestHistoricMessages(peerID, env)
if err != nil {
utils.Fatalf("Failed to send P2P message: %s", err)
}

time.Sleep(time.Second * 5)
}
}

func extractIdFromEnode(s string) []byte {
n, err := discover.ParseNode(s)
```

```
if err != nil {
utils.Fatalf("Failed to parse enode: %s", err)
}
return n.ID[:]
}
```

109:F:\git\coin\ethereum\go-ethereum\common\big.go

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

package common

import "math/big"

// Common big integers often used
var (
Big1   = big.NewInt(1)
Big2   = big.NewInt(2)
Big3   = big.NewInt(3)
Big0   = big.NewInt(0)
Big32  = big.NewInt(32)
Big256 = big.NewInt(0xff)
Big257 = big.NewInt(257)
)
```

110:F:\git\coin\ethereum\go-ethereum\common\bitutil\bitutil.go

```
const wordSize = int(unsafe.Sizeof(uintptr(0)))
const supportsUnaligned = runtime.GOARCH == "386" || runtime.GOARCH == "amd64" ||
runtime.GOARCH == "ppc64" || runtime.GOARCH == "ppc64le" || runtime.GOARCH == "s390x"

// XORBytes xors the bytes in a and b. The destination is assumed to have enough
// space. Returns the number of bytes xor'd.
func XORBytes(dst, a, b []byte) int {
if supportsUnaligned {
return fastXORBytes(dst, a, b)
}
return safeXORBytes(dst, a, b)
}

// fastXORBytes xors in bulk. It only works on architectures that support
// unaligned read/writes.
func fastXORBytes(dst, a, b []byte) int {
n := len(a)
```

```
if len(b) < n {
n = len(b)
}
w := n / wordSize
if w > 0 {
dw := *(*[]uintptr)(unsafe.Pointer(&dst))
aw := *(*[]uintptr)(unsafe.Pointer(&a))
bw := *(*[]uintptr)(unsafe.Pointer(&b))
for i := 0; i < w; i++ {
dw[i] = aw[i] ^ bw[i]
}
}
for i := (n - n%wordSize); i < n; i++ {
dst[i] = a[i] ^ b[i]
}
return n
}

// safeXORBytes xors one by one. It works on all architectures, independent if
// it supports unaligned read/writes or not.
func safeXORBytes(dst, a, b []byte) int {
n := len(a)
if len(b) < n {
n = len(b)
}
for i := 0; i < n; i++ {
dst[i] = a[i] ^ b[i]
}
return n
}

// ANDBytes ands the bytes in a and b. The destination is assumed to have enough
// space. Returns the number of bytes and'd.
func ANDBytes(dst, a, b []byte) int {
if supportsUnaligned {
return fastANDBytes(dst, a, b)
}
return safeANDBytes(dst, a, b)
}

// fastANDBytes ands in bulk. It only works on architectures that support
// unaligned read/writes.
```

```
func fastANDBytes(dst, a, b []byte) int {
n := len(a)
if len(b) < n {
n = len(b)
}
w := n / wordSize
if w > 0 {
dw := *(*[]uintptr)(unsafe.Pointer(&dst))
aw := *(*[]uintptr)(unsafe.Pointer(&a))
bw := *(*[]uintptr)(unsafe.Pointer(&b))
for i := 0; i < w; i++ {
dw[i] = aw[i] & bw[i]
}
}
for i := (n - n%wordSize); i < n; i++ {
dst[i] = a[i] & b[i]
}
return n
}

// safeANDBytes ands one by one. It works on all architectures, independent if
// it supports unaligned read/writes or not.
func safeANDBytes(dst, a, b []byte) int {
n := len(a)
if len(b) < n {
n = len(b)
}
for i := 0; i < n; i++ {
dst[i] = a[i] & b[i]
}
return n
}

// ORBytes ors the bytes in a and b. The destination is assumed to have enough
// space. Returns the number of bytes or'd.
func ORBytes(dst, a, b []byte) int {
if supportsUnaligned {
return fastORBytes(dst, a, b)
}
return safeORBytes(dst, a, b)
}
```

```
// fastORBytes ors in bulk. It only works on architectures that support
// unaligned read/writes.
func fastORBytes(dst, a, b []byte) int {
n := len(a)
if len(b) < n {
n = len(b)
}
w := n / wordSize
if w > 0 {
dw := *(*[]uintptr)(unsafe.Pointer(&dst))
aw := *(*[]uintptr)(unsafe.Pointer(&a))
bw := *(*[]uintptr)(unsafe.Pointer(&b))
for i := 0; i < w; i++ {
dw[i] = aw[i] | bw[i]
}
}
for i := (n - n%wordSize); i < n; i++ {
dst[i] = a[i] | b[i]
}
return n
}

// safeORBytes ors one by one. It works on all architectures, independent if
// it supports unaligned read/writes or not.
func safeORBytes(dst, a, b []byte) int {
n := len(a)
if len(b) < n {
n = len(b)
}
for i := 0; i < n; i++ {
dst[i] = a[i] | b[i]
}
return n
}

// TestBytes tests whether any bit is set in the input byte slice.
func TestBytes(p []byte) bool {
if supportsUnaligned {
return fastTestBytes(p)
}
return safeTestBytes(p)
}
```

```
// fastTestBytes tests for set bits in bulk. It only works on architectures that
// support unaligned read/writes.
func fastTestBytes(p []byte) bool {
n := len(p)
w := n / wordSize
if w > 0 {
pw := *(*[]uintptr)(unsafe.Pointer(&p))
for i := 0; i < w; i++ {
