```
//package hogehoge.com;

import java.math.BigInteger;

public class BigNumber {
//Long.MAX_VALUE: 9223 37203 68547 75807
// 999 99999 80000 00001
//Integer.MAX_VALUE: 21474 83647


//long型に収まる場合
private long lval;

//long型に収まらない場合はword配列に格納（1つの配列には10桁）
public int words[];

//桁数を格納
private int length;

//trueなら負数
private boolean isnegative;

//定数
//private final int longTypelen=9;
private final int intTypelen=9;
private final int longTypelen=18;
private final int intMax=999999999;
public static final BigNumber ZERO=new BigNumber("0");
public static final BigNumber ONE=new BigNumber("1");
public static final BigNumber TWO=new BigNumber("2");
public static final BigNumber MinusONE=new BigNumber("-1");
private static final int[] k =
{100,150,200,250,300,350,400,500,600,800,1250,Integer.MAX_VALUE};
```


```
//package hogehoge.com;

import java.math.BigInteger;

public class BigNumber {
//Long.MAX_VALUE: 9223 37203 68547 75807
// 999 99999 80000 00001
//Integer.MAX_VALUE: 21474 83647


//long型に収まる場合
private long lval;

//long型に収まらない場合はword配列に格納（1つの配列には10桁）
public int words[];

//桁数を格納
private int length;

//trueなら負数
private boolean isnegative;

//定数
//private final int longTypelen=9;
private final int intTypelen=9;
private final int longTypelen=18;
private final int intMax=999999999;
public static final BigNumber ZERO=new BigNumber("0");
public static final BigNumber ONE=new BigNumber("1");
public static final BigNumber TWO=new BigNumber("2");
public static final BigNumber MinusONE=new BigNumber("-1");
private static final int[] k =
{100,150,200,250,300,350,400,500,600,800,1250,Integer.MAX_VALUE};
```

```
private static final int[] t =
{27,18,15,12,9,8,7,6,5,4,3,2};
private static int[] primes;
/*
{2,3,5,7,11,13,17,19,23,29,31,37,41,43,
47,53,59,61,67,71,73,79,83,89,97,101,103,107,
109,113,127,131,137,139,149,151,157,163,167,173,179,181,
191,193,197,199,211,223,227,229,233,239,241,251};
*/

//素数マスクの作成
private static final int maskNum=8;
public static boolean[] primeMask;
public static int maskLen;
public static BigNumber maskLenBN;

private static final int minFixNum = -100;
private static final int maxFixNum = 1024;
private static final int numFixNum = maxFixNum-minFixNum+1;
private static final BigNumber[] smallFixNums = new
BigNumber[numFixNum];
static
{
for (int i = numFixNum; --i >= 0; )

smallFixNums[i] = new BigNumber(i + minFixNum);

}

public static void main(String[] args) {
// TODO Auto-generated method stub
//String str= "12345678901234567890123456789";
String str= "11215515555555555555555555555555555534";
```

```
private static final int[] t =
{27,18,15,12,9,8,7,6,5,4,3,2};
private static int[] primes;
/*
{2,3,5,7,11,13,17,19,23,29,31,37,41,43,
47,53,59,61,67,71,73,79,83,89,97,101,103,107,
109,113,127,131,137,139,149,151,157,163,167,173,179,181,
191,193,197,199,211,223,227,229,233,239,241,251};
*/

//素数マスクの作成
private static final int maskNum=8;
public static boolean[] primeMask;
public static int maskLen;
public static BigInteger maskLenBN;

private static final int minFixNum = -100;
private static final int maxFixNum = 1024;
private static final int numFixNum = maxFixNum-minFixNum+1;
private static final BigInteger[] smallFixNums = new
BigInteger[numFixNum];
static
{
for (int i = numFixNum; --i >= 0; )
smallFixNums[i] = new BigInteger(Integer.toString(i +
minFixNum));
}

public static void main(String[] args) {
// TODO Auto-generated method stub
//String str= "12345678901234567890123456789";
String str= "11215515555555555555555555555555555534";
```

```
String str2=
"99930048965723993204756670330768462076546417507744578
79484360947336864195063455581093234735685020485660037
97341860697511741780753472569201558908763865294293994
20405715998800839412411312081542912 13";

BigNumber.initPrimeList();

makePrimeMask();

BigNumber isp=new BigNumber(str2);

//System.out.println("check prime : " + isp);

//System.out.println("result : " + isp.isProbablePrime(50));

String str3= str2;


//String str3=
"99993004896572399320475667033076846207654641750774457
87948436094733686419506345558109323473568502048566003
79734186069751132399300489657239932047566703307684620
76546417507744578794843609473368641950634555810932347
35685020485660037973418606975113239999300489657239932
04756670330768462076546417507744578794843609473368641
95063455581093234735685020485660037973418606975113239
93004896572399320475667033076846207654641750774457879
48436094733686419506345558109323473568502048566003797
34186069751132399 3";

//String str3= "97";

//String str2= "-12345678987654321";


BigNumber num=new BigNumber(str3);

BigInteger numi=new BigInteger(str3);

long startTime = System.currentTimeMillis();

System.out.println("BI isPrime: " + numi.isProbablePrime(50));

long stopTime = System.currentTimeMillis();

System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );

startTime = System.currentTimeMillis();
```


```
String str2=
"99930048965723993204756670330768462076546417507744578
79484360947336864195063455581093234735685020485660037
97341860697511741780753472569201558908763865294293994
20405715998800839412411312081542912 13";

BigNumber.initPrimeList();

makePrimeMask();

BigNumber isp=new BigNumber(str2);

//System.out.println("check prime : " + isp);

//System.out.println("result : " + isp.isProbablePrime(50));

String str3= str2;


//String str3=
"99993004896572399320475667033076846207654641750774457
87948436094733686419506345558109323473568502048566003
79734186069751132399300489657239932047566703307684620
