Information Security Systems

PRACTICAL FILE

(CSX-324)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Dr. B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY

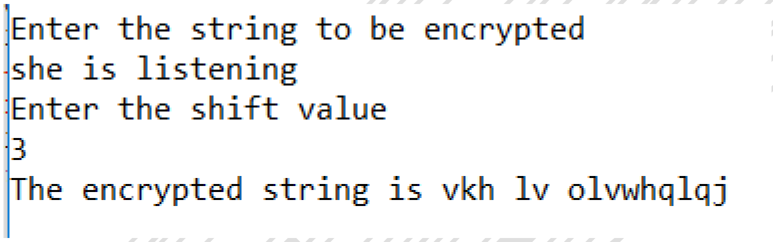
JALANDHAR – 144011, PUNJAB (INDIA)

January- May 2017

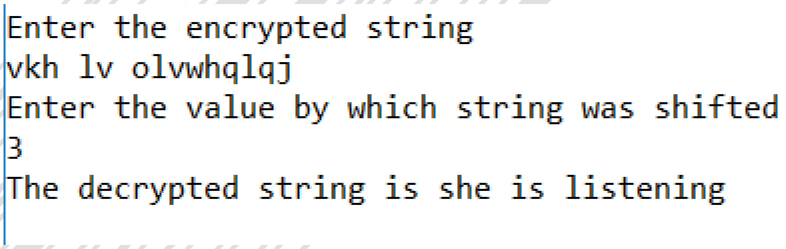
Submitted To:	Submitted By:
Ms. Raj Mohan	saiganesh munduru
Assistant Professor	14103021
Dept. Of CSE	CSE III Year

Aim:- To implement Caesar Substitution Cipher and decipherment .

```
# Caesar Cipher

MAX_KEY_SIZE = 26
def getMode():
    while True:
        print('Do you wish to encrypt or decrypt a message?')
        mode = input().lower()
        if mode in 'encrypt e decrypt d'.split():
            return mode
        else:
            print('Enter either "encrypt" or "e" or "decrypt" or "d".')


def getMessage():
    print('Enter your message:')
    return input()


def getKey():
    key = 0
    while True:
        print('Enter the key number (1-%s)' % (MAX_KEY_SIZE))
        key = int(input())
        if (key >= 1 and key <= MAX_KEY_SIZE):
            return key

def getTranslatedMessage(mode, message, key):
    if mode[0] == 'd':
        key = -key
    translated = ''
    for symbol in message:
        if symbol.isalpha():
            num = ord(symbol)
            num += key

            if symbol.isupper():
                if num > ord('Z'):
                    num -= 26
                elif num < ord('A'):
                    num += 26
            elif symbol.islower():
                if num > ord('z'):
                    num -= 26
                elif num < ord('a'):
                    num += 26

            translated += chr(num)
        else:
            translated += symbol
    return translated
        mode = getMode()
message = getMessage()
key = getKey()
print('Your translated text is:')
print(getTranslatedMessage(mode, message, key))
```



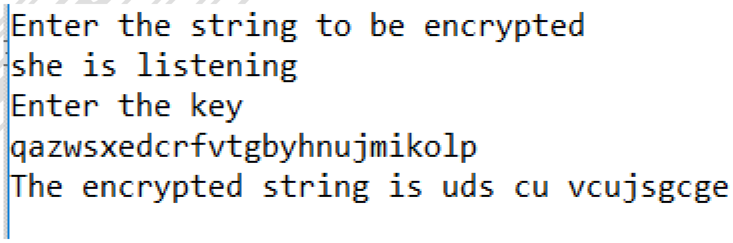
```
Enter the string to be encrypted
she is listening
Enter the shift value
3
The encrypted string is vkh lv olvwhqlqj
```



```
Enter the encrypted string
vkh lv olvwhqlqj
Enter the value by which string was shifted
3
The decrypted string is she is listening
```

Aim:- To implement Monoalphabetic Substitution Cipher.

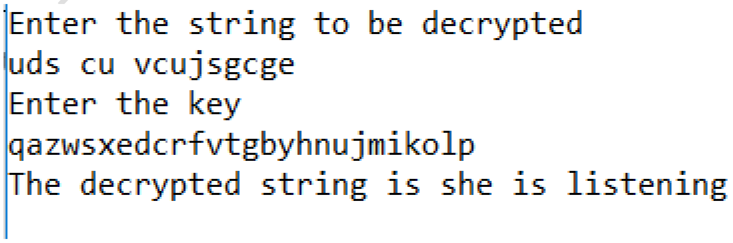
```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char ar[26];
    char br[100010];
    cout<<"Enter the string to be encrypted\n";
    cin.getline(br,100000);
    cout<<"Enter the key\n";
    cin>>ar;
    int l=strlen(br);
    for(int i=0;i<l;i++)
    {
        if(br[i]==' ')
            continue;
        br[i]=ar[br[i]-'a'];
    }
    cout<<"The encrypted string is "<<br<<endl;
}
```



```
Enter the string to be encrypted
she is listening
Enter the key
qazwsxedcrfvtgbyhnujmikolp
The encrypted string is uds cu vcujsgcge
```

Aim:- To implement Monoalphabetic Substitution Decipher.

