Lab 3 Report

# Number Theory

## Student Data :

Name : Mark Nader Fathy

ID : 18011305

Name : Mina Henen Shafik

ID : 18011939

# Sieve of Eratosthenes Algorithm

## Problem Statement

Implement sieve of Eratosthenes algorithm for finding all prime numbers up to any given limit.

## Used Data Structures

- ➢ Boolean Array : of size n + 1 to store a value indicating the number is prime or not.
- ➢ Integer Arraylist : used to store all primes up to a given limit.

## Algorithms used documented using pseudo code

```
For i = 2 to sqrt(n)
        if (flags[i] = true)
                For j = i * i;  j <= n;  j += i
                        flags[j] = false
For i = 2 to n
        if (flags[i] = true)
                primes.add(i)
```

## Sample runs

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
1
Enter an integer to calculate the primes to that integer
24
Primes are
2 3 5 7 11 13 17 19 23
```

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
1
Enter an integer to calculate the primes to that integer
103
Primes 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 97 101 103
```

# Trial Division Algorithm

## Problem Statement

Implement Trial Division algorithm for integer factorization.

## Used Data Structures

- Integer Arraylist : used to store all primes up to a square root of a given limit computed by sieve of eratosthenes.

## Algorithms used documented using pseudo code

```
primes = SieveOfEratosthenes.computePrimes(n)
isComposite = false
For i = 2 to primes.size()
        if (n % primes.get(i) = 0)
                Print (“The number is composite”)
                isComposite = true
if (isComposite = false)
        Print (“The number is prime”)
```

## Sample runs

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
2
Enter an integer
30
The entered number is Composite
```

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
2
Enter an integer
7
The entered number is Prime
```

# Extended Euclidean Algorithm

## Problem statement

Implement the extended Euclidean algorithm that finds the greatest common divisor d of two positive integers a and b.

In addition, it outputs Bezout's coefficients s and t such that d = s a + t b

## Used Data Structures

- Integer array: used to store the GCD in index 0 and the Bezout coefficients in indices 1 and 2.

## Algorithms used documented using pseudo code

```
ExtendedEuclidean(n1,n2)
        If n2 == 0
                return n1,1,0
        ans = ExtendedEuclidean(n2,n1%n2)
        gcd= ans[0]
        a= ans[2]
        b= ans[1]- (n1/n2)*ans[2]
        return gcd,a,b
```

## Sample runs

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
3
Enter 2 integers
252 198
The GCD is 18
The Bezout coefficients are 4 and -5
```

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
3
Enter 2 integers
546 462
The GCD is 42
The Bezout coefficients are -5 and 6
```

# Chinese remainder Algorithm

## Problem statement

Implement Chinese remainder theorem that takes as input m1, m2, m3, …., mn that are pairwise relatively prime and (a1, a2, …., an) and calculates x such that

x = a1 (mod m1)

x = a2 (mod m2)

…

x= an (mod mn)

## Used Data Structures

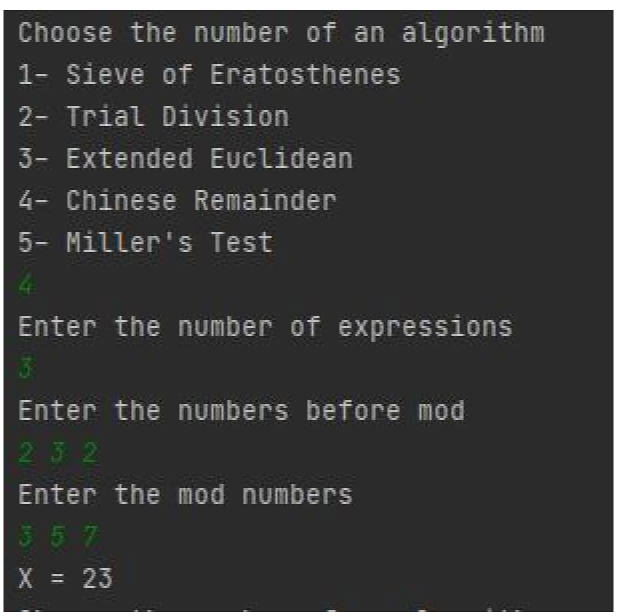
- Integer array: used to store the numbers before mod.
- Integer array: used to store the mod numbers.