if pw[i] != 0 {
return true
}
}
}
for i := (n - n%wordSize); i < n; i++ {
if p[i] != 0 {
return true
}
}
return false
}

// safeTestBytes tests for set bits one byte at a time. It works on all
// architectures, independent if it supports unaligned read/writes or not.
func safeTestBytes(p []byte) bool {
for i := 0; i < len(p); i++ {
if p[i] != 0 {
return true
}
}
return false
}

111:F:\git\coin\ethereum\go-ethereum\common\bitutil\bitutil_test.go
func TestXOR(t *testing.T) {
for alignP := 0; alignP < 2; alignP++ {
for alignQ := 0; alignQ < 2; alignQ++ {
for alignD := 0; alignD < 2; alignD++ {
p := make([]byte, 1023)[alignP:]
q := make([]byte, 1023)[alignQ:]

for i := 0; i < len(p); i++ {
```

```
p[i] = byte(i)
}
for i := 0; i < len(q); i++ {
q[i] = byte(len(q) - i)
}
d1 := make([]byte, 1023+alignD)[alignD:]
d2 := make([]byte, 1023+alignD)[alignD:]

XORBytes(d1, p, q)
safeXORBytes(d2, p, q)
if !bytes.Equal(d1, d2) {
t.Error("not equal", d1, d2)
}
}
}
}
}

// Tests that bitwise AND works for various alignments.
func TestAND(t *testing.T) {
for alignP := 0; alignP < 2; alignP++ {
for alignQ := 0; alignQ < 2; alignQ++ {
for alignD := 0; alignD < 2; alignD++ {
p := make([]byte, 1023)[alignP:]
q := make([]byte, 1023)[alignQ:]

for i := 0; i < len(p); i++ {
p[i] = byte(i)
}
for i := 0; i < len(q); i++ {
q[i] = byte(len(q) - i)
}
d1 := make([]byte, 1023+alignD)[alignD:]
d2 := make([]byte, 1023+alignD)[alignD:]

ANDBytes(d1, p, q)
safeANDBytes(d2, p, q)
if !bytes.Equal(d1, d2) {
t.Error("not equal")
}
}
}
}
```

```
}
}

// Tests that bitwise OR works for various alignments.
func TestOR(t *testing.T) {
for alignP := 0; alignP < 2; alignP++ {
for alignQ := 0; alignQ < 2; alignQ++ {
for alignD := 0; alignD < 2; alignD++ {
p := make([]byte, 1023)[alignP:]
q := make([]byte, 1023)[alignQ:]

for i := 0; i < len(p); i++ {
p[i] = byte(i)
}
for i := 0; i < len(q); i++ {
q[i] = byte(len(q) - i)
}
d1 := make([]byte, 1023+alignD)[alignD:]
d2 := make([]byte, 1023+alignD)[alignD:]

ORBytes(d1, p, q)
safeORBytes(d2, p, q)
if !bytes.Equal(d1, d2) {
t.Error("not equal")
}
}
}
}
}

// Tests that bit testing works for various alignments.
func TestTest(t *testing.T) {
for align := 0; align < 2; align++ {
// Test for bits set in the bulk part
p := make([]byte, 1023)[align:]
p[100] = 1

if TestBytes(p) != safeTestBytes(p) {
t.Error("not equal")
}
// Test for bits set in the tail part
q := make([]byte, 1023)[align:]
```

```
q[len(q)-1] = 1

if TestBytes(q) != safeTestBytes(q) {
t.Error("not equal")
}
}
}

// Benchmarks the potentially optimized XOR performance.
func BenchmarkFastXOR1KB(b *testing.B) { benchmarkFastXOR(b, 1024) }
func BenchmarkFastXOR2KB(b *testing.B) { benchmarkFastXOR(b, 2048) }
func BenchmarkFastXOR4KB(b *testing.B) { benchmarkFastXOR(b, 4096) }

func benchmarkFastXOR(b *testing.B, size int) {
p, q := make([]byte, size), make([]byte, size)

for i := 0; i < b.N; i++ {
XORBytes(p, p, q)
}
}

// Benchmarks the baseline XOR performance.
func BenchmarkBaseXOR1KB(b *testing.B) { benchmarkBaseXOR(b, 1024) }
func BenchmarkBaseXOR2KB(b *testing.B) { benchmarkBaseXOR(b, 2048) }
func BenchmarkBaseXOR4KB(b *testing.B) { benchmarkBaseXOR(b, 4096) }

func benchmarkBaseXOR(b *testing.B, size int) {
p, q := make([]byte, size), make([]byte, size)

for i := 0; i < b.N; i++ {
safeXORBytes(p, p, q)
}
}

// Benchmarks the potentially optimized AND performance.
func BenchmarkFastAND1KB(b *testing.B) { benchmarkFastAND(b, 1024) }
func BenchmarkFastAND2KB(b *testing.B) { benchmarkFastAND(b, 2048) }
func BenchmarkFastAND4KB(b *testing.B) { benchmarkFastAND(b, 4096) }

func benchmarkFastAND(b *testing.B, size int) {
p, q := make([]byte, size), make([]byte, size)
```

```
for i := 0; i < b.N; i++ {
ANDBytes(p, p, q)
}
}

// Benchmarks the baseline AND performance.
func BenchmarkBaseAND1KB(b *testing.B) { benchmarkBaseAND(b, 1024) }
func BenchmarkBaseAND2KB(b *testing.B) { benchmarkBaseAND(b, 2048) }
func BenchmarkBaseAND4KB(b *testing.B) { benchmarkBaseAND(b, 4096) }

func benchmarkBaseAND(b *testing.B, size int) {
p, q := make([]byte, size), make([]byte, size)

for i := 0; i < b.N; i++ {
safeANDBytes(p, p, q)
}
}

// Benchmarks the potentially optimized OR performance.
func BenchmarkFastOR1KB(b *testing.B) { benchmarkFastOR(b, 1024) }
func BenchmarkFastOR2KB(b *testing.B) { benchmarkFastOR(b, 2048) }