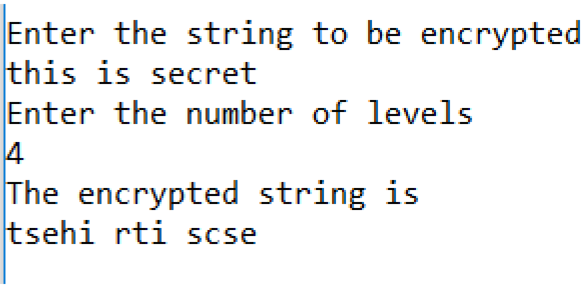
```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char ar[100];
    char br[100010];
    cout<<"Enter the string to be decrypted\n";
    cin.getline(br,100000);
    cout<<"Enter the key\n";
    cin>>ar;
    int l=strlen(br);
    char cr[100];
    for(int i=0;i<26;i++) cr[ar[i]-'a']='a'+i;

    for(int i=0;i<l;i++)
    {
        if(br[i]==' ')
            continue;
        br[i]=cr[br[i]-'a'];
    }
    cout<<"The decrypted string is "<<br<<endl;
}
```



```
Enter the string to be decrypted
uds cu vcujsgcge
Enter the key
qazwsxedcrfvtgbyhnujmikolp
The decrypted string is she is listening
```

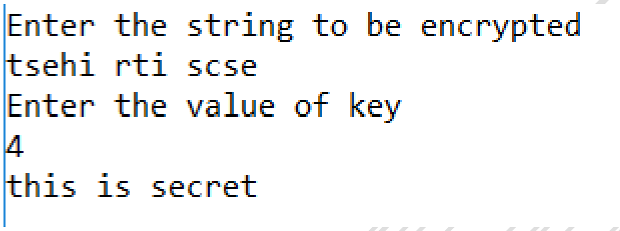
Aim:- To implement Rail-Fence Transposition Cipher.

```
import java.util.*;
class RailFenceBasic{
 int depth;
 String Encryption(String plainText,int depth)throws Exception
 {
  int r=depth,len=plainText.length();
  int c=len/depth;
  char mat[][]=new char[r][c];
  int k=0;

  String cipherText="";

  for(int i=0;i< c;i++)
  {
   for(int j=0;j< r;j++)
   {
    if(k!=len)
     mat[j][i]=plainText.charAt(k++);
    else
     mat[j][i]='X';
   }
  }
  for(int i=0;i< r;i++)
  {
   for(int j=0;j< c;j++)
   {
    cipherText+=mat[i][j];
   }
  }
  return cipherText;
 }


 String Decryption(String cipherText,int depth)throws Exception
 {
  int r=depth,len=cipherText.length();
  int c=len/depth;
  char mat[][]=new char[r][c];
  int k=0;

  String plainText="";
  for(int i=0;i< r;i++)
  {
   for(int j=0;j< c;j++)
   {
    mat[i][j]=cipherText.charAt(k++);
   }
  }
  for(int i=0;i< c;i++)
  {
   for(int j=0;j< r;j++)
   {
    plainText+=mat[j][i];
   }
  }

  return plainText;
 }
}

class RailFence{
 public static void main(String args[])throws Exception
 {
  RailFenceBasic rf=new RailFenceBasic();
		Scanner scn=new Scanner(System.in);
		int depth;

		String plainText,cipherText,decryptedText;

		System.out.println("Enter plain text:");
		plainText=scn.nextLine();

		System.out.println("Enter depth for Encryption:");
		depth=scn.nextInt();

  cipherText=rf.Encryption(plainText,depth);
  System.out.println("Encrypted text is:\n"+cipherText);

		decryptedText=rf.Decryption(cipherText, depth);

  System.out.println("Decrypted text is:\n"+decryptedText);
 }
```



```
Enter the string to be encrypted
this is secret
Enter the number of levels
4
The encrypted string is
tsehi rti scse
```

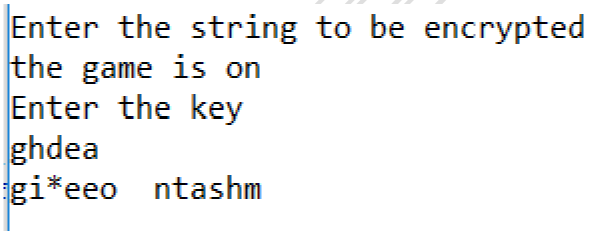
}



```
Enter the string to be encrypted
tsehi rti scse
Enter the value of key
4
this is secret
```

Aim:- To implement Columnar Transposition Cipher.

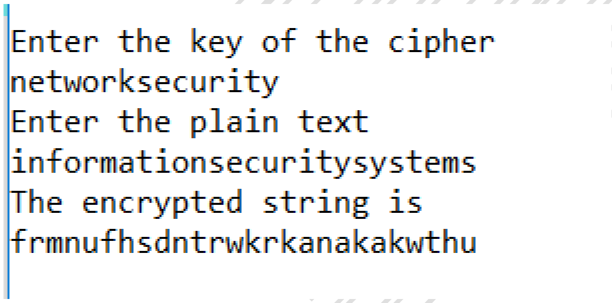
```
def split_len(seq, length):
    return [seq[i:i + length] for i in range(0, len(seq), length)]


def encode(key, plaintext):


    order = {
        int(val): num for num, val in enumerate(key)
    }
    print(order)
    ciphertext = ''
    for index in sorted(order.keys()):
        for part in split_len(plaintext, len(key)):
            print(part)
            try:
                print(index)
                ciphertext += part[order[index]]
            except IndexError:
                continue


    return ciphertext


print(encode('3214', 'IHAVETWOCATS'))
```



```
Enter the string to be encrypted
the game is on
Enter the key
ghdea
gi*eeo  ntashm
```

Aim:- To implement Columnar Transposition Decipher.

```
def split_len(seq, length):
    return [seq[i:i + length] for i in range(0, len(seq), length)]

def encode(key, plaintext):

    order = {
        int(val): num for num, val in enumerate(key)
    }
    print(order)
    ciphertext = ''
    for index in sorted(order.keys()):
        for part in split_len(plaintext, len(key)):
            print(part)
            try:
                print(index)
                ciphertext += part[order[index]]
            except IndexError:
                continue

    return ciphertext

print(encode('3214', 'IHAVETWOCATS'))
```



