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

ASSIGNMENT 1

---

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

   ∃ k ∈ ℤ s.t b=ak
   a|bc; bc =(int) a
   bc = (ak)x = (kx)a
   where kx ∈ ℤ
   So, a|bc

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

   a|b iff ∃ k ∈ ℤ s.t b = ak
   b|c iff ∃ k ∈ ℤ s.t c = bf

   c = bf = (ak)f
   c = bf = (fk)a

   so a|c

2. **Using any programming language of choice (preferably python), implement the following algorithms**
   a. **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));
  }
}
```

b. 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);
    }
}
```

3. **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);
    }
}
```

4. **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)**

//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));

        /** Call function to get the coefficients **/
        GetCoefficients(a, b);
    }

     //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);
    }

    public void GetCoefficients(long a, long b)
    {
        long x = 0, y = 1, lastx = 1, lasty = 0, temp;
        long tempa = a , tempb = b;
        while (b != 0)
        {
            long q = a / b;
            long r = a % b;
            a = b;
            b = r;

            temp = x;
            x = lastx - q * x;
            lastx = temp;
            temp = y;
            y = lasty - q * y;
            lasty = temp;
        }
        System.out.println(tempa + "("+lastx+")+"+tempb+"("+lasty+") =
            gcd("+tempa+","+tempb+")");
        System.out.println("Coefficients x : "+ lastx +" y :"+ lasty);
    }
}
```

5. **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

6. **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

7. **Determine whether the equation 486x + 222y = 6 has a solution such that $x, y \in Zp$ 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 = 42.1-12.3
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

8. **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

9. **Explain the working mechanism of the following signature schemes (15 marks)**

   a. **RSA signature scheme (10 mark)**

   The RSA signature scheme is based on RSA 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.
   <u>RSA Key Generation</u>
   Step 1: *Choose two large primes (1024 bits), say p and q.*
   Step 2: *Compute n (public modulus),* ***n=p*q***
   Step 3: *Compute* ρ*(n),* ρ*(n) = (p-1)(q-1)*
   Step 4: *Select* ρ ∈ {1…, ρ*(n) -1* }

Such that gcd(e, ρ(n)) = 1

Step 5*: Compute private Key d,*

s.t d.e ≡ 1 mod ρ(n)

Computation of signature algorithm

Input consists of the Message and the private key.

The signature s is computed using s = m ^ d mod n.

The signature is attached to the message. The sender (Bob) sends the message and the signature to the recipient(Alice). The message has to be encrypted first for security purposes. He uses symmetric encryption for this. The recipient receives the message and checks using the verification algorithm to check if the signature corresponds to the senders.

Verification Algorithm

Input consists of the Message, the signature and the public key

M' = s ^ e mod n

If M' = m, then return "true"

If M' != M then return "false"

;output is True or false

If the it returns true, then the author of the message was in possession of Bob's key and message M has not been changed in transit.

b. Digital Signature Standard (10 mark)

This makes use of the hash function. The hash code is provided as input to a signature function along with a random number k, generated for this particular signature. The signature function also depends on the sender's 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.

c. Schnorr Signature Scheme (10 mark)

The Schnorr signature scheme is a digital signature scheme known for its simplicity, is efficient and generates short signatures. The scheme is based on discrete logarithms which minimize the amount of computation required to generate a signature.

The message-dependent part of the signature generation requires multiplying a 2n-bit integer with an n-bit integer. The scheme is based on using a prime modulus p, with p - 1 having a prime factor q of appropriate size; that is, p - 1 == (mod q).

The first part of this scheme is the generation of a private/public key pair, which consists of:

1. *Choose primes p and q, such that q is a factor of p - 1.*
2. *Choose an integer a, such that a^q = 1 mod p. The values a, p, and q comprise a global public key that can be common to a group of users.*
3. *Choose a random integer s with 0 < s < q. This is the user's private key.*
4. *Calculate v = a ^ s mod p. This is the user's public key.*

A user with private key s and public key v generates a signature as follows.

1. *Choose a random integer r with 0 < r < q and compute x = a ^ r mod p. This computation is a preprocessing stage independent of the message M to be signed.*
2. *Concatenate the message with x and hash the result to compute the value e:*
3. *Compute y = (r + se) mod q. The signature consists of the pair (e, y).*

The sender then sends the message M together with the signature (e,y). The recipient receives the message M and the signature (e,y).

Any other user can verify the signature as follows.

1. *Compute x' = a ^ y v ^ e mod p.*
2. *Verify that x'= a* ^ y v ^ e = a ^ y a ^ - se = a ^ r = x mod p

   y = r + se
3. *Verify that e = H (M || x').*

To see that the verification works, observe that

Hence, *H (M || x') = H (M || x).*