Substitution Cipher

Source Code

```
#include<stdio.h>

 int main()

{
  char message[100], ch, str;
  int i, key, x;
  printf("Enter a message : ");
  gets(message);
  printf("Enter key: ");
  scanf("%d", &key);

 printf("\nPlease choose following options:\n");
 printf("1 = Encrypt the string.\n");
 printf("2 = Decrypt the string.\n");
 scanf("%d", &x);

 //using switch case statements

  switch(x)
 {
 case 1:
     for(i = 0; message[i] != '\0'; ++i)
      {
         ch = message[i];
         if(ch >= 'a' && ch <= 'z')
         {
            ch = ch + key;
            if(ch > 'z')
            {
               ch = ch - 'z' + 'a' - 1;
            }
            message[i] = ch;

         }
         else if(ch >= 'A' && ch <= 'Z')
         {
            ch = ch + key;
            if(ch > 'Z')
            {
               ch = ch - 'Z' + 'A' - 1;
            }
            message[i] = ch;
         }
      }

      printf("Encrypted message: %s", message);
      break;

 case 2:
    for(i = 0; message[i] != '\0'; ++i)
    {
      ch = message[i];
      if(ch >= 'a' && ch <= 'z')
      {
         ch = ch - key;
         if(ch < 'a')
         {
            ch = ch + 'z' - 'a' + 1;
         }
```

```
            message[i] = ch;
        }
        else if(ch >= 'A' && ch <= 'Z')
        {
        ch = ch - key;
        if(ch < 'A')
        {
            ch = ch + 'Z' - 'A' + 1;
        }
        message[i] = ch;
        }
      }
        printf("Decrypted message: %s", message);
      break;

  default:
     printf("\nError\n");
  }

return 0;
}
```

Sample Output

```
Enter a message : I am studying Data Encryption
Enter key: 4
Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
1
Encrypted message: M eq wxyhcmrk Hexe Irgvctxmsr


Enter a message : M eq wxyhcmrk Hexe Irgvctxmsr
Enter key: 4
Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
2
Decrypted message: I am studying Data Encryption
—----------------------------------------------------------------
```

RSA algorithm

Source code

```
#include <stdio.h>

#include<stdlib.h>

int gcd(int a,int b)
{
int c;
while(a!=b)
{
if(a<b)
{
c=a;a=b;b=c;
}
a-=b;
}
return a;
}

int mod(int m,int e,int n)
{
int a=1;
while(e)
{
a=(a*m)%n;
e--;
}
return a;
}

int main()
{
int p,q,n,e,m,c,d,x,z;
int en[100],de[100],j=0;
printf("\nEnter the value of P & Q\n");
scanf("%d%d",&p,&q);
n=p*q;
z=(p-1)*(q-1);
for(e=1;e<n;e++)
{
if(gcd(e,z)==1)
```

```
{
en[j]=e;
printf(" %d",en[j++]);
}
}
printf("\nChoose e\n");
scanf("%d",&e);

if(gcd(e,z)!=1)
{
printf("\nThe value not from list\n");
exit(0);
}
printf("Enter the message(integer value) to be encrypted:\n");
scanf("%d",&m);
printf("Before encryption:%d\n",m);
c=mod(m,e,n);
printf("After encryption:%d\n",c);

printf("The possible Decryption keys Are:");
for(d=0;d<n;d++)
{
if((d*e)%z==1)
{
de[j]=d;
printf(" %d",de[j++]);
}
}
printf("\nChoose D\n");
scanf("%d",&d);
x=mod(c,d,n);
printf("After decryption=%d\n",x);
return 0;
}
```

Example

Step 1: Select two large prime numbers, p, and **q**.

p = 7

q = 11

Step 2: Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.

First, we calculate

n = p x q

n = 7 x 11

n = 77

Step 3: Choose a number **e** less that **n**, such that n is relatively prime to **(p - 1) x (q -1).** It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n), gcd (e, d (n)) =1.

Second, we calculate

z = (p - 1) x (q-1)

z = (7 - 1) x (11 - 1)

z = 6 x 10

z = 60

Let us now choose relative prime **e of 60 as 7.**

Thus the public key is <e, n> = (7, 77)

Step 4: A plaintext message **m** is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.

To find ciphertext from the plain text following formula is used to get ciphertext C.