76546417507744578794843609473368641950634555810932347
35685020485660037973418606975113239999300489657239932
04756670330768462076546417507744578794843609473368641
95063455581093234735685020485660037973418606975113239
93004896572399320475667033076846207654641750774457879
48436094733686419506345558109323473568502048566003797
34186069751132399 3";

//String str3= "97";

//String str2= "-12345678987654321";


BigNumber num=new BigNumber(str3);

BigInteger numi=new BigInteger(str3);

long startTime = System.currentTimeMillis();

System.out.println("BI isPrime: " + numi.isProbablePrime(50));

long stopTime = System.currentTimeMillis();

System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );

startTime = System.currentTimeMillis();
```

```
System.out.println("BN isPrime: " + num.isProbablePrime(50));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

BigNumber num2=new BigNumber(str);
BigInteger numi2=new BigInteger(str);
System.out.println(num);
BigNumber num3=new BigNumber("3");
BigNumber num97=new BigNumber("97");
//System.out.println("comp:" + TWO.compareTo(TWO));

//System.out.println("2^3 mod 97:" + TWO.modPow(num3, num97));
//実行時間計測
startTime = System.currentTimeMillis();
stopTime = System.currentTimeMillis();

System.out.println(num +"," +num2);
System.out.println("BN aub: " + num.subtract(num2));

startTime = System.currentTimeMillis();
for (int i=0;i<1000;i ++)
numi.divide(numi2);
System.out.println("BI divide: " + numi.divide(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<1000;i ++)
num.divide(num2);
System.out.println("BN divide: " + num.divide(num2));
stopTime = System.currentTimeMillis();
```

```
//System.out.println("BN isPrime: " + num.isProbablePrime(50));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

BigNumber num2=new BigNumber(str);
BigInteger numi2=new BigInteger(str);
System.out.println(num);
BigNumber num3=new BigNumber("3");
BigNumber num97=new BigNumber("97");
//System.out.println("comp:" + TWO.compareTo(TWO));

//System.out.println("2^3 mod 97:" + TWO.modPow(num3, num97));
//実行時間計測
startTime = System.currentTimeMillis();
stopTime = System.currentTimeMillis();

System.out.println(num +"," +num2);
System.out.println("BN aub: " + num.subtract(num2));

startTime = System.currentTimeMillis();
for (int i=0;i<1000;i ++)
numi.divide(numi2);
System.out.println("BI divide: " + numi.divide(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<1000;i ++)
num.divide(num2);
System.out.println("BN divide: " + num.divide(num2));
stopTime = System.currentTimeMillis();
```

```
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
numi.multiply(numi2);
System.out.println("BI times: " + numi.multiply(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
num.multiply(num2);
System.out.println("BN times: " + num.multiply(num2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
numi.add(numi2);
System.out.println("BI add: " + numi.add(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
num.add(num2);
System.out.println("BN add: " + num.add(num2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
```

```
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
numi.multiply(numi2);
System.out.println("BI times: " + numi.multiply(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
num.multiply(num2);
System.out.println("BN times: " + num.multiply(num2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
numi.add(numi2);
System.out.println("BI add: " + numi.add(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
num.add(num2);
System.out.println("BN add: " + num.add(num2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
```

```
" );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
numi.subtract(numi2);
System.out.println("BI sub: " + numi.subtract(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
num.subtract(num2);
System.out.println("BN sub: " + num.subtract(num2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );


/*
System.out.println("BI getLowestBitSet: " +
numi.getLowestSetBit());
long stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );

startTime = System.currentTimeMillis();*/
System.out.println("BN getBitLen: " + num.bitLength());
System.out.println("BI getBitLen: " + numi.bitLength());
System.out.println("BN getLowestBitSet: " +
num.add(MinusONE).getLowestSetBit());
System.out.println("BI getLowestBitSet: " +
numi.add(BigInteger.ONE.negate()).getLowestSetBit());
/*
stopTime = System.currentTimeMillis();
```


```
" );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
numi.subtract(numi2);
System.out.println("BI sub: " + numi.subtract(numi2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );

startTime = System.currentTimeMillis();
for (int i=0;i<10000;i ++)
num.subtract(num2);
System.out.println("BN sub: " + num.subtract(num2));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );


/*
System.out.println("BI getLowestBitSet: " +
numi.getLowestSetBit());
long stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms
" );

startTime = System.currentTimeMillis();*/
System.out.println("BN getBitLen: " + num.bitLength());
System.out.println("BI getBitLen: " + numi.bitLength());
System.out.println("BN getLowestBitSet: " +
num.add(MinusONE).getLowestSetBit());
System.out.println("BI getLowestBitSet: " +
numi.add(BigInteger.ONE.negate()).getLowestSetBit());
/*
stopTime = System.currentTimeMillis();
```

```
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
System.out.println("BI getBitLen: " + numi.bitLength());
