Simple Sieve Agorithm

Solution

Following are the prime numbers smaller than or equal to 10

```
class SieveOfEratosthenes
{
   void sieveOfEratosthenes(int n)
   {
        boolean prime[] = new boolean[n+1];
      for(int i=0;i<=n;i++)
         prime[i] = true;
       for(int p = 2; p*p <=n; p++)
      {
         // If prime[p] is not changed, then it is a prime
         if(prime[p] == true)
         {
            // Update all multiples of p
            for(int i = p*p; i <= n; i += p)
               prime[i] = false;
         }    }
      for(int i = 2; i <= n; i++)
      {
         if(prime[i] == true)
            System.out.print(i + " ");
      }    }

   public static void main(String args[])
   {   int n = 10;
      System.out.print("Following are the prime numbers ");
      System.out.println("smaller than or equal to " + n);
      SieveOfEratosthenes g = new SieveOfEratosthenes();
      g.sieveOfEratosthenes(n);
   }}
```

Output: 10

Segmented & Incremental Sieve Algorithm

Solution

```
import java.util.Scanner;
class SievePrimeGenerator
{static int array[];
static int primes[];
public static void calculate(int n, int m)
{
 int j = 0;
 int sqt = (int) Math.sqrt(m);
 array = new int[sqt + 1];
 primes = new int[sqt + 1];
initialise(sqt + 1);
 for (int i = 2; i <= sqt; i++) {
   if (array[i] == 1) {
   primes[j] = i;
   j++;
     for (int k = i + i; k <= sqt; k += i) {
       array[k] = 0;
     }
 }
 }
 int diff = (m — n + 1);
 array = new int[diff];
 initialise(diff);
 for (int k = 0; k < j; k++) {
 int div = n / primes[k];
 div *= primes[k];
 while (div <= m) {
   if(div>=n && primes[k]!=div)
   array[div-n] = 0;
   div += primes[k];
 } }
```

```
for (int i = 0; i < diff; i++) {
 if (array[i] == 1 && (i+n) !=1)
 System.out.println(i + n);
 } }

public static void initialise(int sqt) {
 for (int i = 0; i < sqt; i++) {
 array[i] = 1;
 }
 }
public static void main(String arg[]) {
 int t, n, m;
 Scanner in = new Scanner(System.in);
 t = in.nextInt();
 for (int i = 0; i < t; i++)
{ n = in.nextInt();
 m = in.nextInt();
 calculate(n, m);
 System.out.println();
 }
 in.close();
 }}
```

Output

Primes in Range 2 to 100 are

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

The Euler Phi Function

Solution

```
class Euler{
    // Function to return GCD of a and b
   static int gcd(int a, int b)
   {
      if (a == 0)
         return b;
      return gcd(b % a, a);
   }
    static int phi(int n)
   {
      int result = 1;
      for (int i = 2; i < n; i++)
         if (gcd(i, n) == 1)
            result++;
      return result;
   }
    public static void main(String[] args)
   {   int n;
       for (n = 1; n <= 10; n++)
          System.out.println("phi(" + n + ") = " + phi(n));
   }}
```

Output

```
phi(1) = 1
phi(2) = 1
phi(3) = 2
phi(6) = 2
phi(7) = 6
phi(8) = 4
phi(9) = 6
phi(10) = 4
```