func BenchmarkFastOR4KB(b *testing.B) { benchmarkFastOR(b, 4096) }

func benchmarkFastOR(b *testing.B, size int) {
p, q := make([]byte, size), make([]byte, size)

for i := 0; i < b.N; i++ {
ORBytes(p, p, q)
}
}

// Benchmarks the baseline OR performance.
func BenchmarkBaseOR1KB(b *testing.B) { benchmarkBaseOR(b, 1024) }
func BenchmarkBaseOR2KB(b *testing.B) { benchmarkBaseOR(b, 2048) }
func BenchmarkBaseOR4KB(b *testing.B) { benchmarkBaseOR(b, 4096) }

func benchmarkBaseOR(b *testing.B, size int) {
p, q := make([]byte, size), make([]byte, size)

for i := 0; i < b.N; i++ {
safeORBytes(p, p, q)
}
```

```
}

// Benchmarks the potentially optimized bit testing performance.
func BenchmarkFastTest1KB(b *testing.B) { benchmarkFastTest(b, 1024) }
func BenchmarkFastTest2KB(b *testing.B) { benchmarkFastTest(b, 2048) }
func BenchmarkFastTest4KB(b *testing.B) { benchmarkFastTest(b, 4096) }

func benchmarkFastTest(b *testing.B, size int) {
p := make([]byte, size)
for i := 0; i < b.N; i++ {
TestBytes(p)
}
}

// Benchmarks the baseline bit testing performance.
func BenchmarkBaseTest1KB(b *testing.B) { benchmarkBaseTest(b, 1024) }
func BenchmarkBaseTest2KB(b *testing.B) { benchmarkBaseTest(b, 2048) }
func BenchmarkBaseTest4KB(b *testing.B) { benchmarkBaseTest(b, 4096) }

func benchmarkBaseTest(b *testing.B, size int) {
p := make([]byte, size)
for i := 0; i < b.N; i++ {
safeTestBytes(p)
}
}

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

package bitutil

import "errors"

var (
// errMissingData is returned from decompression if the byte referenced by
// the bitset header overflows the input data.
errMissingData = errors.New("missing bytes on input")

// errUnreferencedData is returned from decompression if not all bytes were used
// up from the input data after decompressing it.
errUnreferencedData = errors.New("extra bytes on input")
```

```
// errExceededTarget is returned from decompression if the bitset header has
// more bits defined than the number of target buffer space available.
errExceededTarget = errors.New("target data size exceeded")

// errZeroContent is returned from decompression if a data byte referenced in
// the bitset header is actually a zero byte.
errZeroContent = errors.New("zero byte in input content")
)

// The compression algorithm implemented by CompressBytes and DecompressBytes is
// optimized for sparse input data which contains a lot of zero bytes. Decompression
// requires knowledge of the decompressed data length.
//
// Compression works as follows:
//
//   if data only contains zeroes,
//       CompressBytes(data) == nil
//   otherwise if len(data) <= 1,
//       CompressBytes(data) == data
//   otherwise:
//       CompressBytes(data) == append(CompressBytes(nonZeroBitset(data)),
nonZeroBytes(data)...)
//       where
//         nonZeroBitset(data) is a bit vector with len(data) bits (MSB first):
//             nonZeroBitset(data)[i/8] && (1 << (7-i%8)) != 0  if data[i] != 0
//             len(nonZeroBitset(data)) == (len(data)+7)/8
//         nonZeroBytes(data) contains the non-zero bytes of data in the same order

// CompressBytes compresses the input byte slice according to the sparse bitset
// representation algorithm. If the result is bigger than the original input, no
// compression is done.
func CompressBytes(data []byte) []byte {
if out := bitsetEncodeBytes(data); len(out) < len(data) {
return out
}
cpy := make([]byte, len(data))
copy(cpy, data)
return cpy
}

// bitsetEncodeBytes compresses the input byte slice according to the sparse
// bitset representation algorithm.
```

```
func bitsetEncodeBytes(data []byte) []byte {
// Empty slices get compressed to nil
if len(data) == 0 {
return nil
}
// One byte slices compress to nil or retain the single byte
if len(data) == 1 {
if data[0] == 0 {
return nil
}
return data
}
// Calculate the bitset of set bytes, and gather the non-zero bytes
nonZeroBitset := make([]byte, (len(data)+7)/8)
nonZeroBytes := make([]byte, 0, len(data))

for i, b := range data {
if b != 0 {
nonZeroBytes = append(nonZeroBytes, b)
nonZeroBitset[i/8] |= 1 << byte(7-i%8)
}
}
if len(nonZeroBytes) == 0 {
return nil
}
return append(bitsetEncodeBytes(nonZeroBitset), nonZeroBytes...)
}

// DecompressBytes decompresses data with a known target size. If the input data
// matches the size of the target, it means no compression was done in the first