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
System.out.println("BN getBitLen: " + num.bitLength());
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );


*/

startTime = System.currentTimeMillis();
//System.out.println("BI isPrime: " + numi.isProbablePrime(50));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
//System.out.println("BN isPrime: " + num.isProbablePrime(50));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

/*
BigNumber num=new BigNumber(str);
BigInteger numi=new BigInteger(str);
System.out.println(num);
BigNumber num2=new BigNumber(str2);
```


```
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
System.out.println("BI getBitLen: " + numi.bitLength());
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
System.out.println("BN getBitLen: " + num.bitLength());
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );


*/

startTime = System.currentTimeMillis();
//System.out.println("BI isPrime: " + numi.isProbablePrime(50));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

startTime = System.currentTimeMillis();
//System.out.println("BN isPrime: " + num.isProbablePrime(50));
stopTime = System.currentTimeMillis();
System.out.println(" Run Time = " + (stopTime - startTime) + " ms " );

/*
BigNumber num=new BigNumber(str);
BigInteger numi=new BigInteger(str);
System.out.println(num);
BigNumber num2=new BigNumber(str2);
```

```
BigInteger numi2=new BigInteger(str2);
System.out.println(num2);

System.out.println(num.add(num2));
System.out.println(numi.add(numi2));
System.out.println(num.subtract(num2));
System.out.println(numi.subtract(numi2));
System.out.println(num.multiply(num2));
System.out.println(numi.multiply(numi2));
System.out.println(numi.divide(numi2));
System.out.println(num.divide(num2));
System.out.println(numi.remainder(numi2));
System.out.println(num.remainder(num2));
*/

}

public static void initPrimeList(){
//54番目までの素数リストを作成
primes=S1GP.sosuList(300, 54);
}

public static void makePrimeMask(){

long startTime = System.currentTimeMillis();
maskLen=1;
for(int i=0;i<maskNum;i++) maskLen *= primes[i];
System.out.println("MaskLen=" + maskLen);
maskLenBN = new BigNumber(maskLen);
//マスクの初期化
primeMask = new boolean[maskLen];

for(int i=0;i<primeMask.length;i++){
```

```
BigInteger numi2=new BigInteger(str2);
System.out.println(num2);

System.out.println(num.add(num2));
System.out.println(numi.add(numi2));
System.out.println(num.subtract(num2));
System.out.println(numi.subtract(numi2));
System.out.println(num.multiply(num2));
System.out.println(numi.multiply(numi2));
System.out.println(numi.divide(numi2));
System.out.println(num.divide(num2));
System.out.println(numi.remainder(numi2));
System.out.println(num.remainder(num2));
*/

}

public static void initPrimeList(){
//54番目までの素数リストを作成
primes=S1GP.sosuList(300, 54);
}

public static void makePrimeMask(){

long startTime = System.currentTimeMillis();
maskLen=1;
for(int i=0;i<maskNum;i++) maskLen *= primes[i];
System.out.println("MaskLen=" + maskLen);
maskLenBN = new BigInteger(Integer.toString(maskLen));
//マスクの初期化
primeMask = new boolean[maskLen];

for(int i=0;i<primeMask.length;i++){
```

```
primeMask[i]=false;
}

int index=0;
//マスクのセット
for(int i=0;i<maskNum;i++){
int j=1;
index=j*primes[i];
while( index < maskLen){
primeMask[index]=true;
j++;
index=j*primes[i];
}

}
System.out.println(" Run Time(makePrimeMask)= " +
(System.currentTimeMillis() - startTime) + " ms " );

}

public BigNumber(){

}

public BigNumber(String s){
int len=s.length();
int start;

//桁数、正負の設定
if(s.charAt(0)=='-'){
isnegative=true;
length=len-1;
start=1;
}else{
```

```
primeMask[i]=false;
}

int index=0;
//マスクのセット
for(int i=0;i<maskNum;i++){
int j=1;
index=j*primes[i];
while( index < maskLen){
primeMask[index]=true;
j++;
index=j*primes[i];
}

}
System.out.println(" Run Time(makePrimeMask)= " +
(System.currentTimeMillis() - startTime) + " ms " );

}

public BigNumber(){

}

public BigNumber(String s){
int len=s.length();
int start;

//桁数、正負の設定
if(s.charAt(0)=='-'){
isnegative=true;
length=len-1;
start=1;
}else{
```

```
isnegative=false;
length=len;
start=0;
}

while(s.charAt(start)=='0'){
start++;
if(start==len){
length=1;
words=new int[1];
words[0]=0;
isnegative=false;
return;
}
length--;
}

//long型に収まる桁数ならlong型で保持
/*if(len-start <= 18){
lval=(long)Math.abs(Long.parseLong(s));
words=null;
return;
}*/

//long型に収まらない場合は10ケタずつに区切って保持
int countLong=(int)Math.ceil((len-start)/(double)intTypelen);
words=new int[countLong];
for(int i=0; i<words.length; i++){
int startIndex=( len - intTypelen * (i+1)-start<0 ? start : len - intTypelen*(i+1) );
int endIndex=( len - intTypelen*i );
words[i]=Integer.parseInt(s.substring(startIndex, endIndex));
```


```
isnegative=false;
length=len;
start=0;
}

while(s.charAt(start)=='0'){
start++;
if(start==len){
length=1;
words=new int[1];
words[0]=0;
isnegative=false;
return;
}
length--;
}

//long型に収まる桁数ならlong型で保持
/*if(len-start <= 18){
lval=(long)Math.abs(Long.parseLong(s));
words=null;
return;
}*/

//long型に収まらない場合は10ケタずつに区切って保持
int countLong=(int)Math.ceil((len-start)/(double)intTypelen);
words=new int[countLong];
for(int i=0; i<words.length; i++){
int startIndex=( len - intTypelen * (i+1)-start<0 ? start : len - intTypelen*(i+1) );
int endIndex=( len - intTypelen*i );
words[i]=Integer.parseInt(s.substring(startIndex, endIndex));
```

```
}
}

public BigNumber(long l){
new BigNumber(Long.toString(l));
}

public BigNumber(int i){
i = i<0 ? i * (-1) : i ;
int len = Integer.toString(i).length();