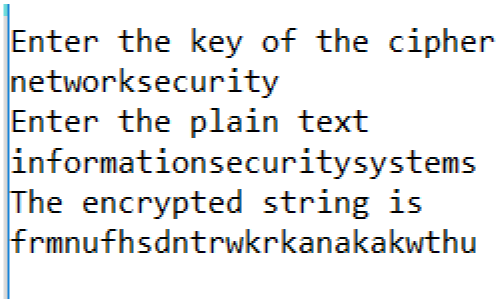
```
Enter the key of the cipher
networksecurity
Enter the plain text
informationsecuritysystems
The encrypted string is
frmnufhsdntrwkrkanakakwthu
```

Aim:- To implement Play-fair cipher.

```
import java.util.Scanner;

public class PlayfairCipherEncryption
{
   private String KeyWord        = new String();
   private String Key            = new String();
   private char   matrix_arr[][] = new char[5][5];

   public void setKey(String k)
   {
      String K_adjust = new String();
      boolean flag = false;
      K_adjust = K_adjust + k.charAt(0);
      for (int i = 1; i < k.length(); i++)
      {
         for (int j = 0; j < K_adjust.length(); j++)
         {
            if (k.charAt(i) == K_adjust.charAt(j))
            {
               flag = true;
            }
         }
         if (flag == false)
            K_adjust = K_adjust + k.charAt(i);
         flag = false;
      }
      KeyWord = K_adjust;
   }

   public void KeyGen()
   {
      boolean flag = true;
      char current;
      Key = KeyWord;
      for (int i = 0; i < 26; i++)
      {
         current = (char) (i + 97);
         if (current == 'j')
            continue;
            for (int j = 0; j < KeyWord.length(); j++)
            {
                if (current == KeyWord.charAt(j))
                {
                    flag = false;
                    break;
                }
            }
            if (flag)
                Key = Key + current;
            flag = true;
        }
        System.out.println(Key);
        matrix();
    }

    private void matrix()
    {
        int counter = 0;
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                matrix_arr[i][j] = Key.charAt(counter);
                System.out.print(matrix_arr[i][j] + " ");
                counter++;
            }
            System.out.println();
        }
    }

    private String format(String old_text)
    {
        int i = 0;
        int len = 0;
        String text = new String();
        len = old_text.length();
        for (int tmp = 0; tmp < len; tmp++)
        {
            if (old_text.charAt(tmp) == 'j')
            {
```

```
            text = text + 'i';
        }
        else
            text = text + old_text.charAt(tmp);
    }
    len = text.length();
    for (i = 0; i < len; i = i + 2)
    {
        if (text.charAt(i + 1) == text.charAt(i))
        {
            text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
        }
    }
    return text;
}

private String[] Divid2Pairs(String new_string)
{
    String Original = format(new_string);
    int size = Original.length();
    if (size % 2 != 0)
    {
        size++;
        Original = Original + 'x';
    }
    String x[] = new String[size / 2];
    int counter = 0;
    for (int i = 0; i < size / 2; i++)
    {
        x[i] = Original.substring(counter, counter + 2);
        counter = counter + 2;
    }
    return x;
}

public int[] GetDiminsions(char letter)
{
    int[] key = new int[2];
    if (letter == 'j')
        letter = 'i';
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (matrix_arr[i][j] == letter)
            {
                key[0] = i;
                key[1] = j;
                break;
            }
        }
    }
    return key;
}

public String encryptMessage(String Source)
{
    String src_arr[] = Divid2Pairs(Source);
    String Code = new String();
    char one;
    char two;
    int part1[] = new int[2];
    int part2[] = new int[2];
    for (int i = 0; i < src_arr.length; i++)
    {
        one = src_arr[i].charAt(0);
        two = src_arr[i].charAt(1);
        part1 = GetDiminsions(one);
        part2 = GetDiminsions(two);
        if (part1[0] == part2[0])
        {
            if (part1[1] < 4)
                part1[1]++;
            else
                part1[1] = 0;
            if (part2[1] < 4)
                part2[1]++;
            else
                part2[1] = 0;
        }
        else if (part1[1] == part2[1])
        {
            if (part1[0] < 4)
                part1[0]++;
            else
                part1[0] = 0;
            if (part2[0] < 4)
                part2[0]++;
            else
                part2[0] = 0;
        }
        else
        {
            int temp = part1[1];
            part1[1] = part2[1];
            part2[1] = temp;
        }
        Code = Code + matrix_arr[part1[0]][part1[1]]
                + matrix_arr[part2[0]][part2[1]];
    }
    return Code;
}

public static void main(String[] args)
{
    PlayfairCipherEncryption x = new PlayfairCipherEncryption();
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter a keyword:");
    String keyword = sc.next();
    x.setKey(keyword);
    x.KeyGen();
    System.out
            .println("Enter word to encrypt: (Make sure length of message is even)");
    String key_input = sc.next();
    if (key_input.length() % 2 == 0)
    {
        System.out.println("Encryption: " + x.encryptMessage(key_input));
    }
    else
    {
        System.out.println("Message length should be even");
    }
    sc.close();
```

```
   }
```



```
Enter the key of the cipher
networksecurity
Enter the plain text
informationsecuritysystems
The encrypted string is
frmnufhsdntrwkrkanakakwthu
```

```
}
```

Aim:- To implement playfair Cipher.

```
import java.util.Scanner;

public class PlayfairCipherDecryption
{
    private String KeyWord        = new String();
    private String Key            = new String();
    private char   matrix_arr[][] = new char[5][5];

    public void setKey(String k)
    {
        String K_adjust = new String();
        boolean flag = false;
        K_adjust = K_adjust + k.charAt(0);
        for (int i = 1; i < k.length(); i++)
        {
            for (int j = 0; j < K_adjust.length(); j++)
            {
                if (k.charAt(i) == K_adjust.charAt(j))
                {
                    flag = true;
                }
            }
            if (flag == false)
                K_adjust = K_adjust + k.charAt(i);
            flag = false;
        }
        KeyWord = K_adjust;
    }

    public void KeyGen()
    {
        boolean flag = true;
        char current;
        Key = KeyWord;
        for (int i = 0; i < 26; i++)
        {
```