// place.
func DecompressBytes(data []byte, target int) ([]byte, error) {
if len(data) > target {
return nil, errExceededTarget
}
if len(data) == target {
cpy := make([]byte, len(data))
copy(cpy, data)
return cpy, nil
}
return bitsetDecodeBytes(data, target)
}
```

```
// bitsetDecodeBytes decompresses data with a known target size.
func bitsetDecodeBytes(data []byte, target int) ([]byte, error) {
out, size, err := bitsetDecodePartialBytes(data, target)
if err != nil {
return nil, err
}
if size != len(data) {
return nil, errUnreferencedData
}
return out, nil
}

// bitsetDecodePartialBytes decompresses data with a known target size, but does
// not enforce consuming all the input bytes. In addition to the decompressed
// output, the function returns the length of compressed input data corresponding
// to the output as the input slice may be longer.
func bitsetDecodePartialBytes(data []byte, target int) ([]byte, int, error) {
// Sanity check 0 targets to avoid infinite recursion
if target == 0 {
return nil, 0, nil
}
// Handle the zero and single byte corner cases
decomp := make([]byte, target)
if len(data) == 0 {
return decomp, 0, nil
}
if target == 1 {
decomp[0] = data[0] // copy to avoid referencing the input slice
if data[0] != 0 {
return decomp, 1, nil
}
return decomp, 0, nil
}
// Decompress the bitset of set bytes and distribute the non zero bytes
nonZeroBitset, ptr, err := bitsetDecodePartialBytes(data, (target+7)/8)
if err != nil {
return nil, ptr, err
}
for i := 0; i < 8*len(nonZeroBitset); i++ {
if nonZeroBitset[i/8]&(1<<byte(7-i%8)) != 0 {
// Make sure we have enough data to push into the correct slot
```

```
if ptr >= len(data) {
return nil, 0, errMissingData
}
if i >= len(decomp) {
return nil, 0, errExceededTarget
}
// Make sure the data is valid and push into the slot
if data[ptr] == 0 {
return nil, 0, errZeroContent
}
decomp[i] = data[ptr]
ptr++
}
}
return decomp, ptr, nil
}

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

// +build gofuzz

package bitutil

import "bytes"

// Fuzz implements a go-fuzz fuzzer method to test various encoding method
// invocations.
func Fuzz(data []byte) int {
if len(data) == 0 {
return -1
}
if data[0]%2 == 0 {
return fuzzEncode(data[1:])
}
return fuzzDecode(data[1:])
}

// fuzzEncode implements a go-fuzz fuzzer method to test the bitset encoding and
// decoding algorithm.
func fuzzEncode(data []byte) int {
proc, _ := bitsetDecodeBytes(bitsetEncodeBytes(data), len(data))
```

```
if !bytes.Equal(data, proc) {
panic("content mismatch")
}
return 0
}

// fuzzDecode implements a go-fuzz fuzzer method to test the bit decoding and
// reencoding algorithm.
func fuzzDecode(data []byte) int {
blob, err := bitsetDecodeBytes(data, 1024)
if err != nil {
return 0
}
if comp := bitsetEncodeBytes(blob); !bytes.Equal(comp, data) {
panic("content mismatch")
}
return 0
}

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

package bitutil

import (
"bytes"
"math/rand"
"testing"

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

// Tests that data bitset encoding and decoding works and is bijective.
func TestEncodingCycle(t *testing.T) {
tests := []string{
// Tests generated by go-fuzz to maximize code coverage
"0x0000000000000000",
"0xef0400",
"0xdf70705335343336363136393436383735323135363 46c1bc3333934383731307070633634303
35336336346c65fefb3930393233383838ac2f65fefb",
"0x7b64000000",
"0x000034000000000000",
```

```
"0x000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000f00000000000000000000",
"0x4912385c0e7b64000000",
"0x000034000000000000000000000000000000",
"0x00",
"0x000003e834ff7f0000",
"0x0000",
"0x0000000000000000000000000000000000000000000000000000000000ff00",
"0x895f0c6a020f850c6a020f85f88df88d",
"0xdf7070533534333636313639343638373432313536346c1bc3315aac2f65fefb",
"0x0000000000",
"0xdf70706336346c65fefb",
"0x00006d643634000000",
"0xdf7070533534333636313639343638373532313536346c1bc333393438373130707063363430
353639343638373532313536346c1bc333393438336336346c65fe",
}
for i, tt := range tests {
data := hexutil.MustDecode(tt)

proc, err := bitsetDecodeBytes(bitsetEncodeBytes(data), len(data))
if err != nil {
t.Errorf("test %d: failed to decompress compressed data: %v", i, err)
continue
}
if !bytes.Equal(data, proc) {
t.Errorf("test %d: compress/decompress mismatch: have %x, want %x", i, proc, data)