length = i<0 ? len -1 : len ;
isnegative = i<0 ? true : false ;

words=new int[1];
words[0]= i<0 ? i * (-1) : i ;
isnegative=false;
}

//正負を逆にする
public BigNumber negate(){
if((this.length==1) & (this.words[0]==0)){
return new BigNumber("0");
}
BigNumber newnum = new BigNumber();
newnum.words=this.words;
newnum.length=this.length;
newnum.isnegative = !this.isnegative;

return newnum;
}

public boolean isZero(){
```

```
}
}

public BigNumber(long l){
new BigNumber(Long.toString(l));
}

public BigNumber(int i){
i = i<0 ? i * (-1) : i ;
int len = Integer.toString(i).length();

length = i<0 ? len -1 : len ;
isnegative = i<0 ? true : false ;

words=new int[1];
words[0]= i<0 ? i * (-1) : i ;
isnegative=false;
}

//正負を逆にする
public BigNumber negate(){
if((this.length==1) & (this.words[0]==0)){
return new BigNumber("0");
}
BigNumber newnum = new BigNumber();
newnum.words=this.words;
newnum.length=this.length;
newnum.isnegative = !this.isnegative;

return newnum;
}

public boolean isZero(){
```

```
if((this.length==1) & (this.words[0]==0)){
return true;
}
return false;
}

public boolean isOne(){
if((this.length==1) & (this.words[0]==1)){
return true;
}
return false;

}

public BigNumber copy(){
BigNumber newnum = new BigNumber();
newnum.words=new int[this.words.length];
for(int i=0;i<newnum.words.length;i++)
newnum.words[i]=this.words[i];
newnum.length=this.length;
newnum.isnegative = this.isnegative;
return newnum;
}

//絶対値を返す
public BigNumber abs(){
if((this.length==1) & (this.words[0]==0)){
return new BigNumber("0");
}

BigNumber newnum = new BigNumber();
newnum.words=this.words;
newnum.length=this.length;
```


```
if((this.length==1) & (this.words[0]==0)){
return true;
}
return false;
}

public boolean isOne(){
if((this.length==1) & (this.words[0]==1)){
return true;
}
return false;

}

public BigNumber copy(){
BigNumber newnum = new BigNumber();
newnum.words=new int[this.words.length];
for(int i=0;i<newnum.words.length;i++)
newnum.words[i]=this.words[i];
newnum.length=this.length;
newnum.isnegative = this.isnegative;
return newnum;
}

//絶対値を返す
public BigNumber abs(){
if((this.length==1) & (this.words[0]==0)){
return new BigNumber("0");
}

BigNumber newnum = new BigNumber();
newnum.words=this.words;
newnum.length=this.length;
```

```
newnum.isnegative = ( this.isnegative ? false : false );

return newnum;
}

//加算する
public BigNumber add(BigNumber num){

//どちらかがゼロの場合
if(this.isZero())
return num.copy();
if(num.isZero())

return this.copy();

if( this.isnegative ^ num.isnegative){
if(num.isnegative){
return this.subtract(num.negate());
}else{
return num.subtract(this.negate());
}
}

BigNumber newnum = new BigNumber();

if(num.isOne()){
if(words[0] != intMax){
newnum=this.copy();
newnum.words[0]++;
return newnum;
}
}

if(this.words.length==1 && num.words.length==1){
long templong = (long)(this.words[0] + num.words[0]);
```

```
newnum.isnegative = ( this.isnegative ? false : false );

return newnum;
}

//加算する
public BigNumber add(BigNumber num){

//どちらかがゼロの場合
if(this.isZero())
return num.copy();
if(num.isZero())

return this.copy();

if( this.isnegative ^ num.isnegative){
if(num.isnegative){
return this.subtract(num.negate());
}else{
return num.subtract(this.negate());
}
}

BigNumber newnum = new BigNumber();

if(num.isOne()){
if(words[0] != intMax){
newnum=this.copy();
newnum.words[0]++;
return newnum;
}
}

if(this.words.length==1 && num.words.length==1){
long templong = (long)(this.words[0] + num.words[0]);
```

```
if(templong<=(long)intMax){
newnum=this.copy();
newnum.words[0]=(int)templong;
return newnum;
}else{
newnum=this.copy();
newnum.words[0]=(int)templong-intMax-1;
newnum.words[1]=1;
newnum.length=10;
return newnum;
}
}

int thisCount = (int)Math.ceil(this.length/(double)intTypelen);
int numCount = (int)Math.ceil(num.length/(double)intTypelen);
//int resultLen = (this.length >= num.length ? this.length :
num.length);
int countInt= (thisCount >= numCount ? thisCount : numCount );
newnum.words=new int[countInt];

int thisTemp;
int numTemp;
int incDigit=0;

for(int i=0;i<countInt;i++){
thisTemp = ( i<thisCount ? this.words[i] : 0);
numTemp = ( i<numCount ? num.words[i] : 0);
newnum.words[i]=thisTemp+numTemp+incDigit;
if(newnum.words[i]>intMax){
newnum.words[i]=newnum.words[i]-(intMax+1);
incDigit=1;
}else{
```

```
if(templong<=(long)intMax){
newnum=this.copy();
newnum.words[0]=(int)templong;
return newnum;
}else{
newnum=this.copy();
newnum.words[0]=(int)templong-intMax-1;
newnum.words[1]=1;
newnum.length=10;
return newnum;
}
}

int thisCount = (int)Math.ceil(this.length/(double)intTypelen);
int numCount = (int)Math.ceil(num.length/(double)intTypelen);
//int resultLen = (this.length >= num.length ? this.length :
num.length);
int countInt= (thisCount >= numCount ? thisCount : numCount );
newnum.words=new int[countInt];

int thisTemp;
int numTemp;
int incDigit=0;

for(int i=0;i<countInt;i++){
thisTemp = ( i<thisCount ? this.words[i] : 0);
numTemp = ( i<numCount ? num.words[i] : 0);
newnum.words[i]=thisTemp+numTemp+incDigit;
if(newnum.words[i]>intMax){
newnum.words[i]=newnum.words[i]-(intMax+1);
incDigit=1;
}else{
```

```
incDigit=0;
}
}

newnum.length=(this.length >= num.length ? this.length :
num.length);
newnum.isnegative=this.isnegative;

if(incDigit==1){
int tempwords[]=new int[countInt];
tempwords=newnum.words;
newnum.words=new int[countInt+1];
for(int i=0;i<tempwords.length;i++){
newnum.words[i]=tempwords[i];
}
newnum.words[countInt]=1;
newnum.length+=1;
}

newnum.length=newnum.getWordsRaw().length();