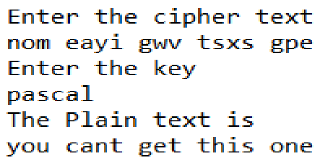
```
        current = (char) (i + 97);
        if (current == 'j')
            continue;
        for (int j = 0; j < KeyWord.length(); j++)
        {
            if (current == KeyWord.charAt(j))
            {
                flag = false;
                break;
            }
        }
        if (flag)
            Key = Key + current;
        flag = true;
    }
    System.out.println(Key);
    matrix();
}

private void matrix()
{
    int counter = 0;
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            matrix_arr[i][j] = Key.charAt(counter);
            System.out.print(matrix_arr[i][j] + " ");
            counter++;
        }
        System.out.println();
    }
}

private String format(String old_text)
{
    int i = 0;
```

```
        int len = 0;
        String text = new String();
        len = old_text.length();
        for (int tmp = 0; tmp < len; tmp++)
        {
            if (old_text.charAt(tmp) == 'j')
            {
                text = text + 'i';
            }
            else
                text = text + old_text.charAt(tmp);
        }
        len = text.length();
        for (i = 0; i < len; i = i + 2)
        {
            if (text.charAt(i + 1) == text.charAt(i))
            {
                text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
            }
        }
        return text;
    }

    private String[] Divid2Pairs(String new_string)
    {
        String Original = format(new_string);
        int size = Original.length();
        if (size % 2 != 0)
        {
            size++;
            Original = Original + 'x';
        }
        String x[] = new String[size / 2];
        int counter = 0;
        for (int i = 0; i < size / 2; i++)
        {
            x[i] = Original.substring(counter, counter + 2);
            counter = counter + 2;
        }
        return x;
    }

    public int[] GetDiminsions(char letter)
    {
        int[] key = new int[2];
        if (letter == 'j')
            letter = 'i';
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                if (matrix_arr[i][j] == letter)
                {
                    key[0] = i;
                    key[1] = j;
                    break;
                }
            }
        }
        return key;
    }

    public String encryptMessage(String Source)
    {
        String src_arr[] = Divid2Pairs(Source);
        String Code = new String();
        char one;
        char two;
        int part1[] = new int[2];
        int part2[] = new int[2];
        for (int i = 0; i < src_arr.length; i++)
        {
            one = src_arr[i].charAt(0);
            two = src_arr[i].charAt(1);
```

```
            part1 = GetDiminsions(one);
            part2 = GetDiminsions(two);
            if (part1[0] == part2[0])
            {
                if (part1[1] < 4)
                    part1[1]++;
                else
                    part1[1] = 0;
                if (part2[1] < 4)
                    part2[1]++;
                else
                    part2[1] = 0;
            }
            else if (part1[1] == part2[1])
            {
                if (part1[0] < 4)
                    part1[0]++;
                else
                    part1[0] = 0;
                if (part2[0] < 4)
                    part2[0]++;
                else
                    part2[0] = 0;
            }
            else
            {
                int temp = part1[1];
                part1[1] = part2[1];
                part2[1] = temp;
            }
            Code = Code + matrix_arr[part1[0]][part1[1]]
                    + matrix_arr[part2[0]][part2[1]];
        }
        return Code;
    }

    public String decryptMessage(String Code)
```

```
{
    String Original = new String();
    String src_arr[] = Divid2Pairs(Code);
    char one;
    char two;
    int part1[] = new int[2];
    int part2[] = new int[2];
    for (int i = 0; i < src_arr.length; i++)
    {
        one = src_arr[i].charAt(0);
        two = src_arr[i].charAt(1);
        part1 = GetDiminsions(one);
        part2 = GetDiminsions(two);
        if (part1[0] == part2[0])
        {
            if (part1[1] > 0)
                part1[1]--;
            else
                part1[1] = 4;
            if (part2[1] > 0)
                part2[1]--;
            else
                part2[1] = 4;
        }
        else if (part1[1] == part2[1])
        {
            if (part1[0] > 0)
                part1[0]--;
            else
                part1[0] = 4;
            if (part2[0] > 0)
                part2[0]--;
            else
                part2[0] = 4;
        }
        else
        {
```

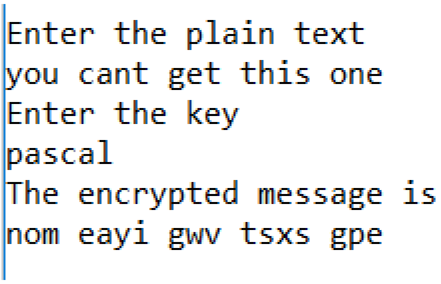
```
        int temp = part1[1];
        part1[1] = part2[1];
        part2[1] = temp;
      }
      Original = Original + matrix_arr[part1[0]][part1[1]]
          + matrix_arr[part2[0]][part2[1]];
    }
    return Original;
  }

  public static void main(String[] args)
  {
    PlayfairCipherDecryption x = new PlayfairCipherDecryption();
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter a keyword:");
    String keyword = sc.next();
    x.setKey(keyword);
    x.KeyGen();
    System.out
        .println("Enter word to encrypt: (Make sure length of message is even)");
    String key_input = sc.next();
    if (key_input.length() % 2 == 0)
    {
      System.out.println("Encryption: " + x.encryptMessage(key_input));
      System.out.println("Decryption: "
          + x.decryptMessage(x.encryptMessage(key_input)));
    }
    else
    {
      System.out.println("Message length should be even");
    }
    sc.close();
  }
}
```