}
}
}

// Tests that data bitset decoding and rencoding works and is bijective.
func TestDecodingCycle(t *testing.T) {
tests := []struct {
size  int
input string
fail  error
}{
{size: 0, input: "0x"},

// Crashers generated by go-fuzz
{size: 0, input: "0x0020", fail: errUnreferencedData},
{size: 0, input: "0x30", fail: errUnreferencedData},
```

```
{size: 1, input: "0x00", fail: errUnreferencedData},
{size: 2, input: "0x07", fail: errMissingData},
{size: 1024, input: "0x8000", fail: errZeroContent},

// Tests generated by go-fuzz to maximize code coverage
{size: 29490, input:
"0x343137343733323134333839373334323073333930783e3078333930783e70706336346c653
03e", fail: errMissingData},
{size: 59395, input: "0x00", fail: errUnreferencedData},
{size: 52574, input: "0x70706336346c65c0de", fail: errExceededTarget},
{size: 42264, input: "0x07", fail: errMissingData},
{size: 52, input: "0xa5045bad48f4", fail: errExceededTarget},
{size: 52574, input: "0xc0de", fail: errMissingData},
{size: 52574, input: "0x"},
{size: 29490, input:
"0x343137343733323134333839373334323073333930780730343338393733343230733339307
83e307833393733343230733339307807306133393078 3e70706336346c65303e", fail:
errMissingData},
{size: 29491, input: "0x3973333930783e30783e", fail: errMissingData},

{size: 1024, input: "0x808080608080"},
{size: 1024, input:
"0x808470705e3632383337363033313434303137393130306c6580ef46806380635a80"},
{size: 1024, input: "0x8080808070"},
{size: 1024, input: "0x808070705e36346c6580ef46806380635a80"},
{size: 1024, input: "0x80808046802680"},
{size: 1024, input: "0x4040404035"},
{size: 1024, input:
"0x4040bf3ba2b3f684402d353234373438373934409fe5b1e7ada94ebfd7d0505e27be4035"},
{size: 1024, input: "0x404040bf3ba2b3f6844035"},
{size: 1024, input: "0x40402d353234373438373934 40bfd7d0505e27be4035"},
}
for i, tt := range tests {
data := hexutil.MustDecode(tt.input)

orig, err := bitsetDecodeBytes(data, tt.size)
if err != tt.fail {
t.Errorf("test %d: failure mismatch: have %v, want %v", i, err, tt.fail)
}
if err != nil {
continue
}
```

```
if comp := bitsetEncodeBytes(orig); !bytes.Equal(comp, data) {
t.Errorf("test %d: decompress/compress mismatch: have %x, want %x", i, comp, data)
}
}
}

// TestCompression tests that compression works by returning either the bitset
// encoded input, or the actual input if the bitset version is longer.
func TestCompression(t *testing.T) {
// Check the the compression returns the bitset encoding is shorter
in := hexutil.MustDecode("0x4912385c0e7b64000000")
out := hexutil.MustDecode("0x80fe4912385c0e7b64")

if data := CompressBytes(in); bytes.Compare(data, out) != 0 {
t.Errorf("encoding mismatch for sparse data: have %x, want %x", data, out)
}
if data, err := DecompressBytes(out, len(in)); err != nil || bytes.Compare(data, in) != 0 {
t.Errorf("decoding mismatch for sparse data: have %x, want %x, error %v", data, in, err)
}
// Check the the compression returns the input if the bitset encoding is longer
in =
hexutil.MustDecode("0xdf70705335343336363136393436383735323135363461bc33339343837
313070706336343035336336346c65fefb3930393233383838ac2f65fefb")
out =
hexutil.MustDecode("0xdf70705335343336363136393436383735323135363461bc33339343837
313070706336343035336336346c65fefb3930393233383838ac2f65fefb")

if data := CompressBytes(in); bytes.Compare(data, out) != 0 {
t.Errorf("encoding mismatch for dense data: have %x, want %x", data, out)
}
if data, err := DecompressBytes(out, len(in)); err != nil || bytes.Compare(data, in) != 0 {
t.Errorf("decoding mismatch for dense data: have %x, want %x, error %v", data, in, err)
}
// Check that decompressing a longer input than the target fails
if _, err := DecompressBytes([]byte{0xc0, 0x01, 0x01}, 2); err != errExceededTarget {
t.Errorf("decoding error mismatch for long data: have %v, want %v", err, errExceededTarget)
}
}

// Crude benchmark for compressing random slices of bytes.
func BenchmarkEncoding1KBVerySparse(b *testing.B) { benchmarkEncoding(b, 1024, 0.0001) }
func BenchmarkEncoding2KBVerySparse(b *testing.B) { benchmarkEncoding(b, 2048, 0.0001) }
```

```
func BenchmarkEncoding4KBVerySparse(b *testing.B) { benchmarkEncoding(b, 4096, 0.0001) }

func BenchmarkEncoding1KBSparse(b *testing.B) { benchmarkEncoding(b, 1024, 0.001) }
func BenchmarkEncoding2KBSparse(b *testing.B) { benchmarkEncoding(b, 2048, 0.001) }