M=9 e=7

$C = m^e \bmod n$

$C = 9^7 \bmod 77$

C = 37

Step 5: The private key is <d, n>. To determine the private key, we use the following formula d such that:

$de \bmod \{(p - 1) \times (q - 1)\} = 1$

7d mod 60 = 1, which gives d = 43 … series z+1,2Z+1,3Z+1…. 61,121,181,241,301,361..

The private key is <d, n> = (43, 77)

Step 6: A ciphertext message **c** is decrypted using private key <d, n>. To calculate plain text **m** from the ciphertext c following formula is used to get plain text m.

$m = c^d \bmod n$

$m = 37^{43} \bmod 77$

m = 9

In this example, Plain text = 9 and the ciphertext = 37

Sample Output

Enter the value of P & Q

7 11

1 7 11 13 17 19 23 29 31 37 41 43 47 49 53 59 61 67 71 73

Choose e

7

Enter the message (integer value) to be encrypted:

9

Before encryption:9

After encryption:37

The possible Decryption Keys Are: 43

Choose D

43

After decryption=9

Password Strength

```
int main()
{
  int i,n,a=0,d=0,s=0;
  char p[10];
  printf("Enter the Password: ");
  gets(p);
  n=strlen(p);

  if(n>=6)
  {
     for(i=0;i<n;i++)
     {
        if(isalpha(p[i]))
        {
           a+=1;
        }
        else if(isdigit(p[i]))
        {
           d+=1;
        }
        else
        {
           s+=1;
        }
     }
     if(a>=1 && d>=1 && s>=1)
     {
        printf("Strong Password");

     }
     else if((a>=1 && d>=1) || (a>=1 && s>1) || (d>=1 && s>=1))
     {
        printf("Moderate Password");
     }
     else
     {
        printf("Weak Password");
     }
  }
  else
  {
```

```
    printf("Invalid Password");
  }

}
```

Sample Output

Enter password : aw1

Invalid Password

Enter password : adckex

Weak Password

Enter password : abc123

Moderate Password

Enter password : abc2#@

Strong Password

Rail fence

Source Code

```
#include<stdio.h>
#include<string.h>

void encryptMsg(char msg[], int key)
{
  int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0;
  char railMatrix[key][msgLen];

  for(i = 0; i < key; ++i)
     for(j = 0; j < msgLen; ++j)
        railMatrix[i][j] = '\n';

  for(i = 0; i < msgLen; ++i)
  {
     railMatrix[row][col++] = msg[i];

     if(row == 0 || row == key-1)
        k= k * (-1);

     row = row + k;
  }

  printf("\nOutput :");
  char nlet[100];
  for(i = 0; i < key; ++i)
     for(j = 0; j < msgLen; ++j)
        if(railMatrix[i][j] != '\n')
        {
           printf("%c", railMatrix[i][j]);
           strncat(nlet, &railMatrix[i][j], 1);
        }
```

```
 }

int main()
{
  char msg[100];
  int key;
  printf("Encryption:");
  printf("\nInput:");
  scanf("%[^\n]s",msg);
  printf("Key = ");
  scanf("%d",&key);
  encryptMsg(msg, key);
  return 0;
}
```

Sample Output

```
Encryption
Input : attack at once
Key = 2
Output : atc toctaka ne
```

Diffie hellman exchange

```
Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
        Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice:    x =(9^4 mod 23) = (6561 mod 23) = 6
        Bob:    y = (9^3 mod 23) = (729 mod 23)  = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
        Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys
        Alice:  ka = y^a mod p = 65536 mod 23 = 9
        Bob:    kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.
```

Source Code

```
#include<stdio.h>


long int power (int a, int b, int mod)
{
  long long int t;
  if(b==1)
     return a;
  t=power (a, b/2, mod);
  if(b%2==0)
     return (t*t) %mod;
  else
     return (((t*t) %mod) *a) %mod;
}


long long int calculateKey (int a, int x, int q)
{
  return power (a, x, q);
}


int main ()
```

```
{
  int q, alpha, x, a, y, b;
// both the persons will be agreed upon the common prime number and prime root
  printf("Enter the prime number and prime root : ");
  scanf("%d %d", &q, &alpha);
// A will choose the x
  printf("Enter the private key of A : ");
  scanf("%d", &x);
  a=power(alpha, x, q);
// B will choose the y
  printf("Enter the private key of B : ");
  scanf("%d", &y);
  b=power(alpha,y,q);
  printf("A computes key K : %lld \n", power(b, x, q));
  printf("B computes key K : %lld\n", power(a, y, q));
  return 0;
}
```

