## Experiment 1

**Write a java program that contains a string (char pointer) with a value \Hello World'. The programs should XOR each character in this string with 0 and display the result.**

```
public class XorString {
   public static void main(String[] args) {
      String str = "Hello World";
      int len = str.length();
      char[] str1 = new char[len];

      for (int i = 0; i < len; i++) {
         str1[i] = (char)(str.charAt(i) ^ 0);  // XOR with 0
         System.out.print(str1[i]);
      }

      System.out.println();
   }
}
```

## Output

Hello World

**Experiment 2**

**Write a java program that contains a string (char pointer) with a value '\Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.**

```
public class AndXorString {
    public static void main(String[] args) {
        String str = "Hello World";
        int len = str.length();

        char[] andResult = new char[len];
        char[] xorEncrypted = new char[len];
        char[] xorDecrypted = new char[len];

        // AND with 127
        System.out.print("AND with 127: ");
        for (int i = 0; i < len; i++) {
            andResult[i] = (char)(str.charAt(i) & 127);
            System.out.print(andResult[i]);
        }
        System.out.println();

        // XOR with 127 (encryption)
        for (int i = 0; i < len; i++) {
            xorEncrypted[i] = (char)(str.charAt(i) ^ 127);
        }

        // XOR again with 127 (decryption → gets back Hello World)
        System.out.print("XOR with 127: ");
        for (int i = 0; i < len; i++) {
            xorDecrypted[i] = (char)(xorEncrypted[i] ^ 127);
```

```
            System.out.print(xorDecrypted[i]);
        }
        System.out.println();
    }
}
```

**Output**

AND with 127: Hello World

XOR with 127: Hello World

**Experiment 3**

**Write a Java program to perform encryption and decryption using the following algorithms:**

**a. Ceaser Cipher**

```
import java.util.*;
import java.io.*;

public class Caesercipher {
   public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";
   public static String encrypt(String ptext, int cserkey)
   {
      String ctext = "";
      for (int i = 0; i < ptext.length(); i++)
      {
         int plainnumeric = ALPHABET.indexOf(ptext.charAt(i));
         int ciphernumeric = (plainnumeric+cserkey) % 26;
         char cipherchar = ALPHABET.charAt(ciphernumeric);
         ctext += cipherchar;
      }
      return ctext;
   }

   public static String decrypt(String ctext, int cserkey)
   {
      String ptext = "";
      for (int i = 0; i < ctext.length(); i++)
      {
         int ciphernumeric = ALPHABET.indexOf(ctext.charAt(i));
         int plainnumeric= (ciphernumeric-cserkey) % 26;
         if (plainnumeric < 0)
```

```
        {
            plainnumeric = ALPHABET.length() + plainnumeric;
        }
        char plainchar = ALPHABET.charAt(plainnumeric);
        ptext += plainchar;
    }
    return ptext;
}

public static void main(String[] args) throws IOException
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the PLAIN TEXT for Encryption: ");
    String plaintext = new String();
    String ciphertext = new String();
    String key;
    int cserkey;
    plaintext = br.readLine();
    System.out.println("Enter the CAESERKEY between 0 and 25:");
    key = br.readLine();
    cserkey = Integer.parseInt(key);

    System.out.println("ENCRYPTION");
    ciphertext = encrypt(plaintext,cserkey);
    System.out.println("CIPHER TEXT :"+ ciphertext);

    System.out.println("DECRYPTION");
    plaintext = decrypt(ciphertext,cserkey);
    System.out.println("PLAIN TEXT :" + plaintext);
}
```

```
}
```

**Output**

```
Enter the PLAIN TEXT for Encryption:
information
Enter the CAESERKEY between 0 and 25:
7
ENCRYPTION
CIPHER TEXT :pumvythapvu
DECRYPTION
PLAIN TEXT :information
```