return newnum;
}

//減算する
public BigNumber subtract(BigNumber num){

//どちらかがゼロの場合
if(this.isZero())
return num.negate();
if(num.isZero())
return this.copy();

//正負のパターンチェック
```


```
incDigit=0;
}
}

newnum.length=(this.length >= num.length ? this.length :
num.length);
newnum.isnegative=this.isnegative;

if(incDigit==1){
int tempwords[]=new int[countInt];
tempwords=newnum.words;
newnum.words=new int[countInt+1];
for(int i=0;i<tempwords.length;i++){
newnum.words[i]=tempwords[i];
}
newnum.words[countInt]=1;
newnum.length+=1;
}

newnum.length=newnum.getWordsRaw().length();

return newnum;
}

//減算する
public BigNumber subtract(BigNumber num){

//どちらかがゼロの場合
if(this.isZero())
return num.negate();
if(num.isZero())
return this.copy();

//正負のパターンチェック
```

```
if( (!this.isnegative) && num.isnegative){
return this.add(num.negate());
}else if( this.isnegative && !(num.isnegative)){
return this.add(num.negate());
}else if ( this.isnegative && num.isnegative){
return (num.negate()).subtract(this.negate());
}

//大小比較
int comp=this.compareTo(num);
//comp=comp+0;
if(comp==0){
return new BigNumber("0");
}else if(comp<0){
return num.subtract(this).negate();
}

//
BigNumber newnum = new BigNumber();

if(num.isOne()){
if(words[0] != 0){
newnum=this.copy();
newnum.words[0]--;
return newnum;
}
}

if(this.words.length==1 && num.words.length==1){
long templong = this.words[0] - this.words[0];
if(templong<=(long)intMax){
newnum=this.copy();
```


```
if( (!this.isnegative) && num.isnegative){
return this.add(num.negate());
}else if( this.isnegative && !(num.isnegative)){
return this.add(num.negate());
}else if ( this.isnegative && num.isnegative){
return (num.negate()).subtract(this.negate());
}

//大小比較
int comp=this.compareTo(num);
//comp=comp+0;
if(comp==0){
return new BigNumber("0");
}else if(comp<0){
return num.subtract(this).negate();
}

//
BigNumber newnum = new BigNumber();

if(num.isOne()){
if(words[0] != 0){
newnum=this.copy();
newnum.words[0]--;
return newnum;
}
}

if(this.words.length==1 && num.words.length==1){
long templong = this.words[0] - this.words[0];
if(templong<=(long)intMax){
newnum=this.copy();
```

```
newnum.words[0]=this.words[0]-num.words[0];
return newnum;

}
}

int thisCount = (int)Math.ceil(this.length/(double)intTypelen);
int numCount = (int)Math.ceil(num.length/(double)intTypelen);
int countInt= (thisCount >= numCount ? thisCount : numCount );
newnum.words=new int[countInt];

int thisTemp;
int numTemp;
int decDigit=0;

for(int i=0;i<countInt-1;i++){
thisTemp = ( i<thisCount ? this.words[i] : 0);
numTemp = ( i<numCount ? num.words[i] : 0);
newnum.words[i]=thisTemp-numTemp-decDigit;
if(newnum.words[i]<0){
newnum.words[i]=newnum.words[i]+(intMax+1);
decDigit=1;
}else{
decDigit=0;
}
}

thisTemp = ( countInt-1<thisCount ? this.words[countInt-1] : 0);
numTemp = ( countInt-1<numCount ? num.words[countInt-1] : 0);
newnum.words[countInt-1]=thisTemp-numTemp-decDigit;

if(newnum.words[countInt-1]<0){
newnum.isnegative=true;
```


```
newnum.words[0]=this.words[0]-num.words[0];
return newnum;

}
}

int thisCount = (int)Math.ceil(this.length/(double)intTypelen);
int numCount = (int)Math.ceil(num.length/(double)intTypelen);
int countInt= (thisCount >= numCount ? thisCount : numCount );
newnum.words=new int[countInt];

int thisTemp;
int numTemp;
int decDigit=0;

for(int i=0;i<countInt-1;i++){
thisTemp = ( i<thisCount ? this.words[i] : 0);
numTemp = ( i<numCount ? num.words[i] : 0);
newnum.words[i]=thisTemp-numTemp-decDigit;
if(newnum.words[i]<0){
newnum.words[i]=newnum.words[i]+(intMax+1);
decDigit=1;
}else{
decDigit=0;
}
}

thisTemp = ( countInt-1<thisCount ? this.words[countInt-1] : 0);
numTemp = ( countInt-1<numCount ? num.words[countInt-1] : 0);
newnum.words[countInt-1]=thisTemp-numTemp-decDigit;

if(newnum.words[countInt-1]<0){
newnum.isnegative=true;
```

```
newnum.words[countInt-1]=newnum.words[countInt-1]*(-1);
}else{

newnum.isnegative=false;
}

int newCountInt=countInt;

for(int i=countInt-1;i>=0;i--){
if(newnum.words[i]==0){
newCountInt--;
}else{
break;
}
}

int tempwords[]=new int[countInt];
tempwords=newnum.words;
newnum.words=new int[newCountInt];
for(int i=0;i<newCountInt;i++){
newnum.words[i]=tempwords[i];
}

newnum.length=newnum.getWordsRaw().length();

return newnum;
}

//積を求める
public BigNumber multiply(BigNumber num){

if(this.isZero() || num.isZero()){
return new BigNumber("0");
}
```

```
newnum.words[countInt-1]=newnum.words[countInt-1]*(-1);
}else{

newnum.isnegative=false;
}

int newCountInt=countInt;

for(int i=countInt-1;i>=0;i--){
if(newnum.words[i]==0){
newCountInt--;
}else{
break;
}
}

int tempwords[]=new int[countInt];
tempwords=newnum.words;
newnum.words=new int[newCountInt];
for(int i=0;i<newCountInt;i++){
newnum.words[i]=tempwords[i];
}

newnum.length=newnum.getWordsRaw().length();

return newnum;
}

//積を求める
public BigNumber multiply(BigNumber num){

if(this.isZero() || num.isZero()){
return new BigNumber("0");
}
```

```
BigNumber newnum = new BigNumber();
int tempx=this.words.length;
int tempy=num.words.length;
long matrix[][]=new long[tempx][tempy];
long incDigit=0;
long templ;
newnum.words=new int[tempx+tempy];

for(int y=0;y<tempy;y++){
for(int x=0;x<tempx;x++){
incDigit=0;
matrix[x][y]=(long)(this.words[x])*(long)(num.words[y]);
templ = (long)(newnum.words[x+y]) + matrix[x][y];
if(templ>intMax){
incDigit = templ / (intMax+1);
newnum.words[x+y] = (int)(templ - incDigit * (intMax+1));
if(newnum.words[x+y+1] + incDigit > intMax){
templ = newnum.words[x+y+1] + incDigit;
