CAESAR CIPHER

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
void main()
{
      char plain[100],cipher[100];
      int key,i,length,result;
      printf("Enter the message\n");
      gets(plain);
      printf("Enter the key value\n");
      scanf("%d",&key);
      printf("Plain text %s is\n",plain);
      printf("Encrypted text:\n");
      for(i=0,length=strlen(plain);i<length;i++)
      {
            cipher[i]=plain[i]+key;
            if(isupper(plain[i])&&(cipher[i]>'Z'))
            cipher[i]=cipher[i]-26;
            if(islower(plain[i])&&(cipher[i]>'z'))
            cipher[i]=cipher[i]-26;
            printf("%c",cipher[i]);
      }
      printf("\nAfter Decryption\n");
      for(i=0;i<length;i++)
      {
            plain[i]=cipher[i]-key;
            if(isupper(cipher[i])&&(plain[i]<'A'))
            plain[i]=plain[i]+26;
            if(islower(cipher[i])&&(plain[i]<'a'))
            plain[i]=plain[i]+26;
            printf("%c",plain[i]);
      }
}
```

OUTPUT:

```
Enter the message
Cryptography
Enter the key value
5
Plain text Cryptography is
Encrypted text:
Hwduytlwfumd
After Decryption
Cryptography
```

MONO ALPHABETIC

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
void main()
{
      char pt[52] =
{'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',
'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'};
      char ct[52] =
{'Z','Y','X','W','V','U','T','S','R','Q','P','O','N','M','L','K','J','I',
'H','G','F','E','D','C','B','A',
'z','y','x','w','v','u','t','s','r','q','p','o','n','m','l','k','j','i','
h','g','f','e','d','c','b','a'};
      char p[20] = {'\0'},c[20] = {'\0'},r[20] = {'\0'};
      int i,j;
      printf("Enter the plain text: ");
      gets(p);
      for(i=0;i<strlen(p);i++)
      {
            for(j=0;j<52;j++)
            {
                  if(pt[j] == p[i])
                        c[i] = ct[j];
            }
      }
      printf("\nCipher text: %s",c);
      for(i=0;i<strlen(c);i++)
      {
            for(j=0;j<52;j++)
            {
                  if(ct[j] == c[i])
                        r[i] = pt[j];
            }
      }
      printf("\nPlain text: %s\n",r);
}
```

OUTPUT:

```
Enter the plain text: Cryptography

Cipher text: Xibkgltizksb
Plain text: Cryptography
```

PLAYFAIR CIPHER

```
#include<stdio.h>
#include<stdio.h>
#include<string.h>
#define SIZE 100
void generatekeyTable(char key[],int ks,char keyT[5][5])
{
      int dicty[26]={0},i=0,j=0;
      dicty['j'-'a']=1;
      for(int k=0;k<ks;k++)
      {
            if(dicty[key[k]-'a']==0&&key[k]!='j')
            {
                  dicty[key[k]-'a']=1;
                  keyT[i][j++]=key[k];
                  if(j==5)i++,j=0;
            }
      }
      for(int k=0;k<26;k++)
      {
            if(dicty[k]==0)
            {
                  keyT[i][j++]=(char)(k+'a');
                  if(j==5)i++,j=0;
            }
      }
      for(i=0;i<5;i++)
      {
            for(j=0;j<5;j++)
            {
                  printf("%c",keyT[i][j]);
            }
      printf("\n");
      }
}
int prepare(char str[],int len)
{
      for(int i=0;i<len;i+=2)
      {
            if(str[i]==str[i+1])
            {
                  for(int j=len;j>i+1;j--)
                  str[j]=str[j-1];
                  str[i+1]='x';
                  len++;
            }
      }
      if(len%2!=0)str[len++]='x';
      str[len]='\0';
      return len;
}
```

```
void search(char keyT[5][5],char a,char b,int arr[])
{
	if(a=='j')a='i';
	if(b=='j')b='i';
	for(int i=0;i<5;i++)
	for(int j=0;j<5;j++)
	{
		if(keyT[i][j]==a)arr[0]=i,arr[1]=j;
		if(keyT[i][j]==b)arr[2]=i,arr[3]=j;
	}
}
void encrypt(char str[],char keyT[5][5],int len)
{
	int pos[4];
	for(int i=0;i<len;i+=2)
	{
		search(keyT,str[i],str[i+1],pos);
		if(pos[0]==pos[2])
		{
			str[i]=keyT[pos[0]][(pos[1]+1)%5];
			str[i+1]=keyT[pos[0]][(pos[3]+1)%5];
		}
		else if(pos[1]==pos[3])
		{
			str[i]=keyT[(pos[0]+1)%5][pos[1]];
			str[i+1]=keyT[(pos[2]+1)%5][pos[1]];
		}
		else
		{
			str[i]=keyT[pos[0]][pos[3]];
			str[i+1]=keyT[pos[2]][pos[1]];
		}
	}
}
void playfaircrypt(char str[],char key[])
{
	char keyT[5][5];
	int ks=strlen(key);
	int ps=strlen(str);
	ps=prepare(str,ps);
	generatekeyTable(key,ks,keyT);
	encrypt(str,keyT,ps);
	printf(" ciphertext:%s\n",str);
}
int main()
{
	char str[SIZE],key[SIZE];
	printf("enter the key:\n");
	scanf("%s",key);
	printf("enter the plaintext:\n");
	scanf("%s",str);
	playfaircrypt(str,key);
}
```

