NAME: **YVONNE KIMANI** REG NUMBER: **SCT212-0475/2017**

## ASSIGNMENT 1

1. **Using the theorem divisibility, prove the following** 
   1. **If a|b , then a|bc ∀a, b, c ∈ ℤ ( 5 marks)**

If a∣b then there exists an integer k such that b=a⋅k (k ∈ ℤ).

Substituting b for ak, bc=akc=a(kc)

Since multiplication is associative and the last expression is clearly divisible by a

Then a|bc

* 1. **If a|b and b|c , then a|c (5 marks)**

if a∣b and b∣c, we can write b=ka, k∈Z and c=bl, l∈Z.

Combining these two gives c=bl=(ka)l=(ak)l=a(kl)

Since multiplication is both commutative and associative. The last expression is clearly divisible by a.

1. **Using any programming language of choice (preferably python), implement the following algorithms**
   1. **Modular exponentiation algorithm (10 marks)**

**//Program written in Java**

Public class ModularExponentiation {

static int power(int x, int y, int p)

{

// Initialize result

int result = 1;

// Update x if it is more than or equal to p

x = x % p;

// In case x is divisible by p;

if (x == 0) return 0;

while (y > 0)

{

// If y is odd, multiply x with result

if((y & 1)==1)

result = (result \* x) % p;

// y must be even now, y = y / 2

y = y >> 1;

x = (x \* x) % p;

}

return result;

}

// Driver Program to test above functions

public static void main(String args[])

{

int x = 2;

int y = 5;

int p = 13;

System.out.println("Exponential is " + power(x, y, p));

}

}

* 1. **The sieve of Eratosthenes (10 marks)**

**//Program written in Java**

Public class SieveOfEratosthenes

{

void sieveOfEratosthenes(int n)

{

// Create a boolean array "prime[0..n]" and initialize

// all entries it as true. A value in prime[i] will

// finally be false if i is Not a prime, else true.

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\*2; i <= n; i += p)

prime[i] = false;

}

}

// Print all prime numbers

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

{

if(prime[i] == true)

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

}

}

// Driver Program to test above function

public static void main(String args[])

{

int n = 30;

System.out.print("Following are the prime numbers ");

System.out.println("smaller than or equal to " + n);

SieveOfEratosthenes g = new SieveOfEratosthenes();

g.sieveOfEratosthenes(n);

}

}

1. **Write a program that implements the Euclidean Algorithm (10 marks)**

**//Program written in Java**

public class GCDExample {

public static void main(String args[]){

//Enter two number whose GCD needs to be calculated.

Scanner scanner = new Scanner(System.in);

System.out.println("Please enter first number to find GCD");

int number1 = scanner.nextInt();

System.out.println("Please enter second number to find GCD");

int number2 = scanner.nextInt();

System.out.println("GCD of two numbers " + number1 +" and " + number2 +" is

:" + findGCD(number1,number2));

}

//Method to find GCD of two number

private static int findGCD(int number1, int number2) {

if(number2 == 0){

return number1;

}

return findGCD(number2, number1 % number2);

}

}

1. **Modify the algorithm above such that it not only returns the gcd of a and b but also the Bezouts coefficients x and y, such that 𝑎𝑥 + 𝑏𝑦 = 1 (10 marks)**
2. **Let m be the gcd of 117 and 299. Find m using the Euclidean algorithm (5 marks)**

(299,117)

299 = 117.2 + 65

(117,65)117 = 65.1 + 52

(65,52)65 = 52.1 + 13

(52,13)52 = 13.4 + 0

GCD = 13

1. **Find the integers p and q , solution to 1002𝑝 + 71𝑞 = 𝑚 (5 marks)**

(1002,71)1002 = 71.14 + 18

(71,8) 71 = 8.8 + 7

(8,7)8 = 7.1 + 1

(7,1)7 = 1.7 + 0

GCD = 1

Solve For Remainders

8 = 1002.1 - 71.4

7 = 71.1 - 8.8

1 = 8.1 - 7.1

Substitute:

1= 8.1 - 7.1

For 7;

8.1 - (71.1 - 8.8)

8(9) - 71(1)

For 8;

9[1002(1)] - 71(14) - 71(1)

1002(9) - 71(126) - 71(1)

1002(9) - 71(127)

1002(9) + 71(-127) = 1

p = 9, q = -127

1. **Determine whether the equation 486𝑥 + 222𝑦 = 6 has a solution such that 𝑥, 𝑦 ∈ 𝑍𝑝 If yes, find x and y. If not, explain your answer. (5 marks)**

(486,222)486 = 222.2 + 42

(222,42)222 = 42.5 + 12

(42,12)42 = 12.3 + 6

(12,6)12 = 6.2 + 0

GCD = 6

Solve for remainder

6 = 42.1 - 12.3

12 = 222.1 - 42.5

42 = 486.1 - 222.2

Substitution: 6 = 42.12.3

For 12: 42 - [(222-42.5)]3

42 - [222.3-42.15]

42 - 222.3 + 42.15

42(16) - 222.3

For 42: 16[486.1 - 222.2] - 222.3

486.16 - 222.32 - 222.3

486.16 - 222.35

x = 16, Y = -35

1. **Determine integers x and y such that 𝑔𝑐𝑑(421, 11) = 421𝑥 + 11𝑦. (5 marks)**

(421,11)421 = 11.38 + 3

(11,3)11 = 3.3 + 2

(3,2)3 = 2.1 + 1

(2,1)2 1.2 + 0

GCD = 1

Solve for remainder

3 = 421.1 - 11.38

2 = 11.1 - 3.3

1 = 3.1 - 2.1

Substitution: 1 = 3.1 - 2.1

For 2:

3.1 - [11-3.3]

3.1 - 11 + 3.3

3.4 - 11

For 3:

4(421 - 11.38) - 11

421.4 - 11.152 - 11

421(4) + 11(-153)

x = 4, y = -153

1. **Explain the working mechanism of the following signature schemes (15 marks)** 
   1. **RSA signature scheme (10 mark)**

The RSA signature scheme is based on PSA encryption. It is the most widely used digital signature in practice. This is how it works

A sender (lets call him Bob) generates the same RSA keys that were used for RSA encryption.

RSA Key Generation

**Step 1:** *Choose two large primes (1024 bits), say p and q.*

**Step 2:** *Compute n (public modulus), n=p.q*

**Step 3:** *Compute α(n), α(n) = (p-1)(q-1)*

**Step 4:** *Select α* ∈ {1…, *α(n) -1* }

**Step 5***: Compute private Key d,*

*s.t d.e* ≡ *1 mod α(n)*

Bob, the sender, takes the message and puts it in a signature algorithm. Bob sends the message as well as the signature to Alice (the recipient). The encrypted message received by Alice undergoes a check to determine if the signature corresponds to Bobs Verification Algorithm.

* 1. **Digital Signature Standard (10 mark)**

This makes use of the hash function. The hashcode is provided as input to a signature function along with a random number key, generated for this particular signature. The signature function also depends on the senders private key and a set of parameters known to a **group** of communicating principle (known in the network). We can consider this set to constitute a global public Key. The result is a signature consisting of two components, s and r.

At the receiving end, the hash code of incoming messages is generated. Signature is input to a verification function which also depends on the global public key as well as the senders public key which is paired with the sender’s private key.

The output of the verification function is a value that’s equal to the signature component r, if the signature is valid.

The signature function is such that only the sender with knowledge of the private key could have produced the valid signature.

* 1. **Schnorr Signature Scheme(10 mark)**