Sample Output

```
Enter the prime number and prime root : 23 9
Enter the private key of A : 4
Enter the private key of B : 3
A computes key K : 9
B computes key K : 9
—----------------------------------------------------
```

Hill Cipher

Source Code

```
#include <stdio.h>
#include <string.h>

// Following function generates the
//  key matrix for the key string
void getKeyMatrix(char key[6], int keyMatrix[][3])
{
   int k = 0;
   for (int i = 0; i < 3; i++)
   {
      for (int j = 0; j < 3; j++)
      {
         keyMatrix[i][j] = (key[k]) % 65;
         k++;
      }
   }
}

// Following function encrypts the message
void encrypt(int cipherMatrix[][1],
          int keyMatrix[][3],
          int messageVector[][1])
{
   int x, i, j;
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 1; j++)
      {
         cipherMatrix[i][j] = 0;

         for (x = 0; x < 3; x++)
         {
            cipherMatrix[i][j] +=
```

```
                keyMatrix[i][x] * messageVector[x][j];
            }

            cipherMatrix[i][j] = cipherMatrix[i][j] % 26;
        }
    }
}

// Function to implement Hill Cipher
void HillCipher(char message[3], char key[9])
{
    // Get key matrix from the key string
    int keyMatrix[3][3];
    getKeyMatrix(key, keyMatrix);

    int messageVector[3][1];

    // Generate vector for the message
    for (int i = 0; i < 3; i++)
        messageVector[i][0] = (message[i]) % 65;

    int cipherMatrix[3][1];

    // Following function generates
    // the encrypted vector
    encrypt(cipherMatrix, keyMatrix, messageVector);

    char CipherText[3];

    // Generate the encrypted text from
    // the encrypted vector
    for (int i = 0; i < 3; i++)
        CipherText[i] = cipherMatrix[i][0] + 65;

    // Finally print the ciphertext
```

```
    printf("%s", CipherText);
}

// Driver function for above code
int main()
{
    // Get the message to be encrypted
    char message[3],key[9];
    printf("Enter the plaintext :");
    scanf("%s",message);



    // Get the key
    printf("Enter the key :");
    scanf("%s",key);

    HillCipher(message, key);

    return 0;
}
```

Sample Output

```
Enter the plaintext :ACT
Enter the key :GYBNQKURP
POH
```

key generation in simplified DES

Algorithm

Step 1: We accepted a 10-bit key and permuted the bits by putting them in the P10 table.

```
Key = 1 0 1 0 0 0 0 0 1 0
(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (1, 0, 1, 0, 0, 0, 0, 0, 1, 0)

P10 Permutation is: P10(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5,
k2, k7, k4, k10, k1, k9, k8, k6)
After P10, we get 1 0 0 0 0 0 1 1 0 0
```

Step 2: We divide the key into 2 halves of 5-bit each.

```
l=1 0 0 0 0, r=0 1 1 0 0
```

Step 3: Now we apply one bit left-shift on each key.

```
l = 0 0 0 0 1, r = 1 1 0 0 0
```

Step 4: Combine both keys after step 3 and permute the bits by putting them in the P8 table. The output of the given table is the first key K1.

```
After LS-1 combined, we get 0 0 0 0 1 1 1 0 0 0
P8 permutation is: P8(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k6, k3,
k7, k4, k8, k5, k10, k9)
After P8, we get Key-1 : 1 0 1 0 0 1 0 0
```

Step 5: The output obtained from step 3 i.e. 2 halves after one bit left shift should again undergo the process of two-bit left shift.

```
Step 3 output - l = 0 0 0 0 1, r = 1 1 0 0 0
After two bit shift - l = 0 0 1 0 0, r = 0 0 0 1 1
```

Step 6: Combine the 2 halves obtained from step 5 and permute them by putting them in the P8 table. The output of the given table is the second key K2.

```
After LS-2 combined = 0 0 1 0 0 0 0 0 1 1
P8 permutation is: P8(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k6, k3,
k7, k4, k8, k5, k10, k9)
After P8, we get Key-2 : 0 1 0 0 0 0 1 1
```