**b. Substitution Cipher**

```
import java.io.*;
import java.util.*;
public class SubstitutionCipher {
    static Scanner sc = new Scanner(System.in);
    static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    public static void main(String[] args) throws IOException {
// TODO code application logic here
        String a = "abcdefghijklmnopqrstuvwxyz";
        String b = "zyxwvutsrqponmlkjihgfedcba";
        System.out.print("Enter any string: ");
        String str = br.readLine();
        String decrypt = "";
        char c;
        for(int i=0;i<str.length();i++)
        {
            c = str.charAt(i);
            int j = a.indexOf(c);
            decrypt = decrypt+b.charAt(j);
        }
        System.out.println("The encrypted data is: " +decrypt);
    }
}
```

**Output**

Enter any string: information

The encrypted data is: rmulinzgrlm

**Experiment 4**

**Using Java Cryptography, encrypt the text "Hello world" using BlowFish. (Note : Create InputFile.txt)**

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import java.util.Base64;

public class BlowFish {
    public static void main(String[] args) throws Exception {
        // Generate Blowfish key (128-bit)
        KeyGenerator keyGenerator = KeyGenerator.getInstance("Blowfish");
        keyGenerator.init(128);
        Key secretKey = keyGenerator.generateKey();

        // Setup cipher for encryption
        Cipher cipherOut = Cipher.getInstance("Blowfish/CFB/NoPadding");
        cipherOut.init(Cipher.ENCRYPT_MODE, secretKey);

        // Get Initialization Vector (IV) and print it in Base64
        byte iv[] = cipherOut.getIV();
        if (iv != null) {
            String ivBase64 = Base64.getEncoder().encodeToString(iv);
            System.out.println("Initialization Vector of the Cipher: " + ivBase64);

            // Open files
            FileInputStream fin = new FileInputStream("inputFile.txt");
```

```
        FileOutputStream fout = new FileOutputStream("outputFile.txt");

        // Wrap output with cipher
        CipherOutputStream cout = new CipherOutputStream(fout, cipherOut);

        // Read file byte by byte and encrypt
        int input;
        while ((input = fin.read()) != -1) {
           cout.write(input);
        }

        // Close resources
        fin.close();
        cout.close();
        fout.close();

        System.out.println("Encryption complete. Encrypted file saved as outputFile.txt");
    }
  }
}
```

**Output**

Initialization Vector of the Cipher: ptdJhtHX8Jg=

Encryption complete. Encrypted file saved as outputFile.txt

**Experiment 5**

**Using Java Cryptography, encrypt a paragraph using Transposition Technique.**

```
import java.util.*;
import java.io.*;

public class transposition {
   public static int key[] = new int[8];
   public char mat[][] = new char[10][8];
   public char pmat[][] = new char[10][8];
   public char cmat[][] = new char[10][8];
   String plain="";
   String cipher="";
   int rows=0, col;

   public String tencryption(String text1)
   {
      int i,j,len,ch,k,p=0;
      String enctxt="";
      String text = "";
      len = text1.length();

      for(i=0;i<len;i++)
         text += text1.charAt(i);

      if (( len % 7 ) != 0)
      {
         rows = ( len / 7 ) + 1;
         ch = len % 7;
         for (i=0;i<(7-ch);i++)
```

```
        text += 'X';
}
else
   rows = len / 7;
k=0;
for(i=1;i<=rows;i++)
{
   for(j=1;j<=7;j++)
      mat[i][j] = text.charAt(k++);
}

for(i=1;i<=rows;i++)
{
   for(j=1;j<=7;j++)
      System.out.print(mat[i][j] + " ");
   System.out.println();
}

k = 1;
j = 1;
while ( k <= 7 )
{
   for(p=0;p<7;p++)
   {
      if ( k == key[p] )
      {
         j=p+1;
         k++;
         break;
      }
```

```
        }
        for(i=1;i<=rows;i++)
            enctxt+=mat[i][j];
    }

    System.out.println(enctxt);
    return enctxt;
}

public String tdecryption(String txt,int plength)
{
    int i,j=1,len,k=1,p,q=0;
    String dectxt="";
    String ptext="";

    while (k<=7)
    {
        for(p=0;p<7;p++)
        {
            if (key[p] == k)
            {
                j = p+1;
                k++;
                break;
            }
        }
        for(i=1;i<=rows;i++)
            cmat[i][j] = txt.charAt(q++);
    }
```