func BenchmarkEncoding4KBSparse(b *testing.B) { benchmarkEncoding(b, 4096, 0.001) }

func BenchmarkEncoding1KBDense(b *testing.B) { benchmarkEncoding(b, 1024, 0.1) }
func BenchmarkEncoding2KBDense(b *testing.B) { benchmarkEncoding(b, 2048, 0.1) }
func BenchmarkEncoding4KBDense(b *testing.B) { benchmarkEncoding(b, 4096, 0.1) }

func BenchmarkEncoding1KBSaturated(b *testing.B) { benchmarkEncoding(b, 1024, 0.5) }
func BenchmarkEncoding2KBSaturated(b *testing.B) { benchmarkEncoding(b, 2048, 0.5) }
func BenchmarkEncoding4KBSaturated(b *testing.B) { benchmarkEncoding(b, 4096, 0.5) }

func benchmarkEncoding(b *testing.B, bytes int, fill float64) {
// Generate a random slice of bytes to compress
random := rand.NewSource(0) // reproducible and comparable

data := make([]byte, bytes)
bits := int(float64(bytes) * 8 * fill)

for i := 0; i < bits; i++ {
idx := random.Int63() % int64(len(data))
bit := uint(random.Int63() % 8)
data[idx] |= 1 << bit
}
// Reset the benchmark and measure encoding/decoding
b.ResetTimer()
b.ReportAllocs()
for i := 0; i < b.N; i++ {
bitsetDecodeBytes(bitsetEncodeBytes(data), len(data))
}
}

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

// Package common contains various helper functions.
package common

import (
"encoding/hex"
```

```
)

func ToHex(b []byte) string {
hex := Bytes2Hex(b)
// Prefer output of "0x0" instead of "0x"
if len(hex) == 0 {
hex = "0"
}
return "0x" + hex
}

func FromHex(s string) []byte {
if len(s) > 1 {
if s[0:2] == "0x" || s[0:2] == "0X" {
s = s[2:]
}
if len(s)%2 == 1 {
s = "0" + s
}
return Hex2Bytes(s)
}
return nil
}

// Copy bytes
//
// Returns an exact copy of the provided bytes
func CopyBytes(b []byte) (copiedBytes []byte) {
copiedBytes = make([]byte, len(b))
copy(copiedBytes, b)

return
}

func HasHexPrefix(str string) bool {
l := len(str)
return l >= 2 && str[0:2] == "0x"
}

func IsHex(str string) bool {
l := len(str)
return l >= 4 && l%2 == 0 && str[0:2] == "0x"
```

```
}

func Bytes2Hex(d []byte) string {
return hex.EncodeToString(d)
}

func Hex2Bytes(str string) []byte {
h, _ := hex.DecodeString(str)

return h
}

func Hex2BytesFixed(str string, flen int) []byte {
h, _ := hex.DecodeString(str)
if len(h) == flen {
return h
} else {
if len(h) > flen {
return h[len(h)-flen:]
} else {
hh := make([]byte, flen)
copy(hh[flen-len(h):flen], h[:])
return hh
}
}
}

func RightPadBytes(slice []byte, l int) []byte {
if l <= len(slice) {
return slice
}

padded := make([]byte, l)
copy(padded, slice)

return padded
}

func LeftPadBytes(slice []byte, l int) []byte {
if l <= len(slice) {
return slice
}
```

```
padded := make([]byte, l)
copy(padded[l-len(slice):], slice)

return padded
}

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

package common

import (
"bytes"
"testing"

checker "gopkg.in/check.v1"
)

type BytesSuite struct{}

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

func (s *BytesSuite) TestCopyBytes(c *checker.C) {
data1 := []byte{1, 2, 3, 4}
exp1 := []byte{1, 2, 3, 4}
res1 := CopyBytes(data1)
c.Assert(res1, checker.DeepEquals, exp1)
}

func (s *BytesSuite) TestIsHex(c *checker.C) {
data1 := "a9e67e"
exp1 := false
res1 := IsHex(data1)
c.Assert(res1, checker.DeepEquals, exp1)

data2 := "0xa9e67e00"
exp2 := true
res2 := IsHex(data2)
c.Assert(res2, checker.DeepEquals, exp2)

}
```

```
func (s *BytesSuite) TestLeftPadBytes(c *checker.C) {
val1 := []byte{1, 2, 3, 4}
exp1 := []byte{0, 0, 0, 0, 1, 2, 3, 4}

res1 := LeftPadBytes(val1, 8)
res2 := LeftPadBytes(val1, 2)

c.Assert(res1, checker.DeepEquals, exp1)
c.Assert(res2, checker.DeepEquals, val1)
}

func (s *BytesSuite) TestRightPadBytes(c *checker.C) {
val := []byte{1, 2, 3, 4}
exp := []byte{1, 2, 3, 4, 0, 0, 0, 0}

resstd := RightPadBytes(val, 8)
resshrt := RightPadBytes(val, 2)

c.Assert(resstd, checker.DeepEquals, exp)
c.Assert(resshrt, checker.DeepEquals, val)
}

func TestFromHex(t *testing.T) {
input := "0x01"
expected := []byte{1}
result := FromHex(input)
if !bytes.Equal(expected, result) {
t.Errorf("Expected % x got % x", expected, result)
}
}

func TestFromHexOddLength(t *testing.T) {
input := "0x1"
expected := []byte{1}
result := FromHex(input)
if !bytes.Equal(expected, result) {
t.Errorf("Expected % x got % x", expected, result)
}
}

117:F:\git\coin\ethereum\go-ethereum\common\compiler\solidity.go
```