Source Code

```
#include<stdio.h>

int main()

{

  int i, cnt=0, p8[8]= {6,7,8,9,1,2,3,4};

  int p10[10]= {6,7,8,9,10,1,2,3,4,5};
```

```
  char input[11], k1[10], k2[10], temp[11];
  char LS1[5], LS2[5];
//k1, k2 are for storing interim keys
//p8 and p10 are for storing permutation key

//Read 10 bits from user...
  printf("Enter 10 bits input:");
  scanf("%s",input);
  input[10]='\0';

//Applying p10...
  for(i=0; i<10; i++)
  {
     cnt = p10[i];
     temp[i] = input[cnt-1];
  }
  temp[i]='\0';
  printf("\nYour p10 key is    :");
  for(i=0; i<10; i++)
  {
     printf("%d,",p10[i]);
  }

  printf("\nBits after p10     :");
  puts(temp);
//Performing LS-1 on first half of temp
  for(i=0; i<5; i++)
  {
     if(i==4)
        temp[i]=temp[0];
     else
        temp[i]=temp[i+1];
  }
//Performing LS-1 on second half of temp
  for(i=5; i<10; i++)
```

```
  {
     if(i==9)
        temp[i]=temp[5];
     else
        temp[i]=temp[i+1];
  }
  printf("Output after LS-1  :");
  puts(temp);

  printf("\nYour p8 key is     :");
  for(i=0; i<8; i++)
  {
     printf("%d,",p8[i]);
  }

//Applying p8...
  for(i=0; i<8; i++)
  {
     cnt = p8[i];
     k1[i] = temp[cnt-1];
  }
  printf("\nYour key k1 is     :");
  puts(k1);
//This program can be extended to generate k2 as per DES algorithm.
}
```

Sample Output

```
Enter 10 bits input:1100011100
Your p10 key is    :6,7,8,9,10,1,2,3,4,5,
Bits after p10     :1110011000
Output after LS-1  :1100110001

Your p8 key is     :6,7,8,9,1,2,3,4,
Your key k1 is     :10001100
```

Implementation of vigenere cipher

Algorithm

Encryption

The plaintext(P) and key(K) are added modulo 26.

$E_i = (P_i + K_i) \bmod 26$

Decryption

$D_i = (E_i - K_i + 26) \bmod 26$

Source Code

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
main()
{
  int i,j,k,numstr[100],numkey[100],numcipher[100];
  char str[100],key[100];
  printf("Enter a string\n");
  gets(str);
//converting entered string to Capital letters
  for(i=0,j=0; i<strlen(str); i++)
  {
     if(str[i]!=' ')
     {
        str[j]=toupper(str[i]);
        j++;
     }
  }
  str[j]='\0';
  printf("Entered string is : %s \n",str);
//Storing string in terms of ascii
  for(i=0; i<strlen(str); i++)
  {
     numstr[i]=str[i]-'A';
  }
```

```
printf("Enter a key\n");
gets(key);
//converting entered key to Capital letters
for(i=0,j=0; i<strlen(key); i++)
{
   if(key[i]!=' ')
   {
      key[j]=toupper(key[i]);
      j++;
   }
}
key[j]='\0';
//Assigning key to the string
for(i=0; i<strlen(str);)
{
   for(j=0; (j<strlen(key))&&(i<strlen(str)); j++)
   {
      numkey[i]=key[j]-'A';
      i++;
   }

}

for(i=0; i<strlen(str); i++)
{
   numcipher[i]=numstr[i]+numkey[i];
}
for(i=0; i<strlen(str); i++)
{
   if(numcipher[i]>25)
   {
      numcipher[i]=numcipher[i]-26;
   }
}
printf("Vigenere Cipher text is\n");
```

```
  for(i=0; i<strlen(str); i++)
  {
     printf("%c",(numcipher[i]+'A'));
  }

  printf("\n");
}
```

Sample Output:

```
Enter a string
getupearly
Entered string is : GETUPEARLY
Enter a key
monday
Vigenere Cipher text is
SSGXPCMFYB
```

Play Fair Cipher

Source Code