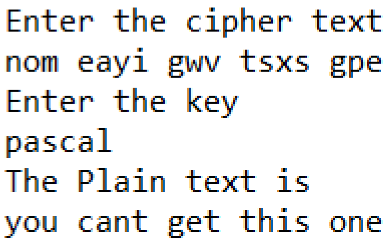
```
Enter the cipher text
nom eayi gwv tsxs gpe
Enter the key
pascal
The Plain text is
you cant get this one
```

Aim:- To implement Vigenere Cipher.

```
def encrypt(plaintext, key):
    key_length = len(key)
    key_as_int = [ord(i) for i in key]
    plaintext_int = [ord(i) for i in plaintext]
    ciphertext = ''
    for i in range(len(plaintext_int)):
        value = (plaintext_int[i] + key_as_int[i % key_length]) % 26
        ciphertext += chr(value + 65)
    return ciphertext



def decrypt(ciphertext, key):
    key_length = len(key)
    key_as_int = [ord(i) for i in key]
    ciphertext_int = [ord(i) for i in ciphertext]
    plaintext = ''
    for i in range(len(ciphertext_int)):
        value = (ciphertext_int[i] - key_as_int[i % key_length]) % 26
        plaintext += chr(value + 65)
    return plaintext
print(encrypt('saiganesh','sai'))
print(decrypt('dfsdfs','fs'))
```



```
Enter the plain text
you cant get this one
Enter the key
pascal
The encrypted message is
nom eayi gwv tsxs gpe
```



```
Enter the cipher text
nom eayi gwv tsxs gpe
Enter the key
pascal
The Plain text is
you cant get this one
```

Aim:- To implement DES

```
# include<bits/stdc++.h>
using namespace std;

void round_two(vector<int>&EP,vector<int>&p_four,vector<int>&k2,
                vector<int>&new_PT)
{
    cout<<"\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\nRound two begins\n";
    cout<<"\n\nThe new plain text becomes\n";
    for(int i=0;i<8;i++)
    cout<<new_PT[i];
    vector<int>left_four(4);
    vector<int>right_four(4);

    for(int i=0;i<4;i++)
    left_four[i]=new_PT[i];

    for(int i=0;i<4;i++)
    right_four[i]=new_PT[i+4];


    cout<<"\n\nThe new plain text again becomes after applying EP\n";
    for(int i=0;i<4;i++)
    {
            new_key_second.push_back(right_four[EP[i]-1]);
    }
    for(int i=0;i<4;i++)
    {
            new_key_second.push_back(right_four[EP[i+4]-1]);
    }

    vector<vector<int>>s_box_zero ={ {1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
    vector<vector<int>>s_box_one={{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};

    int c1,r1,c2,r2;
    r1=2*new_PT[0] + 1*new_PT[3];
    c1=2*new_PT[1] + 1*new_PT[2];

    r2=2*new_PT[4] + 1*new_PT[7];
    c2=2*new_PT[5] + 1*new_PT[6];

    int a=s_box_zero[r1][c1];
```

```
int b=s_box_one[r2][c2];
vector<int>s_box_first_res;



s_box_first_res.push_back(0);
s_box_first_res.push_back(0);
s_box_first_res.push_back(0);
s_box_first_res.push_back(0);

cout<<endl;
swap(s_box_first_res[0],s_box_first_res[1]);
swap(s_box_first_res[2],s_box_first_res[3]);

cout<<"\nAfter combining from the sboxes\n";
for(unsigned int i=0;i<s_box_first_res.size();i++)
cout<<s_box_first_res[i];


vector<int>temps;
cout<<endl;
for(int i=0;i<4;i++)
{
        p_four[i]--;
}
for(int i=0;i<4;i++){temps.push_back(s_box_first_res[p_four[i]]);}

cout<<endl<<"\nAfter applying p4\n";
for(unsigned int i=0;i<temps.size();i++)
cout<<temps[i];

cout<<"\n\nAfter exoring with 4 left bits of PT\n";
for(int i=0;i<4;i++)
{
        temps[i]=temps[i]^left_four[i];
        cout<<temps[i];
}

for(int i=0;i<4;i++)
cout<<right_four[i];

cout<<"\n\nFinal answer is\n";
vector<int>round_2_input;
```

```
	for(int i=0;i<4;i++)
	round_2_input.push_back(left_four[i]);

	for(int i=0;i<4;i++)
	round_2_input.push_back(right_four[i]);


	for(int i=0;i<8;i++)
	cout<<round_2_input[i];

}
int main()
{
	vector<int>PT = {1,0,1,1,1,1,0,1};
	vector<int>ten_bit_key = {1,0,1,0,0,0,0,0,1,0};
	vector<int>p_ten =	{3,5,2,7,4,10,1,9,8,6};
	vector<int>p_eight = {6,3,7,4,8,5,10,9};
	vector<int>IP = {2,6,3,1,4,8,5,7};
	vector<int>EP = {4,1,2,3,2,3,4,1};
	vector<int>p_four = {2,4,3,1};



	vector<int>new_key;

	for(int i=0;i<10;i++){new_key.push_back(ten_bit_key[p_ten[i]-1]);}
	cout<<"after applying P 10 \n";
	for(int i=0;i<10;i++)
	cout<<new_key[i];


	// performing left shift 1

	int temp = new_key[0];
	for(int i=0;i<5;i++) {new_key[i]= new_key[i+1];}

	new_key[4]=temp;

	temp =new_key[5];
	for(int i=5;i<10;i++){new_key[i]= new_key[i+1];}

	new_key[9]=temp;

	cout<<"\n\nAfter applying left shift 1 time\n";
```