incDigit = templ / (intMax+1);
newnum.words[x+y+1] = (int)(templ - incDigit * (intMax+1));
newnum.words[x+y+2] += incDigit;
}else{
newnum.words[x+y+1] += incDigit;
}
}else{
newnum.words[x+y] = (int)templ;
}
}
}

if(newnum.words[tempx+tempy-1]==0){
```


```
BigNumber newnum = new BigNumber();
int tempx=this.words.length;
int tempy=num.words.length;
long matrix[][]=new long[tempx][tempy];
long incDigit=0;
long templ;
newnum.words=new int[tempx+tempy];

for(int y=0;y<tempy;y++){
for(int x=0;x<tempx;x++){
incDigit=0;
matrix[x][y]=(long)(this.words[x])*(long)(num.words[y]);
templ = (long)(newnum.words[x+y]) + matrix[x][y];
if(templ>intMax){
incDigit = templ / (intMax+1);
newnum.words[x+y] = (int)(templ - incDigit * (intMax+1));
if(newnum.words[x+y+1] + incDigit > intMax){
templ = newnum.words[x+y+1] + incDigit;
incDigit = templ / (intMax+1);
newnum.words[x+y+1] = (int)(templ - incDigit * (intMax+1));
newnum.words[x+y+2] += incDigit;
}else{
newnum.words[x+y+1] += incDigit;
}
}else{
newnum.words[x+y] = (int)templ;
}
}
}

if(newnum.words[tempx+tempy-1]==0){
```

```
int tempwords[]=new int[tempx+tempy];
tempwords=newnum.words;
newnum.words=new int[tempx+tempy-1];
for(int i=0;i<tempx+tempy-1;i++){
newnum.words[i]=tempwords[i];
}
}

newnum.length=newnum.getWordsRaw().length();
newnum.isnegative = ( this.isnegative ^ num.isnegative );

return newnum;
}

//商除を求める(小数点以下は切り下げる)
public BigNumber[] divideandremainder(BigNumber num){

BigNumber[] temp= new BigNumber[2];

//ゼロの場合
if(this.isZero()){
temp[0]=ZERO;
temp[1]=ZERO;
return temp;
}
if(num.isZero()) //ゼロ除算の場合は例外を返す
throw new ArithmeticException("ZERO Division");

if(this.words.length==1 && num.words.length==1 &&
!this.isnegative && !num.isnegative){
int resdiv = this.words[0] / num.words[0];
int resrem = this.words[0] % num.words[0];

BigNumber newdiv=this.copy();
```

```
int tempwords[]=new int[tempx+tempy];
tempwords=newnum.words;
newnum.words=new int[tempx+tempy-1];
for(int i=0;i<tempx+tempy-1;i++){
newnum.words[i]=tempwords[i];
}
}

newnum.length=newnum.getWordsRaw().length();
newnum.isnegative = ( this.isnegative ^ num.isnegative );

return newnum;
}

//商除を求める(小数点以下は切り下げる)
public BigNumber[] divideandremainder(BigNumber num){

BigNumber[] temp= new BigNumber[2];

//ゼロの場合
if(this.isZero()){
temp[0]=ZERO;
temp[1]=ZERO;
return temp;
}
if(num.isZero()) //ゼロ除算の場合は例外を返す
throw new ArithmeticException("ZERO Division");

if(this.words.length==1 && num.words.length==1 &&
!this.isnegative && !num.isnegative){
int resdiv = this.words[0] / num.words[0];
int resrem = this.words[0] % num.words[0];

BigNumber newdiv=this.copy();
```

```
newdiv.words[0]=resdiv;
newdiv.length=String.valueOf(resdiv).length();
BigNumber newrem=this.copy();
newrem.words[0]=resrem;
newrem.length=String.valueOf(resrem).length();
temp[0]=newdiv;
temp[1]=newrem;
return temp;

}

BigNumber tempthis= this.abs();
BigNumber tempnum= num.abs();

//割る数の方が大きい場合はゼロを返す
if(tempthis.compareTo(tempnum) < 0 ){
temp[0]=ZERO;
temp[1]=new BigNumber(this.getWordsRaw());
temp[1].isnegative=this.isnegative;
return temp;
}

StringBuffer buf = new StringBuffer();

//割る数で桁数固定
BigNumber tempnumx= new BigNumber();
BigNumber tempnummul= new BigNumber();
BigNumber tempnumrem= new BigNumber();
int tempx;
int tempy;
int tempdiv;
boolean existSol=false;
```

```
newdiv.words[0]=resdiv;
newdiv.length=String.valueOf(resdiv).length();
BigNumber newrem=this.copy();
newrem.words[0]=resrem;
newrem.length=String.valueOf(resrem).length();
temp[0]=newdiv;
temp[1]=newrem;
return temp;

}

BigNumber tempthis= this.abs();
BigNumber tempnum= num.abs();

//割る数の方が大きい場合はゼロを返す
if(tempthis.compareTo(tempnum) < 0 ){
temp[0]=ZERO;
temp[1]=new BigNumber(this.getWordsRaw());
temp[1].isnegative=this.isnegative;
return temp;
}

StringBuffer buf = new StringBuffer();

//割る数で桁数固定
BigNumber tempnumx= new BigNumber();
BigNumber tempnummul= new BigNumber();
BigNumber tempnumrem= new BigNumber();
int tempx;
int tempy;
int tempdiv;
boolean existSol=false;
```

```
String tempxstr=tempthis.getWordsRaw();
String tempystr=tempnum.getWordsRaw();

if(tempnum.length > intTypelen-1){
tempx = Integer.parseInt(tempxstr.substring(0, intTypelen-1));
}else{
tempx = Integer.parseInt(tempxstr.substring(0, tempnum.length));
}

if(tempnum.length > intTypelen-1){
tempy = Integer.parseInt(tempystr.substring(0, intTypelen-1));
}else{
tempy = Integer.parseInt(tempystr);
}
tempdiv = tempx/tempy; //仮の商

BigNumber[] tempmulti=new BigNumber[11];

for(int i=0;i<11;i++){
tempmulti[i] = tempnum.multiply(new BigNumber(i));
}

//tempnumx = new BigNumber(tempxstr.substring(0, num.length-1));
tempnumx = new BigNumber(tempxstr.substring(0, tempnum.length));

//for(int i=0;i<= this.length - num.length +1 ;i++){
for(int i=0;i<= tempthis.length - tempnum.length ;i++){

//tempnummul = tempnum.multiply(new BigNumber(Integer.toString(tempdiv)));
tempnummul = ( tempdiv <= 10 ? tempmulti[tempdiv] : tempnum.multiply(new BigNumber(Integer.toString(tempdiv))));
tempnumrem = tempnumx.subtract(tempnummul);
```

```
String tempxstr=tempthis.getWordsRaw();
String tempystr=tempnum.getWordsRaw();

if(tempnum.length > intTypelen-1){
tempx = Integer.parseInt(tempxstr.substring(0, intTypelen-1));
}else{
tempx = Integer.parseInt(tempxstr.substring(0, tempnum.length));
}

if(tempnum.length > intTypelen-1){
tempy = Integer.parseInt(tempystr.substring(0, intTypelen-1));
}else{
tempy = Integer.parseInt(tempystr);
}
tempdiv = tempx/tempy; //仮の商

//tempnumx = new BigNumber(tempxstr.substring(0, num.length-1));