OUTPUT:

```
enter the key:
occurrence
enter the plaintext:
cryptographyandnetworksecurity
ocure
nabdf
ghikl
mpqst
vwxyz
ciphertext:uewsmekohwkwbafafzvcdstruruksz
```

HILL CIPHER

```
#include <stdio.h>
#include <string.h>
void encryption(int msg[100][2], int key[2][2], int cipher_mat[100][2],
int len1);
void decryption(int cipher_mat[100][2], int key[2][2], int len1);
int main()
{
    char message[100];
    int key[2][2], msg[100][2] = {0}, cipher_mat[100][2] = {0};
    int len, len1, count = 0, i, j;
    printf("Enter the message:\n");
    gets(message);
    printf("Enter the 2x2 key matrix:\n");
    for (i = 0; i < 2; i++)
    for (j = 0; j < 2; j++)
    scanf("%d", &key[i][j]);
    len = strlen(message);
    len1 = (len % 2 == 0) ? len / 2 : (len / 2) + 1;
    for (i = 0; i < len1; i++)
    {
        for (j = 0; j < 2; j++)
        {
            if (count < len)
                msg[i][j] = message[count++] - 'a';
            else
                msg[i][j] = 'x' - 'a';
        }
    }
    printf("Encryption is:\n");
    encryption(msg, key, cipher_mat, len1);
    printf("Decryption is:\n");
    decryption(cipher_mat, key, len1);
    return 0;
}
void encryption(int msg[100][2], int key[2][2], int cipher_mat[100][2],
int len1)
{
    int i, j, k;
    for (i = 0; i < len1; i++)
    {
        for (j = 0; j < 2; j++)
        {
            cipher_mat[i][j] = 0;
            for (k = 0; k < 2; k++) \
            {
                cipher_mat[i][j] += msg[i][k] * key[k][j];
            }
            cipher_mat[i][j] %= 26;
            printf("%c", cipher_mat[i][j] + 'a');
        }
    }
```

```
    printf("\n");
}
void decryption(int cipher_mat[100][2], int key[2][2], int len1)
{
    int det, inv_det, adj[2][2], inv_key[2][2];
    int decrypt_mat[100][2] = {0};
    int i, j, k;
    det = (key[0][0] * key[1][1] - key[0][1] * key[1][0]) % 26;
    if (det < 0)
        det += 26;
    for (inv_det = 1; (det * inv_det) % 26 != 1; inv_det++);
    adj[0][0] = key[1][1];
    adj[1][1] = key[0][0];
    adj[0][1] = -key[0][1];
    adj[1][0] = -key[1][0];
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            inv_key[i][j] = (adj[i][j] * inv_det) % 26;
            if (inv_key[i][j] < 0)
                inv_key[i][j] += 26;
        }
    }
    for (i = 0; i < len1; i++)
    {
        for (j = 0; j < 2; j++)
        {
            decrypt_mat[i][j] = 0;
            for (k = 0; k < 2; k++)
            {
                decrypt_mat[i][j] += cipher_mat[i][k] * inv_key[k][j];
            }
            decrypt_mat[i][j] %= 26;
            printf("%c", decrypt_mat[i][j] + 'a');
        }
    }
    printf("\n");
}
```

OUTPUT:

```
Enter the message:
cryptography
Enter the 2x2 key matrix:
2 3
3 4
Encryption is:
dwpccjlitiin
Decryption is:
cryptography
```

