CNS LAB PROGRAMS

CAESAR CIPHER

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

void main()
{
  char plain[20], cipher[20];
  int key, i, length;
  int result;
  printf("\nEnter the plain text: ");
  scanf("%s",plain);
  printf("\nEnter the key value: ");
  scanf("%d",&key);
  printf("\nThe plain text is: %s",plain);

  printf("\nEncrypted text: ");
  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: ");
  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]);
  }
  printf("\n");
}
```

OUTPUT:

```
Enter the plain text: hello

Enter the key value: 3

The plain text is: hello
Encrypted text: khoor
After decryption: hello
```

```
Enter the plain text: 12345

Enter the key value: 3

The plain text is: 12345
Encrypted text: 45678
After decryption: 12345
```

PLAYFAIR CIPHER

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#define MX 5
void playfair(char ch1, char ch2, char key[MX][MX])
{
   int i, j, w, x, y, z;
   FILE *out;
   if((out = fopen("cipher.text","a+"))==NULL)
   {
      printf("File corrupted");
   }
   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;
       }
     }
   }
```

```
  if(w==y)
  {
    x = (x+1)%5;
    z = (z+1)%5;
    printf("%c%c",key[w][x],key[y][z]);
    fprintf(out,"%c%c", key[w][x],key[y][z]);
  }
  else if(x==z)
  {
    w = (w+1)%5;
    y = (y+1)%5;
    printf("%c%c",key[w][x],key[y][z]);
    fprintf(out,"%c%c", key[w][x],key[y][z]);
  }
  else
  {
    printf("%c%c",key[w][z],key[y][x]);
    fprintf(out,"%c%c", key[w][z],key[y][x]);
  }
  fclose(out);
}

void main()
{
  int i, j, k=0, l, m=0, n;
  char key[MX][MX], keyminus[25], keystr[10], str[25] = {0};
  char alpha[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("\nEnter the key: ");
```

```
scanf("%s", keystr);
printf("\nEnter the plain text: ");
scanf("%s", str);
n = strlen(keystr);
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]);
}
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]);
}
j=0;
for(i=0; i<26; i++)
{
   for(k=0; k<n; k++)
   {
      if(keystr[k]==alpha[i])
        break;
      else if(alpha[i]=='J')
        break;
```

```
    }
    if(k==n)
    {
      keyminus[j]=alpha[i];
      j++;
    }
  }
  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");
  }
  printf("\n\nEntered text: %s ",str);
  printf("\nCipher text: ");
  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++;
      }
    }
  }
  printf("\n\n");
}
```

OUTPUT:

```
Enter the key: monarchy

Enter the plain text: king
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: KING
Cipher text: EKYQ
```

HILL CIPHER

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

void encryption(int msg[100][100],int key[3][3],int cipher_mat[100][100],int
len1,int order);
int main()
{
  char message[100];
  int order,i,j,key[3][3],count,msg[100][100],l,len,len1,cipher_mat[100][100];

  printf("\nEnter the message: ");
  scanf("%s",message);

  printf("\nEnter the order of the key: ");
  scanf("%d",&order);

  printf("\nEnter the key: \n");
  for(i=0;i<order;i++)
    for(j=0;j<order;j++)
     scanf("%d",&key[i][j]);
  len=strlen(message);
  l=len%order;

  if(l==0)
    len1=len/order;
  else
```

```
  len1=(len/order)+1;
  count=0;
  for(i=0;i<len1;i++)
  {
    for(j=0;j<order;j++)
    {
      if(count<len)
        msg[i][j]=message[count++]-97;
      else
        msg[i][j]='x'-97;
    }
  }
  printf("\nEncryption is:\n");
  encryption(msg,key,cipher_mat,len1,order);
  return 0;
}

void encryption(int msg[100][100],int key[3][3],int cipher_mat[100][100],int
len1,int order)
{
  int i,j,k,u=0;
  char encrypt[100];
  for(i=0;i<len1;i++)
  {
    for(j=0;j<order;j++)
    {
      for(k=0;k<order;k++)
        cipher_mat[i][j]+=msg[i][k]*key[k][j];
    }
```

```
  }
  for(i=0;i<len1;i++)
  {
    for(j=0;j<order;j++)
    {
      cipher_mat[i][j]=cipher_mat[i][j]%26;
      printf("%c ",cipher_mat[i][j]+97);
      encrypt[u++]=cipher_mat[i][j]+97;
    }
  printf("\n");
  }
  encrypt[u]='\0';
  printf("\nEncrypted message: %s\n\n", encrypt);
}
```

OUTPUT:

```
Enter the message: paymoremoney

Enter the order of the key: 3

Enter the key:
17 17 5
21 18 21
2 2 19

Encryption is:
r r l
m w b
k a s
p d h

Encrypted message: rrlmwbkaspdh
```

RAIL FENCE CIPHER

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

int main()
{
  int i,j,len,rail,count,code[100][100];
  char str[1000];
  printf("\nEnter secret msg: ");
  scanf("%s",str);
  len=strlen(str);

  printf("\nEnter no of rails: ");
  scanf("%d",&rail);
  for(i=0;i<rail;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<rail;i++)
```

```
    {
      code[i][j]=(int)str[j];
      j++;
    }
  }
  else
  {
    for(i=rail-2;i>0;i--)
    {
      code[i][j]=(int)str[j];
      j++;
    }
  }
  count++;
}
printf("\nEncrypted message: ");
for(i=0;i<rail;i++)
{
  for(j=0;j<len;j++)
  {
    if(code[i][j]!=0)
    printf("%c",code[i][j]);
  }
}
printf("\nDecrypted message: ");
count = 0;
j = 0;
while(j<len)
{
```

```
    if(count%2 == 0)
    {
      for(i=0; i<rail; i++)
      {
        if(code[i][j]!=0)
          printf("%c",code[i][j]);
        j++;
      }
    }
    else
    {
      for(i=rail-2; i>0; i--)
      {
        if(code[i][j]!=0)
          printf("%c",code[i][j]);
          j++;
      }
    }
   count++;
  }
  printf("\n\n");
  return 0;
}
```

OUTPUT:

```
Enter secret msg: Cryptography

Enter no of rails: 3

Encrypted message: Ctarporpyygh
Decrypted message: Cryptography
```

MONO-ALPHABETIC CIPHER

```
#include <stdio.h>

#include <string.h>

#include<ctype.h>


void main()

{

  char pt[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'};

  char ct[26] =
{'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("\nEnter plain text: ");

  scanf("%s",p);

  for(i=0; i<strlen(p);i++)

  {

    for(j=0; j<26; j++)

    {

      if(toupper(p[i]) == pt[j])

      c[i] = ct[j];

    }

  }


  printf("\nCipher text: %s",c);

  for(i=0; i<strlen(c);i++)

  {
```

```
    for(j=0; j<26; j++)
    {
      if(toupper(c[i]) == ct[j])
      r[i] = pt[j];
    }
  }
  printf("\nPlain text is: %s",r);
  printf("\n");
}
```

OUTPUT:

```
Enter plain text: Tumkur

Cipher text: GFNPFI
Plain text is: TUMKUR
```

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;
   printf("\nS-box numbers: \n\n");
   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\nPermutted output");
  int permutationTable[32]=
{16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24,14,32,27,3,9,19,13,30,6,2
2,11,4,25};
  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]);
  }
 printf("\n\n");
 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))+(numberInpu
t[3]*pow(2,1))+(numberInput[4]*pow(2,0));
    printf(" 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:

```
S-box numbers:
76931151012913101121136
12248118713610143315910
62137183105135789144
1121361144114241113127

1091411112714114710121 16
1581210120111121512412 81112
2101431110101011916512124
581419912118915511113

1515709110510111108741 3
1531481211454310514119 3
1104411491210412211015 11
3133087512111529101113 5

1111012153898411351014 8
8280913134121514710111 25
1261111010521405311412 3
6431520414152591321 49

8042109481010125148 84
1211314127131192564315 13
3415131346714212121040 6
0351211274412119010 63

145112973790334394
71402086541144124810
9963010109101412141538
4310411190274141112 95

5158592143010220510 4
858462489875501010
1520941412591479311312
6501274511212611201211

2124610111315212349151 0
1515763127158131413131515
946351744378136212
5928510813791561415
```

```
0: Number in sbox 0,3,0 = 11
1: Number in sbox 1,0,4 = 1
2: Number in sbox 2,0,0 = 5
3: Number in sbox 3,0,9 = 4
4: Number in sbox 4,3,7 = 4
5: Number in sbox 5,0,2 = 1
6: Number in sbox 6,0,11 = 2
7: Number in sbox 7,2,11 = 8

S-box output
1011
0001
0101
0100
0100
0001
0010
1000

Permutted output
0000
1100
1000
0101
0111
0110
0000
0010
```

RSA

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
long int e,d,n;
long int val[50];

char decode(long int ch)
{
  int i;
  long int temp=ch;
  for(i=1;i<d;i++)
    ch=(temp*ch)%n;
    return 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 prime(int a)
```