```
    for(i=1;i<=rows;i++)
    {
        for(j=1;j<=7;j++)
            System.out.print(cmat[i][j] + " ");
        System.out.println();
    }

    for(i=1;i<=rows;i++)
    {
        for(j=1;j<=7;j++)
            dectxt += cmat[i][j];
    }

    len = dectxt.length();
    if (plength < len)
    {
        for(i=0;i<plength;i++)
            ptext += dectxt.charAt(i);
    }

    return ptext;
}

public static void main(String[] args) throws IOException
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    transposition tp = new transposition();
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter key: ");
```

```
String keyword = br.readLine().toUpperCase();

// convert keyword to numeric key
int len = keyword.length();
Integer[] order = new Integer[len];
for (int i = 0; i < len; i++) {
   order[i] = i;
}

// sort positions by corresponding character
Arrays.sort(order, (a, b) -> Character.compare(keyword.charAt(a), keyword.charAt(b)));

// assign numbers based on sorted order
int[] numericKey = new int[len];
for (int i = 0; i < len; i++) {
   numericKey[order[i]] = i + 1;
}

// store into static key[]
for (int i = 0; i < len; i++) {
   key[i] = numericKey[i];
}

System.out.println("KEY (numeric): " + Arrays.toString(key));

System.out.println("Enter PLAIN TEXT: ");
String plain = sc.next().toUpperCase();
int k = plain.length();

System.out.println("Plain: " + plain);
```

```
        String ctext = tp.tencryption(plain);
        System.out.println("\nCIPHER TEXT: " + ctext);

        String plaintext = tp.tdecryption(ctext, k);
        System.out.println("\nDECRYPTED TEXT: " + plaintext);

        sc.close();
    }
}
```

**Output**

```
Enter key:
AUTHOR

KEY (numeric): [1, 6, 5, 2, 3, 4, 0, 0]

Enter PLAIN TEXT:
ATTACKPOSTPONEDUNTILTWOAM

Plain: ATTACKPOSTPONEDUNTILTWOAM

A T T A C K P
O S T P O N E
D U N T I L T
W O A M X X X
```

**Experiment 6**

**Calculate the message digest of a text using the SHA-1 algorithm in JAVA.**

```
import java.security.*;
public class SHA1 {
   public static void main(String[] a) {
      try {
         MessageDigest md = MessageDigest.getInstance("SHA1");
         System.out.println("Message digest object info: ");
         System.out.println(" Algorithm = " +md.getAlgorithm());
         System.out.println(" Provider = " +md.getProvider());
         System.out.println(" ToString = " +md.toString());
         String input = "";
         md.update(input.getBytes());
         byte[] output = md.digest();
         System.out.println();
         System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
         input = "abc";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println();
         System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
         input = "abcdefghijklmnopqrstuvwxyz";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println();
         System.out.println("SHA1(\"" +input+"\") = " +bytesToHex(output));
         System.out.println(""); }
      catch (Exception e) {
         System.out.println("Exception: " +e);
```

```
        }
    }
    public static String bytesToHex(byte[] b) {
        char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
        StringBuffer buf = new StringBuffer();
        for(int j=0; j<b.length; j++) {
            buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
            buf.append(hexDigit[b[j] & 0x0f]); }
        return buf.toString();
    }
}
```

**Output**

```
Message digest object info:
 Algorithm = SHA1
 Provider = SUN version 23
 ToString = SHA1 Message Digest from SUN, <initialized>


SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz") = 32D10C7B8CF96570CA04CE37F2A19D84240D3A89
```

**Experiment 7**

**Calculate the message digest of a text using the MD5 algorithm in JAVA**