RAIL FENCE

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
      int i, j, len, rails, count, code [100] [1000];
      char str[1000];
      printf("Enter the secret msg\n");
      gets(str);
      len=strlen(str);
      printf("Enter the number of rails\n");
      scanf("%d",&rails);
      for(i=0;i<rails; i++)
      {
      for(j=0;j<len; j++)
      {
      code[i][j]=0;
      }
      }
      count=0;
      j=0;
      while(j<len)
      {
            if(count%2==0)
            {
                  for(i=0;i<rails; i++)
                  {
                        code[i][j]=(int)str[j];
                        j++;
                  }
            }
            else
            {
                  for(i=rails-2;i>0; i--)
                  {
                        code[i][j]=(int)str[j];
                        j++;
                  }
            }
            count++;
      }
      printf("Cipher :");
      for(i=0; i<rails; i++)
      {
            for(j=0;j<len; j++)
            {
                  if(code[i][j]!=0)
                  printf("%c", code[i][j]);
            }
      }
      printf("\n");
}
```

OUTPUT:

```
Enter the secret msg
cryptographyandnetworksecurity
Enter the number of rails
2
Cipher :cytgahadewrscrtrporpynntokeuiy
```

MILLER ROBIN

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
long long int modulo(int base, int pw, int mod)
{
      long long int i,a=1;
      for(i=0;i<pw;i++)
      a=a*base;
      return a%mod;
}
int Miller(int p) \
{
      long long int c=0,m,b,temp,a=2,k=0;
      if (p < 2)
      {
            return 0;
      }
      if ((p != 2) && (p % 2 == 0))
      {
            return 0;
      }
      m = p - 1;
      while ((m % 2) == 0)
      {
            m /= 2;
            k++;
      }
      b=modulo(a,m,p);
      if(b==1)
      return 1;
      while(b!=1&&b!=p-1&&k>0)
      {
            b=modulo(b,2,p);
            if(b==0)
            return 0;
            k--;
      }
      if(b==p-1)
      return 1;
      else
      return 0;
}
int main()
{
      int num;
      printf("Enter integer to test primality: ");
      scanf("%d", & num);
      if (Miller(num))
      printf("\n%d is prime\n", num);
      else
      printf("\n%d is not prime\n", num);
      return 0;
}
```

OUTPUT:

```
---->Enter integer to test primality: 97

        97 is prime

---->Enter integer to test primality: 1661

        1661 is not prime
```

DES

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void sboxAccess(int[8][4][16],int[48],int*);
void decimalToBinary(int,int*);
int main()
{
      int sboxes[8][4][16],i,j,k;
      for(i=0;i<8;i++)
      for(j=0;j<4;j++)
      for(k=0;k<16;k++)
      sboxes[i][j][k]=rand()%16;
      for(k=0;k<8;k++)
      {
            for(i=0;i<4;i++)
            {
                  for(j=0;j<16;j++)
                  {
                        printf("%d",sboxes[k][i][j]);
                  }
                  printf("\n");
            }
            printf("\n");
      }
      int input[48];
      for(k=0;k<48;k++)
      {
            input[k]=rand()%2;
      }
      int output[32];
      sboxAccess(sboxes,input,output);
      printf("\nS-box output");
      for(i=0;i<32;i++)
      {
            if(i%4==0)
            printf("\n");
            printf("%d",output[i]);
      }
      printf("\n Permutted output");
      int permutationTable[32];
      for(k=0;k<32;k++)
      {
            permutationTable[k]=rand()%32;
      }
      int permutedoutput[32];
      for(i=0;i<32;i++)
      {
            permutedoutput[i]=output[permutationTable[i]-1];
            if(i%4==0)
            printf("\n");
            printf("%d",permutedoutput[i]);
```

```
	}
	return 0;
}
void sboxAccess(int sboxes[8][4][16],int input[48],int output[32])
{
	int i,j,k;
	int numberInput[6],row,column,binaryVersion[4];
	for(i=0;i<8;i++)
	{
		printf("%d:",i);
		j=i*6;
		for(k=0;k<6;k++)
		{
			numberInput[k]=input[j+k];
		}
		row=(numberInput[0]*pow(2,1))+(numberInput[5]*pow(2,0));

	column=(numberInput[1]*pow(2,3))+(numberInput[2]*pow(2,2))+(numberI
nput[3]*pow(2,1))+ (numberInput[4]*pow(2,0));
		printf("\n Number in sbox
%d,%d,%d=%d\n",i,row,column,sboxes[i][row][column]);
		decimalToBinary(sboxes[i][row][column],binaryVersion);
		for(k=0;k<4;k++)
		{
			output[(i*4)+k]=binaryVersion[k];
		}
	}
}
void decimalToBinary(int number,int *binary)
{
	int bin[4]={0,0,0,0};
	int i=3;
	if(number!=0)
	{
		while(number!=1)
		{
			bin[i--]=number%2;
			number/=2;
		}
		bin[i]=number;
	}
	for(i=0;i<4;i++)
	{
		binary[i]=bin[i];
	}
}
```

OUTPUT:

```
0:
 Number in sbox 0,1,2=11
1:
 Number in sbox 1,0,10=2
```

```
2:
 Number in sbox 2,3,12=15
3:
 Number in sbox 3,2,5=4
4:
 Number in sbox 4,2,12=8
5:
 Number in sbox 5,2,5=14
6:
 Number in sbox 6,3,1=8
7:
 Number in sbox 7,1,2=0

S-box output
1011
0010
1111
0100
1000
1110
1000
0000
 Permutted output
1101
190119
0011
1101
0100
10190
0111
1000
```

RSA

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
long int e, d, n;
long int val[50];
char decode(long int ch)
{
      long int temp = ch;
      for (int i = 1; i < d; i++)
      ch = (temp * ch) % n;
      return (char)ch;
}
int gcd(long int a, long int b)
{
      long int temp;
      while (b != 0)
      {
            temp = b;
            b = a % b;
            a = temp;
      }
      return a;
}
int encode(char ch)
{
      long int temp = ch;
      for (int i = 1; i < e; i++)
      temp = (temp * ch) % n;
      return (int)temp;
}
void main()
{
      long int p, q, phi;
      char text[50], ctext[50];
      printf("\nEnter a prime number for p: ");
      scanf("%ld", &p);
      printf("Enter a prime number for q: ");
      scanf("%ld", &q);
      printf("Enter the text to be encoded: ");
      scanf("%s", text);
      n = p * q;
      phi = (p - 1) * (q - 1);
      for(e=2;e<phi;e++)
      {
            if(gcd(e,phi)==1)
            break;
      }
      for(d=1;d<phi;d++)
      {
            if((d*e)%phi==1)
            break;
```

```
	}
	printf("\np=%ld\tq=%ld\tn=%ld\tphi=%ld\n", p, q, n, phi);
	for (int i = 0; text[i] != '\0'; i++)
	{
		val[i] = encode(text[i]);
	}
	val[strlen(text)] = -999;
	printf("Encoded Message:\n");
	for (int i = 0; val[i] != -999; i++)
	{
		printf("%ld ", val[i]);
	}
	printf("\n");
	for (int i = 0; val[i] != -999; i++)
	{
		ctext[i] = decode(val[i]);
	}
	ctext[strlen(text)] = '\0';
	printf("\nDecoded message is: %s\n\n", ctext);
}
```

OUTPUT:

```
Enter a prime number for p: 89
Enter a prime number for q: 97
Enter the text to be encoded: network

p=89    q=97    n=8633  phi=8448
Encoded Message:
5409 1412 950 8073 1511 456 187

Decoded message is: network
```

DIFFI HELLMEN

```
#include<stdio.h>
int power(int base, int pw, int mod)
{
      int i,b=1;
      for(i=0;i<pw;i++)
      b=b*base;
      return b%mod;
}
int main()
{
      int a, q, x1, y1, x2, y2;
      printf("Enter the value of q and a: ");
      scanf("%d%d", &q, &a);
      printf("Enter the value of x1 for the first person : ");
      scanf("%d", &x1);
      y1 = power(a,x1,q);
      printf("Enter the value of x2 for the second person : ");
      scanf("%d", &x2);
      y2 = power(a,x2,q);
      printf("key for the first person is : %d\n", power(y1,x2,q));
      printf("key for the second person is : %d\n", power(y2,x1,q));
      return 0;
}
```

OUTPUT:

```
Enter the value of q and a: 17 5
Enter the value of x1 for the first person : 4
Enter the value of x2 for the second person : 6
key for the first person is : 16
key for the second person is : 16
```

