1.Simple Sieve :

**package** myPack;

**import** java.util.Scanner;

**public** **class** SimpleSieve {

**public** **static** **void** main(String args[])

{

Scanner sc = **new** Scanner(System.***in***);

**int** n = sc.nextInt();

**int** arr[] = **new** **int**[n+1];

**for**(**int** i = 2;i<=Math.*sqrt*(n);i++ )

{

**for**(**int** j=i\*i;j<=n;j+=i)

{

**if**(j%i==0)

arr[j]=-1;

}

}

**for**(**int** i=2;i<=n;i++)

**if**(arr[i]!=-1)

System.***out***.print(i+" ");

sc.close();

}

}

2.Segmented Sieve:

**package** myPack;

**import** java.util.\*;

**public** **class** SegmentedSieve {

**public** **static** **void** SimpleSieve(**int** n,Vector<Integer> Prime)

{

**int** arr[]=**new** **int**[n+1];

**for**(**int** i=2;i<=Math.*sqrt*(n);i++)

{

**for**(**int** j=i\*i;j<=n;j+=i)

arr[j]=-1;

}

**for**(**int** i=2;i<=n;i++)

**if**(arr[i]!=-1)

Prime.add(i);

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

**int** low = sc.nextInt(),high = sc.nextInt();

Vector<Integer> Prime = **new** Vector<>();

*SimpleSieve*((**int**)Math.*sqrt*(high),Prime);

**int** arr[]=**new** **int**[high-low];

**for**(**int** i=0;i<high-low;i++)

arr[i]=low+i;

**for**(**int** i=0;i<Prime.size();i++)

{

**int** x = Prime.get(i);

**for**(**int** j=0;j<high-low;j++)

{

**if**(arr[j]%x == 0 && arr[j]!=x)

arr[j]=-1;

}

}

**for**(**int** i=0;i<high-low;i++)

**if**(arr[i]!=-1 && arr[i]!=1)

System.***out***.print(arr[i]+" ");

}

}

3.Remainder Theorem:

**package** myPack;

**import** java.util.\*;

**class** Remainder{

**public** **static** **void** main(String args[])

{

Scanner sc = **new** Scanner(System.***in***);

**int** k = sc.nextInt();

**int** num[]=**new** **int**[k];

**int** rem[]=**new** **int**[k];

**int** pp[]=**new** **int**[k];

**int** inv[] = **new** **int**[k];

**int** prod=1;

**for**(**int** i=0;i<k;i++) {

num[i]=sc.nextInt();

prod\*=num[i];

}

**int** x =0;

**for**(**int** i=0;i<k;i++)

rem[i]=sc.nextInt();

**for**(**int** i=0;i<k;i++)

{

pp[i]=prod/num[i];

**for**(**int** j=0;j<=num[i];j++)

{

**if**(pp[i]\*j % num[i] ==1)

{

inv[i]=j;

**break**;

}

}

x+=rem[i]\*pp[i]\*inv[i];

}

x=x%prod;

System.***out***.println(x);

}

}

4.Eulers Algorithm

**package** myPack;

**import** java.util.Scanner;

**public** **class** Euler {

**public** **static** **int** GCD(**int** a,**int** b)

{

**int** min = a>b ? b:a;

**for**(**int** i = min ; i>=1;i--)

**if**(a%i==0 && b%i==0)

**return** i;

**return** 1;

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

**int** n = sc.nextInt();

**int** count=0;

**for**(**int** i=1;i<n;i++)

**if**(*GCD*(i,n)==1)

count++;

System.***out***.println(count);

}

}

5.Strobogrammatic Number :

**package** myPack;

**import** java.util.Scanner;

**public** **class** StrobogrammaticNumber {

**public** **static** **boolean** checkStrobogrammatic(**int** n,**int** rev)

{

**while**(n!=0) {

**switch**(n%10)

{

**case** 0 :

**if**(rev%10 != 0)

**return** **false**;

**break**;

**case** 1 :

**if**(rev%10 != 1)

**return** **false**;

**break**;

**case** 6:

**if**(rev%10 != 9)

**return** **false**;

**break**;

**case** 8:

**if**(rev%10 != 8)

**return** **false**;

**break**;

**case** 9:

**if**(rev%10 != 6)

**return** **false**;

**break**;

**default** :

**return** **false**;

}

rev/=10;

n/=10;

}

**return** **true**;

}

**public** **static** **int** reverse(**int** n){

**int** rev = 0;

**while**(n!=0) {

**int** rem = n%10;

rev = rev\*10 + rem;

n/=10;

}

**return** rev;

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

**int** n = sc.nextInt();

**for**(**int** i = (**int**)Math.*pow*(10,n-1);i<Math.*pow*(10,n);i++) {

**int** rev = *reverse*(i);

**if**(*checkStrobogrammatic*(i,rev))

System.***out***.print(i+ " ");

}

}

}