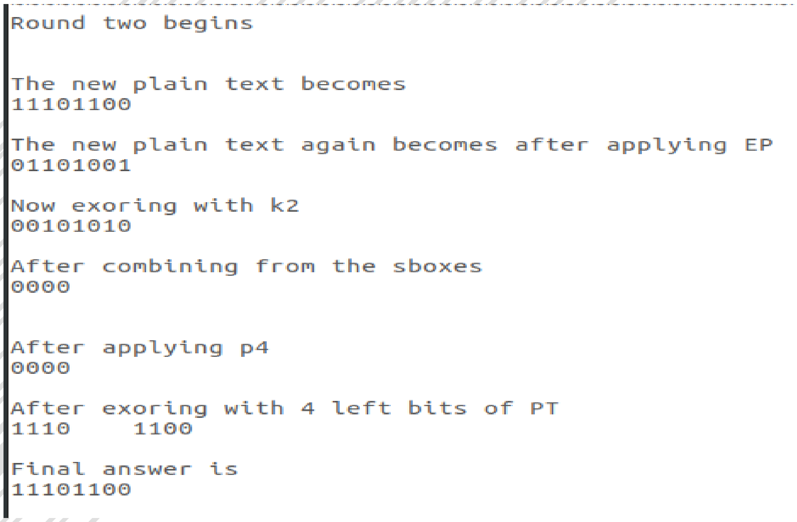
```
for(int i=0;i<10;i++)
cout<<new_key[i];

vector<int>k1;
cout<<"\n\nAfter applying P 8 K1 is: \n";

for(int i=0;i<8;i++) {k1.push_back(new_key[p_eight[i]-1]);}


for(int i=0;i<8;i++)
cout<<k1[i];

temp = new_key[0];
for(int i=0;i<5;i++) {new_key[i]= new_key[i+1];}

new_key[4]=temp;

temp = new_key[0];
for(int i=0;i<5;i++) {new_key[i]= new_key[i+1];}

new_key[4]=temp;

temp =new_key[5];
for(int i=5;i<10;i++)
{
        new_key[i]= new_key[i+1];
}
new_key[9]=temp;

temp =new_key[5];
for(int i=5;i<10;i++)
{
        new_key[i]= new_key[i+1];
}
new_key[9]=temp;

cout<<"\n\nAfter applying left shift 2 times on new key\n";
for(int i=0;i<10;i++)
cout<<new_key[i];


vector<int>k2;
cout<<"\n\nAfter applying P 8 K2 is: \n";
```

```
for(int i=0;i<8;i++)
{
        k2.push_back(new_key[p_eight[i]-1]);
}

for(int i=0;i<8;i++)
cout<<k2[i];

vector<int>new_PT;


for(int i=0;i<8;i++)
{
        new_PT.push_back(PT[IP[i]-1]);
}

cout<<"\n\nThe new plain text becomes\n";
for(int i=0;i<8;i++)
cout<<new_PT[i];

vector<int>left_four(4);
vector<int>right_four(4);
vector<int>new_key_second;

for(int i=0;i<4;i++)
left_four[i]=new_PT[i];

for(int i=0;i<4;i++)
right_four[i]=new_PT[i+4];

for(int i=0;i<4;i++)
{
        new_key_second.push_back(new_PT[EP[i+4]+4-1]);
}




cout<<"\n\nThe new plain text again becomes after applying EP\n";

for(int i=0;i<4;i++)
new_PT[i+4]=new_key_second[i];

for(int i=0;i<8;i++)
```

```
cout<<new_PT[i];

cout<<"\n\nNow exoring with k1\n";
for(int i=0;i<8;i++)
new_PT[i]=new_PT[i] ^ k1[i];

for(int i=0;i<8;i++)
cout<<new_PT[i];

vector<vector<int>>s_box_zero ={ {1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
vector<vector<int>>s_box_one={{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};

int c1,r1,c2,r2;
r1=2*new_PT[0] + 1*new_PT[3];
c1=2*new_PT[1] + 1*new_PT[2];

r2=2*new_PT[4] + 1*new_PT[7];
c2=2*new_PT[5] + 1*new_PT[6];

int a=s_box_zero[r1][c1];
int b=s_box_one[r2][c2];
vector<int>s_box_first_res;
swap(s_box_first_res[0],s_box_first_res[1]);
swap(s_box_first_res[2],s_box_first_res[3]);

cout<<"\nAfter combining from the sboxes\n";
for(unsigned int i=0;i<s_box_first_res.size();i++)
cout<<s_box_first_res[i];


vector<int>temps;
cout<<endl;
for(int i=0;i<4;i++){p_four[i]--;}


cout<<endl<<"After applying p4\n";
for(unsigned int i=0;i<temps.size();i++)
cout<<temps[i];

cout<<"\n\nAfter exoring with 4 left bits of PT\n";
for(int i=0;i<4;i++)
{
	temps[i]=temps[i]^left_four[i];
	cout<<temps[i];
```

```
    }

    cout<<"\t";
    for(int i=0;i<4;i++)
    cout<<right_four[i];

    cout<<"\n\nnew plain text to go to round 2 is\n";

    vector<int>round_2_input;
    for(int i=0;i<4;i++)
    round_2_input.push_back(right_four[i]);

    for(int i=0;i<4;i++)
    round_2_input.push_back(temps[i]);

    for(int i=0;i<8;i++)
    cout<<round_2_input[i];

    round_two(EP,p_four,k2,round_2_input);
}
```