SHA-1

```
#include<stdio.h>
#include<string.h>
#include<malloc.h>
#include<math.h>
#include<stdlib.h>
#define rotateleft(x,n) ((x<<n)|(x>>(32-n)))
#define rotateright(x,n) ((x>>n)|(x<<(32-n)))
void SHA1(unsigned char * str1)
{
      unsigned int h0,h1,h2,h3,h4,a,b,c,d,e,f,k,temp;
      int i,j,m;
      h0=0x67452301;
      h1=0xEFCDAB89;
      h2=0x98BADCFE;
      h3=0x10325476;
      h4=0xC3D2E1F0;
      unsigned char * str;
      str = (unsigned char *)malloc(strlen((const char *)str1)+100);
      strcpy((char *)str,(const char *)str1);
      int current_length = strlen((const char *)str);
      int original_length = current_length;
      str[current_length] = 0x80;
      str[current_length + 1]='\0';
      char ic = str[current_length];
      current_length++;
      int ib = current_length % 64;
      if(ib<56)
      ib=56-ib;
      else
      ib=120-ib;
      for(i=0;i<ib;i++)
      {
            str[current_length]=0x00;
            current_length++;
      }
      str[current_length+1]='\0';
      for(i=0;i<6;i++)
      {
            str[current_length]=0x0;
            current_length++;
      }
      str[current_length]=(original_length*8)/0x100 ;
      current_length++;
      str[current_length]=(original_length * 8) % 0x100;
      current_length++;
      str[current_length+i]='\0';
      int number_of_chunks = current_length/64;
      unsigned long int word[80];
      for(i=0;i<number_of_chunks;i++)
      {
            for(j=0;j<16;j++)
            {
```

```
                word[j] = str[i*64 + j*4 + 0] * 0x1000000 + str[i*64 +
j*4 + 1] * 0x10000 + str[i*64 + j*4 + 2] * 0x100 +
                str[i*64 + j*4 + 3];
          }
          for(j=16;j<80;j++)
          {
                word[j] = rotateleft((word[j-3]^word[j-8]^word[j-
14]^word[j-16]),1);
          }
          a=h0;
          b=h1;
          c=h2;
          d=h3;
          e=h4;
          for(m=0;m<80;m++)
          {
                if(m<=19)
                {
                      f=(b & c)|((~b) & d);
                      k=0x5A827999;
                }
                else if(m<=39)
                {
                      f=b^c^d;
                      k=0x6ED9EBA1;
                }
                else if(m<=59)
                {
                      f=(b & c)|(b & d)|(c & d);
                      k=0x8F1BBCDC;
                }
                else
                {
                      f=b^c^d;
                      k=0xCA62C1D6;
                }
                temp=(rotateleft(a,5)+f+e+k+word[m])&0xFFFFFFFF;
                e=d;
                d=c;
                c=rotateleft(b,30);
                b=a;
                a=temp;
          }
          h0=h0+a;
          h1=h1+b;
          h2=h2+c;
          h3=h3+d;
          h4=h4+e;
     }
     printf("\n\n");
     printf("Hash: %x %x %x %x %x",h0, h1, h2, h3, h4);
     printf("\n\n");
}
```

```
void main()
{
      SHA1((unsigned char *)"The quick brown fox jumps over the lazy
dog");
}
```

OUTPUT:

```
Hash: 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12
```

DIGITAL SIGNATURE

```
def euclid(m, n):
      if n == 0:
            return m
      else:
            r = m % n
            return euclid(n, r)
def exteuclid(a, b):
      r1 = a
      r2 = b
      s1 = int(1)
      s2 = int(0)
      t1 = int(0)
      t2 = int(1)
      while r2 > 0:
            q = r1//r2
            r = r1-q * r2
            r1 = r2
            r2 = r
            s = s1-q * s2
            s1 = s2
            s2 = s
            t = t1-q * t2
            t1 = t2
            t2 = t
      if t1 < 0:
            t1 += a
      return (r1, t1)
p = 823
q = 953
n = p * q
pn = (p-1)*(q-1)
key = []
for i in range(2, pn):
      gcd = euclid(pn, i)
      if gcd == 1:
            key.append(i)
e = int(313)
r, d = exteuclid(pn, e)
if r == 1:
      d = d % pn
      print("decryption key is: ", d)
else:
      print("Multiplicative inverse for the given encryption key does not
exist. Choose a different encryption key ")
m = 56234
s = (m**d) % n
m1 = (s**e) % n
if m == m1:
      print("As m = m1, Accept the  message sent by Alice")
else:
      print("As m not equal to m1,  Do not accept the message sent by
Alice ")
```

OUTPUT:

```
Python3 dss.py
./a.out

(Changes should be done in code only)
----->
(p=823,q=953)
decryption key is:  160009
As m = m1, Accept the  message sent by Alice

----->
(p=899,q=999)
decryption key is:  231925
As m not equal to m1,  Do not accept the message sent by Alice
```