## Algorithms used documented using pseudo code

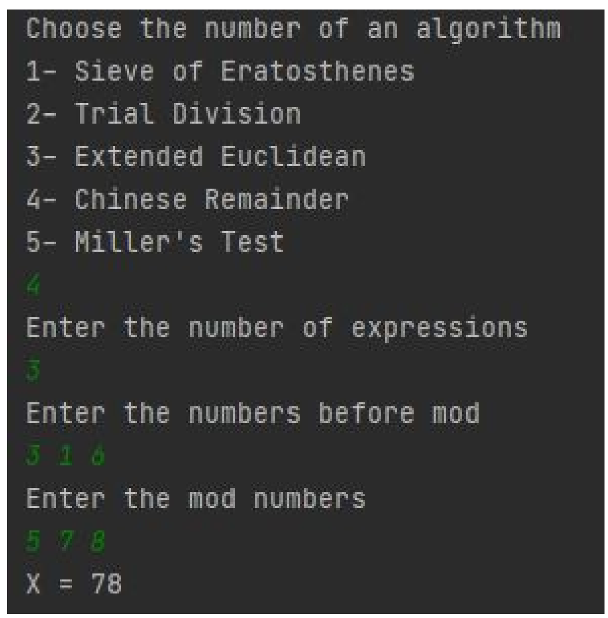
```
chineseReminder(array a,array m)
        product= 1
        result= 0
        for i in m
                product = product*m
        for i from 0 to a length
                p = product /m[i]
                result = result + a[i] * p * modInverse(p,m[i])
        return result%product

modInverse(a,m)
        a = a%m
        for i from 1 to m
                if (a*i)%m == 1
                        return i
        return 1
```

## Sample runs



```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
4
Enter the number of expressions
3
Enter the numbers before mod
2 3 2
Enter the mod numbers
3 5 7
X = 23
```



```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
4
Enter the number of expressions
3
Enter the numbers before mod
3 1 6
Enter the mod numbers
5 7 8
X = 78
```

# Miller’s test

## Problem statement

Implement Miller’s test (a probabilistic primality test).

## Used Data Structures

- No Data Structures are used.

## Algorithms used documented using pseudo code

```
power(base,power,mod)
    Put x = 1
    Put power = a % m
    For i = b.length - 1 to 0 :
        If b.charAt(i) = '1' :
            x = (x * power) % m
            If x > Integer.Max_value :
                Throw OverFlow Exception
        power = (power * power) % m
        i = i + 1
        Loop Again :
    Return x
millersTest(m,n)
    a = 2 + random number %(n-4)
    x = power(a,m,n)
    if (x == 1 or x == n - 1)
        return true
    while (m != n - 1)
        x = x^2 % n
        m = m*2
        if (x == 1)
            return false
        if (x == n - 1)
            return true
    return false
```

```
isPrimeUseMillerTest(n,k)
    if (n <= 1 or n == 4)
        return false
    if (n <= 3)
        return true
    int d = n - 1
    while (d % 2 == 0)
        d = d/2
    for i from 0 to k-1
        if ( ! millersTest(d, n) )
            return false
    return true
```

## Sample runs

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
5
Enter an integer
64689
Enter number of iterations (more iterations more accuracy)
555
The number 64689 is not prime using miller's test.
```

```
Choose the number of an algorithm
1- Sieve of Eratosthenes
2- Trial Division
3- Extended Euclidean
4- Chinese Remainder
5- Miller's Test
5
Enter an integer
5003
Enter number of iterations (more iterations more accuracy)
2
The number 5003 is prime using miller's test.
```