```
{
   int i;
   for(i=2;i<a;i++)
   {
     if((a%i)==0)
       return 0;
   }
   return 1;
}

int encode(char ch)
{
   int i;
   long int temp;
   temp=ch;
   for(i=1;i<e;i++)
   temp=(temp*ch)%n;
   return temp;
}

int main()
{
   int i;
   long int p;
   long int q,phi,c[50];
   char text[50],ctext[50];
   system("clear");
   printf("\nEnter the text to be encoded: ");
   scanf("%s",text);
```

```
do
{
  p=rand()%30;
}while(!prime(p));

do
{
  q=rand()%30;
}while(!prime(q));

n=p*q;
phi=(p-1)*(q-1);
printf("\np=%ld\tq=%ld\tn=%ld\tphi=%ld\n",p,q,n,phi);

do
{
  e=rand()%phi;
}while(!gcd(e,phi));

do
{
  d=rand()%phi;
}while(((d*e)%phi)!=1);

printf("\n************** Encoding Message **************\n");
sleep(3);

for(i=0;text[i]!='\0';i++)
val[i]=encode(text[i]);
```

```
    val[i]=-999;
    printf("Encoded+ Message:\n");

    for(i=0;val[i]!=-999;i++)
    printf("%ld",val[i]);
    printf("\n");
    printf("\n********* Decoding encrypted Message *********\n");
    sleep(3);

    for(i=0;val[i]!=-999;i++)
    ctext[i]=decode(val[i]);
    ctext[i]='\0';
    printf("\nDecoded message is:%s\n\n",ctext);
}
```

OUTPUT:

```
Enter the text to be encoded: Hello
p=13    q=23    n=299   phi=264
************** Encoding Message **************
Encoded Message:
5875147147227
********* Decoding encrypted Message *********
Decoded message is: Hello
```

DIFFIE-HELLMAN

```
#include<stdio.h>

long 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 n)
{
   return power(a,x,n);
}

int main()
{
   int n,g,x,a,y,b;
   printf("Enter the value of n and g: ");
   scanf("%d%d",&n,&g);

   printf("Enter the value of x for the first person ");
   scanf("%d",&x);
   a = power(g,x,n);
```

```
    printf("Enter the value of y for the second person: ");
    scanf("%d",&y);
    b = power(g,y,n);
    printf("Key for the first person is: %lld\n", power(b,x,n));
    printf("Key for the second person is: %lld\n", power(a,y,n));

    return 0;
}
```

OUTPUT:

```
Enter the value of n and g: 7 9

Enter the value of x for the first person: 6
Enter the value of y for the second person: 15

Key for the first person is: 1
Key for the second person is: 1
```

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: d3f91 fdcc13de c5345a9d c2bf958b ca125bad
```

MILLER RABIN PRIMALITY

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

long long mulmod(long long a, long long b, long long mod)
{
   long long x=0, y=a%mod;
   while(b>0)
   {
     if(b%2==1)
     {
        x=((x+y)%mod);
     }
     y=((y*2)%mod);
     b /= 2;
   }
   return (x%mod);
}

long long module(long long base, long long exponent, long long mod)
{
   long long x=1;
   long long y=base;
   while(exponent>0)
   {
     if (exponent%2 == 1)
     {
      x=(x*y)%mod;
```

```
    }
    y=(y*y)%mod;
    exponent = exponent/2;
  }
  return (x%mod);
}

int miller(long long p, int iteration)
{
  int i;
  long long s;
  if(p<2)
  {
    return 0;
  }
  if((p!=2)&&(p%2==0))
  {
    return 0;
  }
  s = p-1;
  while((s%2)==0)
  {
    s /= 2;
  }
  for(i=0;i<iteration;i++)
  {
    long long a = rand()%(p-1)+1,temp=s;
    long long mod = module(a,temp,p);
    while((temp!=p-1)&&(mod!=1)&&(mod!=p-1))
```

```
    {
      mod = mulmod(mod, mod, p);
      temp *= 2;
    }
    if((mod!=p-1)&&(temp%2 == 0))
    {
       return 0;
    }
  }
  return 1;
}

int main()
{
  int iteration = 5;
  long long num;
  printf("\nEnter integer to test primality: ");
  scanf("%lld",&num);
  if(miller(num, iteration))
     printf("\n%lld is prime \n",num);
  else
     printf("\n%lld is not prime \n",num);

  printf("\n");
  return 0;
}
```

OUTPUT:

```
Enter integer to test primality: 111

111 is not prime
```

```
Enter integer to test primality: 13

13 is prime
```