```
import java.security.*;
public class MD5 {
   public static void main(String[] a) {
      try {
         MessageDigest md = MessageDigest.getInstance("MD5");
         System.out.println("Message digest object info: ");
         System.out.println(" Algorithm = " + md.getAlgorithm());
         System.out.println(" Provider = " + md.getProvider());
         System.out.println(" ToString = " + md.toString());
         String input = "";
         md.update(input.getBytes());
         byte[] output = md.digest();
         System.out.println();
         System.out.println("MD5(\"" + input + "\") = " + bytesToHex(output));
         input = "abc";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println();
         System.out.println("MD5(\"" + input + "\") = " + bytesToHex(output));
         input = "abcdefghijklmnopqrstuvwxyz";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println();
         System.out.println("MD5(\"" + input + "\") = " + bytesToHex(output));
         System.out.println("");
      } catch (Exception e) {
         System.out.println("Exception: " + e);
```

```
    }
  }
    public static String bytesToHex(byte[] b) {
      char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
      StringBuffer buf = new StringBuffer();
      for (int j=0; j<b.length; j++) {
        buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
        buf.append(hexDigit[b[j] & 0x0f]);
      }
      return buf.toString();
    }
  }
```

**Output**

Message digest object info:
Algorithm = MD5
Provider = SUN version 23
ToString = MD5 Message Digest from SUN, <initialized>

MD5("") = D41D8CD98F00B204E9800998ECF8427E

MD5("abc") = 900150983CD24FB0D6963F7D28E17F72

MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B

**Experiment 8**

**Write a program to implement the digital signature scheme in Java**

```
import java.util.*;
import java.math.BigInteger;
public class dsaAlg {
   final static BigInteger one = new BigInteger("1");
   final static BigInteger zero = new BigInteger("0");

   /* incrementally tries for next prime */
   public static BigInteger getNextPrime(String ans)
   {
      BigInteger test = new BigInteger(ans);
      while (!test.isProbablePrime(99))
      {
         test = test.add(one);
      }
      return test;
   }

   /* finds largest prime factor of n */
   public static BigInteger findQ(BigInteger n)
   {
      BigInteger start = new BigInteger("2");
      while (!n.isProbablePrime(99))
      {
         while (!((n.mod(start)).equals(zero)))
         {
            start = start.add(one);
         }
```

```
        n = n.divide(start);
    }
    return n;
}
/* finds a generator mod p */
public static BigInteger getGen(BigInteger p, BigInteger q, Random r)
{
    BigInteger h = new BigInteger(p.bitLength(), r);
    h = h.mod(p);
    return h.modPow((p.subtract(one)).divide(q), p);
}

public static void main (String[] args) throws java.lang.Exception
{
    Random randObj = new Random();

    /* establish the global public key components */
    BigInteger p = getNextPrime("10600"); /* approximate prime */
    BigInteger q = findQ(p.subtract(one));
    BigInteger g = getGen(p,q,randObj);

    /* public key components */
    System.out.println("Digital Signature Algorithm");
    System.out.println("global public key components are:");
    System.out.println("p is: " + p);
    System.out.println("q is: " + q);
    System.out.println("g is: " + g);

    /* find the private key */
    BigInteger x = new BigInteger(q.bitLength(), randObj);
```

```
x = x.mod(q);

/* corresponding public key */
BigInteger y = g.modPow(x,p);

/* random value message */
BigInteger k = new BigInteger(q.bitLength(), randObj);
k = k.mod(q);

/* randomly generated hash value and digital signature */
BigInteger r = (g.modPow(k,p)).mod(q);
BigInteger hashVal = new BigInteger(p.bitLength(), randObj);
BigInteger kInv = k.modInverse(q);
BigInteger s = kInv.multiply(hashVal.add(x.multiply(r)));
s = s.mod(q);
/* secret information */
System.out.println("secret information are:");
System.out.println("x (private) is: " + x);
System.out.println("k (secret) is: " + k);
System.out.println("y (public) is: " + y);
System.out.println("h (rndhash) is: " + hashVal);
System.out.println("Generating digital signature:");
System.out.println("r is : " + r);
System.out.println("s is : " + s);

/*verify the digital signature */
BigInteger w = s.modInverse(q);
BigInteger u1 = (hashVal.multiply(w)).mod(q);
BigInteger u2 = (r.multiply(w)).mod(q);
BigInteger v = (g.modPow(u1,p)).multiply(y.modPow(u2,p));
```