```
#include<stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>

#define MX 5
int choice;
void playfair(char ch1, char ch2, char key[MX][MX]) {
  int i, j, w, x, y, z;
  for (i = 0; i < MX; i++) {
     for (j = 0; j < MX; j++) {
        if (ch1 == key[i][j]) {
           w = i;
           x = j;
        } else if (ch2 == key[i][j]) {
           y = i;
           z = j;
        }
     }
  }
  //printf("%d%d %d%d",w,x,y,z);
  if (w == y) {
     if(choice==1){
        x = (x + 1) % 5;
        z = (z + 1) % 5;
     }
     else{
        x = ((x - 1) % 5+5)%5;
        z = ((z - 1) % 5+5)%5;
     }
     printf("%c%c", key[w][x], key[y][z]);
  } else if (x == z) {
```

```
    if(choice==1){
      w = (w + 1) % 5;
      y = (y + 1) % 5;
    }
    else{
      w = ((w - 1) % 5+5)%5;
      y = ((y - 1) % 5+5)%5;
    }
    printf("%c%c", key[w][x], key[y][z]);
  }
  else {
    printf("%c%c", key[w][z], key[y][x]);
  }
}
void removeDuplicates(char str[]){
  int hash[256]      =  {0};
  int currentIndex    = 0;
  int lastUniqueIndex  = 0;
  while(*(str+currentIndex)){
    char temp = *(str+currentIndex);
    if(0 == hash[temp]){
      hash[temp] = 1;
      *(str+lastUniqueIndex) = temp;
      lastUniqueIndex++;
    }
    currentIndex++;
  }
  *(str+lastUniqueIndex) = '\0';

}
int main() {
  int i, j, k = 0, l, m = 0, n;
  char key[MX][MX], keyminus[25], keystr[10], str[25] = {
    0
  };
```

```
char alpa[26] = {
   'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'
};
printf("\n1.Encryption\t2.Decryption\n\nChoice(1 or 2):");
scanf("%d",&choice);
if(choice!=1 && choice!=2){ printf("Invalid Choice"); return -1;}
fflush(stdin);
printf("\nEnter key:");
scanf("%s",keystr);
printf("Enter the text:");
scanf("%s",str);
removeDuplicates(keystr);
n = strlen(keystr);
//convert the characters to uppertext
for (i = 0; i < n; i++) {
   if (keystr[i] == 'j') keystr[i] = 'i';
   else if (keystr[i] == 'J') keystr[i] = 'I';
   keystr[i] = toupper(keystr[i]);
}
//convert all the characters of plaintext to uppertext
for (i = 0; i < strlen(str); i++) {
   if (str[i] == 'j') str[i] = 'i';
   else if (str[i] == 'J') str[i] = 'I';
   str[i] = toupper(str[i]);
}
// store all characters except key
j = 0;
for (i = 0; i < 26; i++) {
   for (k = 0; k < n; k++) {
      if (keystr[k] == alpa[i]) break;
      else if (alpa[i] == 'J') break;
   }
   if (k == n) {
      keyminus[j] = alpa[i];
```

```
            j++;
        }
    }
    //construct key keymatrix
    k = 0;
    for (i = 0; i < MX; i++) {
        for (j = 0; j < MX; j++) {
            if (k < n) {
                key[i][j] = keystr[k];
                k++;
            } else {
                key[i][j] = keyminus[m];
                m++;
            }
            printf("%c ", key[i][j]);
        }
        printf("\n");
    }
    // construct diagram and convert to cipher text
    printf("\nEntered text :%s\nOutput Text :", str);
    for (i = 0; i < strlen(str); i++) {
        if (str[i] == 'J') str[i] = 'I';
        if (str[i + 1] == '\0') playfair(str[i], 'X', key);
        else {
            if (str[i + 1] == 'J') str[i + 1] = 'I';
            if (str[i] == str[i + 1]) playfair(str[i], 'X', key);
            else {
                playfair(str[i], str[i + 1], key);
                i++;
            }
        }
    }
    if(choice==2) printf(" (Remove unnecessary X)");
}
```

Sample Output

```
1.Encryption      2.Decryption


Choice(1 or 2):1
Enter key:monarchy
Enter the text:feelingnice
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z

Entered text :FEELINGNICE
Output Text :GFLUGAQYEBIU
```