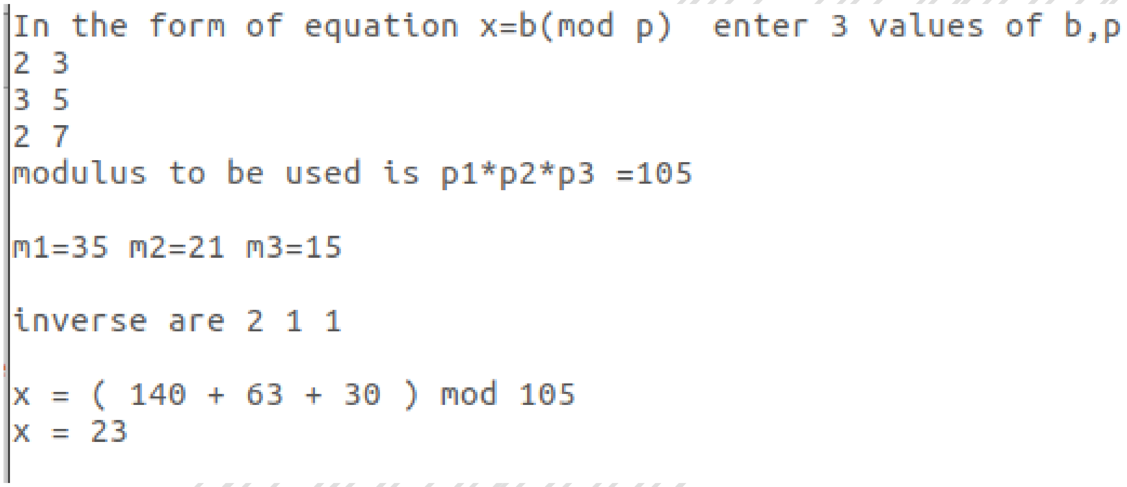
OUTPUT



```
Round two begins


The new plain text becomes
11101100

The new plain text again becomes after applying EP
01101001

Now exoring with k2
00101010

After combining from the sboxes
0000


After applying p4
0000

After exoring with 4 left bits of PT
1110    1100

Final answer is
11101100
```

Aim:- To implement CRT

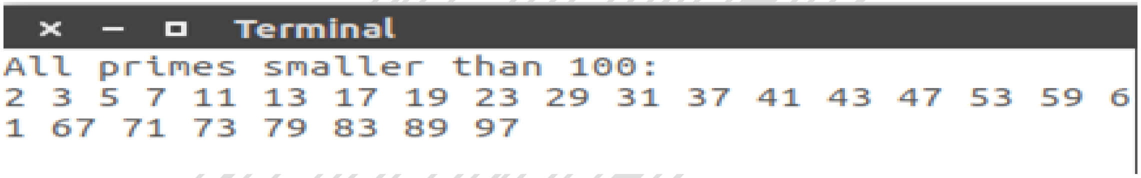
```
# include<bits/stdc++.h>
using namespace std;
int gcdExtended(int a, int b, int &x, int &y)
{

   if (a == 0)
   {
      x = 0, y = 1;
      return b;
   }

   int x1, y1;
   int gcd = gcdExtended(b%a, a, x1, y1);
   x = y1 - (b/a) * x1;
   y = x1;

   return gcd;
}

int modInverse(int a, int m)
{
   int x, y;
    gcdExtended(a, m, x, y);

      // m is added to handle negative x
      int res = (x%m + m) % m;
    return res; }
int main(){

                int b1,b2,b3,p1,p2,p3,x;

                cout<<"In the form of equation x=b(mod p)  enter 3 values of b
                    and p”;
                cin>>b1>>p1>>b2>>p2>>b3>>p3;
                int mod=p1*p2*p3;
                cout<<"modulus to be used is p1*p2*p3  <<mod<<endl<<endl;
                int m1,m2,m3;
                m1=mod/p1; m2=mod/p2; m3=mod/p3;
                cout<<"m1="<<m1<<" m2="<<m2<<" m3="<<m3<<endl<<endl;
                int inverse_1,inverse_2,inverse_3;
                inverse_1 = modInverse(m1,p1);
                inverse_2 = modInverse(m2,p2);
                inverse_3 = modInverse(m3,p3);


                cout<<"inverse are "<<inverse_1<<" "<<inverse_2<<" "<<inverse_3;

                cout<<"x = ( "<<b1*m1*inverse_1<<" + "<< b2*m2*inverse_2 <<"
            + "<< b3*m3*inverse_3<<" ) mod "<<mod;

                x=(b1*m1*inverse_1+b2*m2*inverse_2+b3*m3*inverse_3)%mod;
                cout<<"\nx = "<<x;
}
```



```
In the form of equation x=b(mod p)  enter 3 values of b,p
2 3
3 5
2 7
modulus to be used is p1*p2*p3 =105

m1=35 m2=21 m3=15

inverse are 2 1 1

x = ( 140 + 63 + 30 ) mod 105
x = 23
```

Aim:- To implement Fermats Theorem

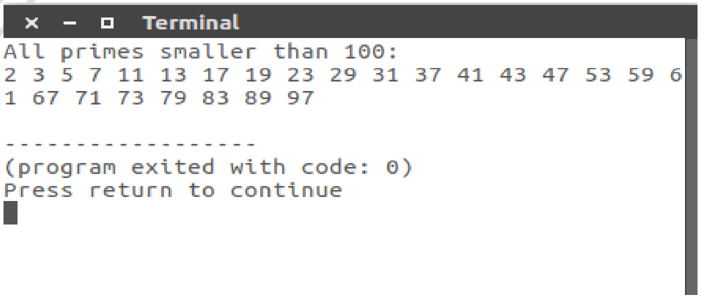
```
import random
def fermat_test(n, k):
    if n == 2:
        return True

    if n % 2 == 0:
        return False

    for i in range(k):
        a = random.randint(1, n - 1)

        if pow(a, n - 1) % n != 1:
            return False
    return True
if (fermat_test(2,7)==True):
    print('prime')
else:
    print('not prime')
```



```
All primes smaller than 100:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 6
1 67 71 73 79 83 89 97
```