tempnumx = new BigNumber(tempxstr.substring(0, tempnum.length));

//for(int i=0;i<= this.length - num.length +1 ;i++){
for(int i=0;i<= tempthis.length - tempnum.length ;i++){

tempnummul = tempnum.multiply(new BigNumber(Integer.toString(tempdiv)));

tempnumrem = tempnumx.subtract(tempnummul);
```

```
loopWhile:
while(tempdiv>0){

if(tempnumx.compareTo(tempnummul) < 0){
tempdiv--;
//tempnummul = tempnum.multiply(new
BigNumber(Integer.toString(tempdiv)));
tempnummul = ( tempdiv <= 10 ? tempmulti[tempdiv]
: tempnum.multiply(new BigNumber(Integer.toString(tempdiv))));
tempnumrem = tempnumx.subtract(tempnummul);
continue loopWhile;
}
if(tempnumrem.compareTo(ZERO) < 0){
tempdiv++;
//tempnummul = tempnum.multiply(new
BigNumber(Integer.toString(tempdiv)));
tempnummul = ( tempdiv <= 10 ? tempmulti[tempdiv]
: tempnum.multiply(new BigNumber(Integer.toString(tempdiv))));
tempnumrem = tempnumx.subtract(tempnummul);
continue loopWhile;
}
break loopWhile;
}

if(tempdiv!=0)
existSol=true;

//商を確定（1桁分）
if(existSol)
buf.append(tempdiv);



//if(i==this.length - num.length+1)
```

```
loopWhile:
while(tempdiv>0){

if(tempnumx.compareTo(tempnummul) < 0){
tempdiv--;
tempnummul = tempnum.multiply(new
BigNumber(Integer.toString(tempdiv)));


tempnumrem = tempnumx.subtract(tempnummul);
continue loopWhile;
}
if(tempnumrem.compareTo(ZERO) < 0){
tempdiv++;
tempnummul = tempnum.multiply(new
BigNumber(Integer.toString(tempdiv)));


tempnumrem = tempnumx.subtract(tempnummul);
continue loopWhile;
}
break loopWhile;
}

if(tempdiv!=0)
existSol=true;

//商を確定（1桁分）
if(existSol)
buf.append(tempdiv);

//if(i==this.length - num.length+1)
```

```
if(i==tempthis.length - tempnum.length)
break;

tempnumx = new BigNumber(tempnumrem.getWordsRaw() +
tempxstr.substring(i+tempnum.length, i+tempnum.length+1));
// tempxstr.substring(i+num.length-1, i+num.length));

if(tempnumx.length > intTypelen){
tempx = Integer.parseInt(tempnumx.getWordsRaw().substring(0,
intTypelen));
}else{
tempx = Integer.parseInt(tempnumx.getWordsRaw());
}
tempdiv = tempx/tempy; //仮の商


}

BigNumber newnum = new BigNumber(buf.toString());
newnum.isnegative = ( this.isnegative ^ num.isnegative );

temp[0]=newnum;
if(this.isnegative & !num.isnegative){
temp[1]=tempnumrem.negate();
}else if(this.isnegative & num.isnegative){
temp[1]=tempnumrem.negate();
}else if(!this.isnegative & num.isnegative){
temp[1]=tempnumrem;
}else{
temp[1]=tempnumrem;
}

return temp;
}
```

```
if(i==tempthis.length - tempnum.length)
break;

tempnumx = new BigNumber(tempnumrem.getWordsRaw() +
tempxstr.substring(i+tempnum.length, i+tempnum.length+1));
// tempxstr.substring(i+num.length-1, i+num.length));

if(tempnumx.length > intTypelen){
tempx = Integer.parseInt(tempnumx.getWordsRaw().substring(0,
intTypelen));
}else{
tempx = Integer.parseInt(tempnumx.getWordsRaw());
}
tempdiv = tempx/tempy; //仮の商
}


BigNumber newnum = new BigNumber(buf.toString());
newnum.isnegative = ( this.isnegative ^ num.isnegative );

temp[0]=newnum;
if(this.isnegative & !num.isnegative){
temp[1]=tempnumrem.negate();
}else if(this.isnegative & num.isnegative){
temp[1]=tempnumrem.negate();
}else if(!this.isnegative & num.isnegative){
temp[1]=tempnumrem;
}else{
temp[1]=tempnumrem;
}

return temp;
}
```

```
public BigNumber divide(BigNumber num){
return this.divideandremainder(num)[0];
}

public BigNumber remainder(BigNumber num){
return this.divideandremainder(num)[1];
}

//2つの数値を比較する
public int compareTo(BigNumber num){

//片方が負数の場合
if( this.isnegative ^ num.isnegative ){
return (this.isnegative ? -1 : 1 );
}

//桁数が違う場合
if(this.length>num.length){
return (this.isnegative ? -1 : 1 );
}else if(this.length<num.length){
return (this.isnegative ? 1 : -1 );

//桁数が同じ場合
}else{
for(int i=this.words.length-1;i>=0;i--){
if(this.words[i]>num.words[i]){
return (this.isnegative ? -1 : 1 );
}else if(this.words[i]<num.words[i]){
return (this.isnegative ? 1 : -1 );
}
}
}
```


```
public BigNumber divide(BigNumber num){
return this.divideandremainder(num)[0];
}

public BigNumber remainder(BigNumber num){
return this.divideandremainder(num)[1];
}

//2つの数値を比較する
public int compareTo(BigNumber num){

//片方が負数の場合
if( this.isnegative ^ num.isnegative ){
return (this.isnegative ? -1 : 1 );
}

//桁数が違う場合
if(this.length>num.length){
return (this.isnegative ? -1 : 1 );
}else if(this.length<num.length){
return (this.isnegative ? 1 : -1 );

//桁数が同じ場合
}else{
for(int i=this.words.length-1;i>=0;i--){
if(this.words[i]>num.words[i]){
return (this.isnegative ? -1 : 1 );
}else if(this.words[i]<num.words[i]){
return (this.isnegative ? 1 : -1 );
}
}
}
```

```
return 0;
}

//数値部分のみを返す（符号なし）
public String getWordsRaw(){
String temp;
StringBuffer buf = new StringBuffer();
buf.append(Integer.toString(words[words.length-1]));
for(int i=words.length-2; i>=0; i--){
temp=Integer.toString(words[i]);
for(int j=0;j<intTypelen-temp.length();j++){
buf.append("0");
}
buf.append(temp);
}
return buf.toString();
}

public String toString(){

String temp;
StringBuffer buf = new StringBuffer();

if(words==null){
buf.append(Long.toString((long)Math.abs(lval)));
}else{
buf.append(Integer.toString(words[words.length-1]));

for(int i=words.length-2; i>=0; i--){
temp=Integer.toString(words[i]);
for(int j=0;j<intTypelen-temp.length();j++){
buf.append("0");
}
```

```
return 0;
}

//数値部分のみを返す（符号なし）
public String getWordsRaw(){
String temp;
StringBuffer buf = new StringBuffer();
buf.append(Integer.toString(words[words.length-1]));
for(int i=words.length-2; i>=0; i--){