```
        v = (v.mod(p)).mod(q);
        System.out.println("verifying digital signature (checkpoints):");
        System.out.println("w is : " + w);
        System.out.println("u1 is : " + u1);
        System.out.println("u2 is : " + u2);
        System.out.println("v is : " + v);
        if (v.equals(r))
        {
           System.out.println("success: digital signature is verified! " + r);
        }
        else
        {
           System.out.println("error: incorrect digital signature");
        }
    }
}
```

**Output**

```
Digital Signature Algorithm
global public key components are:
p is: 10601
q is: 53
g is: 2619
secret information are:
x (private) is: 9
k (secret) is: 36
y (public) is: 1910
h (rndhash) is: 6826
Generating digital signature:
r is : 46
```

s is : 48

verifying digital signature (checkpoints):

w is : 21

u1 is : 34

u2 is : 12

v is : 46

success: digital signature is verified! 46

## Experiment 9

## Implement Diffi-Hellmen Key exchange Method.

```
import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.SecureRandom;
import javax.crypto.spec.DHParameterSpec;
import javax.crypto.spec.DHPublicKeySpec;


public class DiffeHellman {
   public final static int
         pValue = 47;
   public final static int gValue = 71;
   public final static int XaValue = 9;
   public final static int XbValue = 14;
   public static void main(String[] args) throws Exception {
      BigInteger p = new BigInteger(Integer.toString(pValue));
      BigInteger g = new BigInteger(Integer.toString(gValue));
      BigInteger Xa = new BigInteger(Integer.toString(XaValue));
      BigInteger Xb = new BigInteger(Integer.toString(XbValue));
      createKey();
      int bitLength = 512; // 512 bits
      SecureRandom rnd = new SecureRandom();
      p = BigInteger.probablePrime(bitLength, rnd);
      g = BigInteger.probablePrime(bitLength, rnd);
      createSpecificKey(p, g);
   }
   public static void createKey() throws Exception {
```

```
        KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");
        kpg.initialize(512);
        KeyPair kp = kpg.generateKeyPair();
        KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");
        DHPublicKeySpec kspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(),
DHPublicKeySpec.class);
        System.out.println("Public key is: " +kspec);
    }
    public static void createSpecificKey(BigInteger p, BigInteger g) throws Exception {
        KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");
        DHParameterSpec param = new DHParameterSpec(p, g); kpg.initialize(param);
        KeyPair kp = kpg.generateKeyPair();
        KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");
        DHPublicKeySpec kspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(),
DHPublicKeySpec.class);
        System.out.println("\nPublic key is : " +kspec);
    }
}
```

**Output**

```
Public key is: javax.crypto.spec.DHPublicKeySpec@66cd51c3
Public key is : javax.crypto.spec.DHPublicKeySpec@1b9e1916
```

**Experiment 10**

**Implement encryption and decryption using RSA algorithm generating public and private keys.**

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.math.*;
import java.util.Random;
import java.util.Scanner;


public class RSA {
   static Scanner sc = new Scanner(System.in);
   public static void main(String[] args) {
      System.out.print("Enter a Prime number: ");
      BigInteger p = sc.nextBigInteger();
      System.out.print("Enter another prime number: ");
      BigInteger q = sc.nextBigInteger();
      BigInteger n = p.multiply(q);
      BigInteger n2 = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
      BigInteger e = generateE(n2);
      BigInteger d = e.modInverse(n2);
      System.out.println("Encryption keys are: " + e + ", " + n);
      System.out.println("Decryption keys are: " + d + ", " + n);
   }
   public static BigInteger generateE(BigInteger fiofn) {
      int y, intGCD;
      BigInteger e;
      BigInteger gcd;
      Random x = new Random();
      do {
         y = x.nextInt(fiofn.intValue()-1);
```

```
        String z = Integer.toString(y);
        e = new BigInteger(z);
        gcd = fiofn.gcd(e);
        intGCD = gcd.intValue();
      }
      while(y <= 2 || intGCD != 1);
      return e;
   }
}
```

**Output**

Enter a Prime number: 5

Enter another prime number: 11

Encryption keys are: 33, 55

Decryption keys are: 17, 55