Aim:- To implement Miller-Rabin Primality test

```
 import random
def miller_rabin(n, k=10):
    if n == 2:
        return True
    if not n & 1:
        return False

    def check(a, s, d, n):
        x = pow(a, d, n)
        if x == 1:
            return True
        for i in range(s - 1):
            if x == n - 1:
                return True
            x = pow(x, 2, n)
        return x == n - 1

    s = 0
    d = n - 1

    while d % 2 == 0:
        d >>= 1
        s += 1

    for i in range(k):
        a = random.randrange(2, n - 1)
        if not check(a, s, d, n):
            return False
    return True

if(miller_rabin(4,7)==True):
   print('prime')
else:
   print('Not true')
```



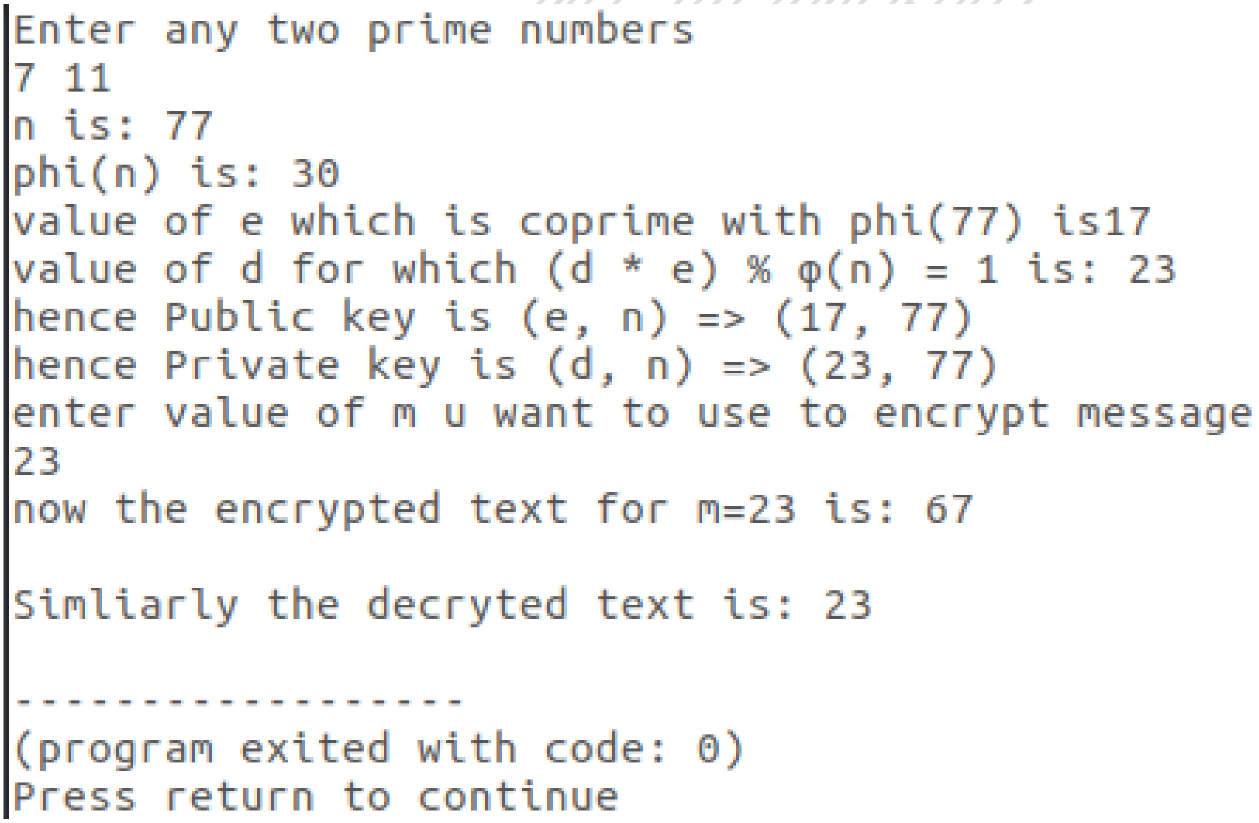
```
All primes smaller than 100:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 6
1 67 71 73 79 83 89 97

------------------
(program exited with code: 0)
Press return to continue
```

Aim:- To implement RSA

```
def mul(a,b):
    return a*b

def gcd(x, y):
   while(y):
       x, y = y, x % y
   return x

def lcm(x, y):
   lcm = (x*y)//gcd(x,y)
   return lcm
def totient(a,b):
    return lcm( (a-1), (b-1) )

def inverse(p,q):
    def egcd(a, b):
        if a == 0:
            return (b, 0, 1)
        else:
            g, y, x = egcd(b % a, a)
            return (g, x - (b // a) * y, y)

    def modinv(j, u):
        g, x, y = egcd(j, u)
        if g != 1:
            raise Exception('modular inverse does not exist')
        else:
            return x % u
    return modinv(p,q)
x = int(input())
y = int(input())
```

```
totp = totient(x,y)
print(totp)
k = int(input("input random number coprime to "))
key = inverse(k,totp)
print(key)
message = int(input('enter the number to encrypt'))
encrypted = ((message**k)%mul(x,y))
print(encrypted)
decrypted = ((1211**key)%mul(x,y))
print(decrypted)
```

OUTPUT



```
Enter any two prime numbers
7 11
n is: 77
phi(n) is: 30
value of e which is coprime with phi(77) is17
value of d for which (d * e) % φ(n) = 1 is: 23
hence Public key is (e, n) => (17, 77)
hence Private key is (d, n) => (23, 77)
enter value of m u want to use to encrypt message
23
now the encrypted text for m=23 is: 67

Simliarly the decryted text is: 23

------------------
(program exited with code: 0)
Press return to continue
```