temp=Integer.toString(words[i]);
for(int j=0;j<intTypelen-temp.length();j++){
buf.append("0");
}
buf.append(temp);
}
return buf.toString();
}

public String toString(){

String temp;
StringBuffer buf = new StringBuffer();

if(words==null){
buf.append(Long.toString((long)Math.abs(lval)));
}else{
buf.append(Integer.toString(words[words.length-1]));

for(int i=words.length-2; i>=0; i--){
temp=Integer.toString(words[i]);
for(int j=0;j<intTypelen-temp.length();j++){
buf.append("0");
}
```

```
buf.append(temp);
}

}

temp=buf.toString();
buf = new StringBuffer();
int len=temp.length();
if(isnegative) buf.append("-");
//int start=( isnegative ? 1 : 0 );

for (int i = 0; i <len-1; i++) {
buf.append(temp.substring(i,i+1));
if((len-i-1)%5==0){
buf.append(" ");
}
}
buf.append(temp.substring(len-1,len));
buf.append("(" + length + ")");

return buf.toString();
}

public boolean isProbablePrime(int cert){
int i;
//素数の簡易チェック
for (i=0;i<primes.length;i++){
if(this.words.length == 1 && words[0] == primes[i])
return true;

if(this.remainder(smallFixNums[primes[i]-minFixNum]).isZero())

return false;
```

```
buf.append(temp);
}

}

temp=buf.toString();
buf = new StringBuffer();
int len=temp.length();
if(isnegative) buf.append("-");
//int start=( isnegative ? 1 : 0 );

for (int i = 0; i <len-1; i++) {
buf.append(temp.substring(i,i+1));
if((len-i-1)%5==0){
buf.append(" ");
}
}
buf.append(temp.substring(len-1,len));
buf.append("(" + length + ")");

return buf.toString();
}

public static boolean isProbablePrime(BigInteger bint,int cert){
int i;
//素数の簡易チェック
for (i=0;i<primes.length;i++){
if(bint.compareTo(smallFixNums[primes[i]-minFixNum])==0)
return true;
if(bint.remainder(smallFixNums[primes[i]-
minFixNum]).compareTo(BigInteger.ZERO)==0)
return false;
```

```
}

//ラビン・ミラー法で判定
BigNumber pMinus1=this.add(MinusONE);
int b=pMinus1.getLowestSetBit();
//System.out.println( (2L << b - 1) );
BigNumber divtemp = new BigNumber(Long.toString(2L << b - 1));
BigNumber m = pMinus1.divide(divtemp);

int bits=this.bitLength();

for(i=0;i<k.length;i++)
if(bits <= k[i])
break;
int trials=t[i];

if(cert>80)
trials *= 2;

BigNumber z;

for(int t=0;t<trials;t++){
z=smallFixNums[primes[t]-minFixNum].modPow(m,this);

if(z.isOne() || z.compareTo(pMinus1)==0)

continue;

for(i=0;i<b;){
if(z.isOne())
return false;
i++;
if( z.compareTo(pMinus1)==0)
break;
z=z.modPow(TWO,this);
```

```
}

//ラビン・ミラー法で判定
BigInteger pMinus1=bint.add(BigInteger.ONE.negate());
int b=pMinus1.getLowestSetBit();
//System.out.println( (2L << b - 1) );
BigInteger divtemp = new BigInteger(Long.toString(2L << b - 1));
BigInteger m = pMinus1.divide(divtemp);

int bits=bint.bitLength();

for(i=0;i<k.length;i++)
if(bits <= k[i])
break;
int trials=t[i];

if(cert>80)
trials *= 2;

BigInteger z;

for(int t=0;t<trials;t++){
z=smallFixNums[primes[t]-minFixNum].modPow(m,bint);
if(z.compareTo(BigInteger.ONE)==0 ||
z.compareTo(pMinus1)==0)
continue;

for(i=0;i<b;){
if(z.compareTo(BigInteger.ONE)==0)
return false;
i++;
if( z.compareTo(pMinus1)==0)
break;
z=z.modPow(smallFixNums[2-minFixNum],bint);
```

```
}
if(i==b && !(z.compareTo(pMinus1)==0))
return false;
}
return true;
}

public BigNumber modPow(BigNumber exp,BigNumber m){
if(m.isnegative || m.isZero())
throw new ArithmeticException("non positive modulo");

if(exp.isnegative)
throw new ArithmeticException("non positive exp");

if(exp.isOne())
return mod(m);

BigNumber s=ONE;
BigNumber t=this;
BigNumber u =exp;

while(!u.isZero()){
//if(u.and(ONE).isOne){
if(u.words[0] % 2 == 1){
s=s.multiply(t).mod(m);
}
//u=u.shiftRight(1);
u=u.divide(TWO);
t=t.multiply(t).mod(m);
}

return s;
}
```

```
}
if(i==b && !(z.compareTo(pMinus1)==0))
return false;
}
return true;
}

public BigNumber modPow(BigNumber exp,BigNumber m){
if(m.isnegative || m.isZero())
throw new ArithmeticException("non positive modulo");

if(exp.isnegative)
throw new ArithmeticException("non positive exp");

if(exp.isOne())
return mod(m);

BigNumber s=ONE;
BigNumber t=this;
BigNumber u =exp;

while(!u.isZero()){
//if(u.and(ONE).isOne){
if(u.words[0] % 2 == 1){
s=s.multiply(t).mod(m);
}
//u=u.shiftRight(1);
u=u.divide(TWO);
t=t.multiply(t).mod(m);
}

return s;
}
```

```
public BigNumber mod(BigNumber m){
if(m.isnegative || m.isZero())
throw new ArithmeticException("non positive modulus");

return this.remainder(m);
}

public boolean checkPrimeFirst(){
int i;

return true;
}

public int getLowestSetBit(){

if(this.isZero())

return -1;

int i=0;
BigNumber temp[] = this.divideandremainder(TWO);
while(temp[1].compareTo(ZERO)==0){
i++;
temp=temp[0].divideandremainder(TWO);
}
return i;
}

//2進数表現時のビット数を返却
// log2(A)=log10(A)/log10(2)で近似値を求める
public int bitLength(){
//log10(2)=0.30102999
return (int)((this.length-0.5)/0.30102999);
}
```


```
public BigNumber mod(BigNumber m){
if(m.isnegative || m.isZero())
throw new ArithmeticException("non positive modulus");

return this.remainder(m);
}

public boolean checkPrimeFirst(){
int i;

return true;
}

public int getLowestSetBit(){

if(this.isZero())

return -1;

int i=0;
BigNumber temp[] = this.divideandremainder(TWO);
while(temp[1].compareTo(ZERO)==0){
i++;
temp=temp[0].divideandremainder(TWO);
}
return i;
}

//2進数表現時のビット数を返却
// log2(A)=log10(A)/log10(2)で近似値を求める
public int bitLength(){
//log10(2)=0.30102999
return (int)((this.length-0.5)/0.30102999);
}
```

}

文字数: 20671
空白数: 3716 空白込み文字数: 24387
改行数: 955 改行込み文字数: 25342
単語数: 2145

}

文字数: 20432
空白数: 3641 空白込み文字数: 24073
改行数: 944 改行込み文字数: 25017
単語数: 2098

全体を表示 | カラー1 カラー2 モノクロ