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

// Package compiler wraps the Solidity compiler executable (solc).
package compiler

import (
"bytes"
"encoding/json"
"errors"
"fmt"
"io/ioutil"
"os/exec"
"regexp"
"strconv"
"strings"
)

var versionRegexp = regexp.MustCompile(`([0-9]+)\.([0-9]+)\.([0-9]+)`)

type Contract struct {
Code string      `json:"code"`
Info ContractInfo `json:"info"`
}

type ContractInfo struct {
Source          string      `json:"source"`
Language        string      `json:"language"`
LanguageVersion string      `json:"languageVersion"`
CompilerVersion string      `json:"compilerVersion"`
CompilerOptions string      `json:"compilerOptions"`
AbiDefinition   interface{} `json:"abiDefinition"`
UserDoc         interface{} `json:"userDoc"`
DeveloperDoc    interface{} `json:"developerDoc"`
Metadata        string      `json:"metadata"`
}

// Solidity contains information about the solidity compiler.
type Solidity struct {
Path, Version, FullVersion string
Major, Minor, Patch        int
}
```

```
// --combined-output format
type solcOutput struct {
Contracts map[string]struct {
Bin, Abi, Devdoc, Userdoc, Metadata string
}
Version string
}

func (s *Solidity) makeArgs() []string {
p := []string{
"--combined-json", "bin,abi,userdoc,devdoc",
"--add-std",  // include standard lib contracts
"--optimize", // code optimizer switched on
}
if s.Major > 0 || s.Minor > 4 || s.Patch > 6 {
p[1] += ",metadata"
}
return p
}

// SolidityVersion runs solc and parses its version output.
func SolidityVersion(solc string) (*Solidity, error) {
if solc == "" {
solc = "solc"
}
var out bytes.Buffer
cmd := exec.Command(solc, "--version")
cmd.Stdout = &out
err := cmd.Run()
if err != nil {
return nil, err
}
matches := versionRegexp.FindStringSubmatch(out.String())
if len(matches) != 4 {
return nil, fmt.Errorf("can't parse solc version %q", out.String())
}
s := &Solidity{Path: cmd.Path, FullVersion: out.String(), Version: matches[0]}
if s.Major, err = strconv.Atoi(matches[1]); err != nil {
return nil, err
}
if s.Minor, err = strconv.Atoi(matches[2]); err != nil {
return nil, err
```

```
}
if s.Patch, err = strconv.Atoi(matches[3]); err != nil {
return nil, err
}
return s, nil
}

// CompileSolidityString builds and returns all the contracts contained within a source string.
func CompileSolidityString(solc, source string) (map[string]*Contract, error) {
if len(source) == 0 {
return nil, errors.New("solc: empty source string")
}
s, err := SolidityVersion(solc)
if err != nil {
return nil, err
}
args := append(s.makeArgs(), "--")
cmd := exec.Command(s.Path, append(args, "-")...)
cmd.Stdin = strings.NewReader(source)
return s.run(cmd, source)
}

// CompileSolidity compiles all given Solidity source files.
func CompileSolidity(solc string, sourcefiles ...string) (map[string]*Contract, error) {
if len(sourcefiles) == 0 {
return nil, errors.New("solc: no source files")
}
source, err := slurpFiles(sourcefiles)
if err != nil {
return nil, err
}
s, err := SolidityVersion(solc)
if err != nil {
return nil, err
}
args := append(s.makeArgs(), "--")
cmd := exec.Command(s.Path, append(args, sourcefiles...)...)
return s.run(cmd, source)
}

func (s *Solidity) run(cmd *exec.Cmd, source string) (map[string]*Contract, error) {
var stderr, stdout bytes.Buffer
```

```
cmd.Stderr = &stderr
cmd.Stdout = &stdout
if err := cmd.Run(); err != nil {
return nil, fmt.Errorf("solc: %v\n%s", err, stderr.Bytes())
}
var output solcOutput
if err := json.Unmarshal(stdout.Bytes(), &output); err != nil {
return nil, err
}

// Compilation succeeded, assemble and return the contracts.
contracts := make(map[string]*Contract)
for name, info := range output.Contracts {
// Parse the individual compilation results.
var abi interface{}
if err := json.Unmarshal([]byte(info.Abi), &abi); err != nil {
return nil, fmt.Errorf("solc: error reading abi definition (%v)", err)
}
var userdoc interface{}
if err := json.Unmarshal([]byte(info.Userdoc), &userdoc); err != nil {
return nil, fmt.Errorf("solc: error reading user doc: %v", err)
}
var devdoc interface{}
if err := json.Unmarshal([]byte(info.Devdoc), &devdoc); err != nil {
return nil, fmt.Errorf("solc: error reading dev doc: %v", err)
}
contracts[name] = &Contract{
Code: "0x" + info.Bin,
Info: ContractInfo{
Source:          source,
Language:        "Solidity",
LanguageVersion: s.Version,
CompilerVersion: s.Version,
CompilerOptions: strings.Join(s.makeArgs(), " "),
AbiDefinition:   abi,
UserDoc:         userdoc,
DeveloperDoc:    devdoc,
Metadata:        info.Metadata,
},
}
}
return contracts, nil
```

```
}

func slurpFiles(files []string) (string, error) {
var concat bytes.Buffer
for _, file := range files {
content, err := ioutil.ReadFile(file)
if err != nil {
return "", err
}
concat.Write(content)
}
return concat.String(), nil
}

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

package compiler

import (
"os/exec"
"testing"
)

const (
testSource = `
contract test {
   /// @notice Will multiply ` + "`a`" + ` by 7.
   function multiply(uint a) returns(uint d) {
       return a * 7;
   }
}
`
)

func skipWithoutSolc(t *testing.T) {
if _, err := exec.LookPath("solc"); err != nil {
t.Skip(err)
}
}

func TestCompiler(t *testing.T) {
```

```
skipWithoutSolc(t)

contracts, err := CompileSolidityString("", testSource)
if err != nil {
t.Fatalf("error compiling source. result %v: %v", contracts, err)
}
if len(contracts) != 1 {
t.Errorf("one contract expected, got %d", len(contracts))
}
c, ok := contracts["test"]
if !ok {
c, ok = contracts["<stdin>:test"]
if !ok {
t.Fatal("info for contract 'test' not present in result")
}
}
if c.Code == "" {
t.Error("empty code")
}
if c.Info.Source != testSource {
t.Error("wrong source")
}
if c.Info.CompilerVersion == "" {
t.Error("empty version")
}
}

func TestCompileError(t *testing.T) {
skipWithoutSolc(t)

contracts, err := CompileSolidityString("", testSource[4:])
if err == nil {
t.Errorf("error expected compiling source. got none. result %v", contracts)
}
t.Logf("error: %v", err)
}

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

package common
```

```
import (
"fmt"
"os"
"runtime"
"runtime/debug"
"strings"
)

// Report gives off a warning requesting the user to submit an issue to the github tracker.
func Report(extra ...interface{}) {
fmt.Fprintln(os.Stderr, "You've encountered a sought after, hard to reproduce bug. Please report this to the developers <3 https://github.com/ethereum/go-ethereum/issues")
fmt.Fprintln(os.Stderr, extra...)

_, file, line, _ := runtime.Caller(1)
fmt.Fprintf(os.Stderr, "%v:%v\n", file, line)

debug.PrintStack()

fmt.Fprintln(os.Stderr, "#### BUG! PLEASE REPORT ####")
}

// PrintDeprecationWarning prinst the given string in a box using fmt.Println.
func PrintDeprecationWarning(str string) {
line := strings.Repeat("#", len(str)+4)
emptyLine := strings.Repeat(" ", len(str))
fmt.Printf(`
%s
# %s #
# %s #
# %s #
%s

`, line, emptyLine, str, emptyLine, line)
}
```