SIR GURUDAS MAHAVIDYALAYA

DEPARTMENT OF

Computer Sc.

LAB MANUAL

Subject: Data Communication (Algo. Using C)

Paper Code: CC8

INDEX

1. a) Implement character stuffing on given data

Algorithm:

Step 1: Initially give the user 2 choices, whether to character stuff or to directly exit, if wrong choice is entered then prompt an invalid choice message.

Step 2: Intake from the user the number of characters which are to be character stuffed.

Step 3: Then the characters which are to be stuffed are to be taken inside the for loop.

Step 4: Original data is displayed and the characters to be stuffed at the start and end of the frame are uploaded in the program.

Step 5: If DLE character is present then stuff DLE character before it.

Step 6: The characters DLESTX are inserted at the start and end of the data.

Step 7: The data along with the stuffed characters are displayed

Step 8: The original data is recovered and displayed on the receiving side

Step 9: Stop

Program :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void charc(void);
void main()
{
int choice;
while(1)
{
printf("\n\n\n1.character stuffing");
printf("\n\n2.exit");printf("\n\n\nenterchoice");
scanf("%d",&choice);
printf("%d",choice);if(choice>2)
printf("\n\n invalid option    please renter");
switch(choice)
{
case 1:charc();
```

```
break;
case 2:_exit(0);
```

```
}
}
}
void charc(void)
{
clrscr();
char c[50],d[50],t[50];
int i,m,j;
printf("enter the number of characters\n");
scanf("%d",&m);printf("\n enter the characters\n");
for(i=0;i<m+1;i++)
{
scanf("%c",&c[i]);
}
printf("\n original data\n");
for(i=0;i<m+1;i++)
printf("%c",c[i]);d[0]='d';
d[1]='l';d[2]='e';
d[3]='s';d[4]='t';
d[5]='x';
for(i=0,j=6;i<m+1;i++,j++)
{
if((c[i]=='d'&&c[i+1]=='l'&& c[i+2]=='e'))
{
d[j]='d';
j++;
d[j]='l';
j++;
d[j]='e';
j++;
m=m+3;
}
d[j]=c[i];
}
m=m+6;
m++;
d[m]='d';
m++;
d[m]='l';
m++;
d[m]='e';
m++;
d[m]='s';
m++;
```

```
d[m]='t';
m++;
```

```
d[m]='x';
m++;
printf("\n\n transmitted data: \n");
for(i=0;i<m;i++)
{
printf("%c",d[i]);
}
for(i=6,j=0;i<m-6;i++,j++)
{
if(d[i]=='d'&&d[i+1]=='l'&&d[i+2]=='e'&&d[i+3]=='d'&&d[i+4]=='l'&&d[i+5]=='e')
i=i+3;
t[j]=d[i];
}
printf("\n\nreceived data:");
for(i=0;i<j;i++)
{
printf("%c",t[i]);
}
}
```

o/p :

1 b) Implement character count on given data

Algorithm for Character count

1. Start
2. Append DLE STX at the beginning of the string
3. Check the data if character is present; if character DLE is present in the string (example DOODLE) insert another DLE in the string (ex: DOODLEDLE)
4. Transmit DLE ETXat the end of the string
5. Display the string
6. Stop

PROGRAM:

```
#include<stdio.h>
#include<string.h>
main()
{
int i,j,k,l,count=0,n;
char s[100],cs[50];
clrscr();
printf("\n ENTER THE BIT STRING:");
gets(s);
n=strlen(s);
printf("\nTHE STRING IS\n");
for(i=0;i<n;)
     {
if(s[i]==s[i+1])
{
 count=2;
 i++;
 while(s[i]==s[i+1])
 {
 i++;
 count++;
 }
 if(count>=5)
 {
 printf("$");
 if(count<10)
 printf("0");
 printf("%d%c",count,s[i]);
```

```
i++;
```

```
}
else
{
for(j=0;j<count;j++)
printf("%c",s[i]);
i++;
}
}
else
{
printf("%c",s[i]);
i++;
}
}
   getch(); }
```

INPUT/OUTPUT:

ENTER THE BIT STRING:

123AAAAAAAAAATYKKKPPPPP

THE STRING IS

123$10ATYKKK$05P

b) Decode the stuffed data

Algorithm for Character De−stuffing

1. Start
2. Neglect initial DLE STX
3. If DLE is present in the text, ngelect it; if another DLE follows, copy the same to output. 4. Neglect the trailing DLE ETX
5. Stop

PROGRAM:

```
#include<stdio.h>
#include<string.h>
main()
{
int i,j,k,l,n,count;
char s[100],cs[50];
clrscr();

printf("\n ENTER THE STUFFED STRING :");
gets(s);
n=strlen(s);
printf("\nTHE STRING IS\n");
        for(i=0;i<n;)
    {
        if(s[i]=='$')
        { i++;
         count=(s[i]-'0')*10+(s[i+1]-'0');
         if(count<5)
         { clrscr();
           printf("INVALIDE MESSAGE");
           exit(1);
         }
         while(count>0)
         { printf("%c",s[i+2]);
          count--;
         }
         i=i+3;
        }
          else
```

```
{
printf("%c",s[i]);
```

```
        i++;
        }
    }
  getch();    }
```

INPUT/OUTPUT:

ENTER THE STUFFED STRING: 123$10ATY$06K

THE STRING IS:

123AAAAAAAAAATYKKKKKK

c) Bit stuffing on given binary data

Algorithm for Bit−Stuffing

1. Start
2. Initialize the array for transmitted stream with the special bit pattern 0111 1110 which indicates the beginning of the frame.
3. Get the bit stream to be transmitted in to the array.
4. Check for five consecutive ones and if they occur, stuff a bit 0
5. Display the data transmitted as it appears on the data line after appending 0111 1110 at the end
6. For de−stuffing, copy the transmitted data to another array after detecting the stuffed bits 7. Display the received bit stream
8. Stop

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
int b[100],b1[100],l,k,n=0,i,j,z,i1,s[20],f[8]={0,1,1,1,1,1,1,0},j1;
static int a[100];
char ch='y',bs[50];
clrscr();
do
{
i1=z=n=0;
clrscr();
printf("\n Enter the bit string(space for each byte)");
gets(bs);
for(i=0;bs[i]!='\0';i++)
if(bs[i]!=' ')
b[n++]=bs[i]-'0';
for(i=0;i<n;i++)
{
if(b[i]==1){
i1++;
if(i1==5){s[z++]=i+1;i1=0;}
}
else i1=0;
}
```

```
j1=j=0;
```

```
for(i=0;i<z;i++)
{
while(j<s[i])
b1[j1++]=b[j++];
b1[j1++]=0;
}
while(j1<n+z)
b1[j1++]=b[j++];
l=n/8;
for(i=0;l>0;i++
)
{
a[i]=l%2;
l=l/2;
}
printf("\nAfter stuffing :");
for(j=7;j>=0;j--)
printf("%d",a[j]);
printf("   ");
for(k=0;k<8;k++)
printf("%d",f[k]);
printf("     ");
for(k=0;k<j1;k++)
printf("%d",b1[k])
; printf(" ");
for(k=0;k<8;k++)
printf("%d",f[k]);
printf("\n\n Do u want to continue?");
ch=getch();
}
while(ch=='y' || ch=='Y');
getch();
}
```

INPUT/OUTPUT:

Enter the bit string (space for each byte)
11111111 01111110 00111110

After stuffing : 00000011 01111110 11111011101111010001111100 01111110

d) Destuff the given stuffed data frame

```
include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
int i,n,n1,k,j,ni,len;
char f[8]={'0','1','1','1','1','1','1','0'},st[100];
static int ds[100];
clrscr();
printf("\n\nEnter the stuffed data");
gets(st);
n=strlen(st);
ni=k=0;
        for(i=8;i<16;i++)
        if(st[i]!=f[k++])
        {
        printf("\nError in flag");
        exit(1);
        }
        k=0;
        for(i=n-8;i<n;i++
        ) if(f[k++]!=st[i])
        {
        printf("\nError in flag");
        exit(1);
        }
        for(i=0;i<n;i++
        ) st[i]=st[i]-'0';
        len=0;
        j=7;
        for(i=0;i<8;i++)
        len+=pow(2,i)*st[j--]
        ; k=ni=j=0;
        for(i=16;i<n-8;i++)
        {
        if(st[i]==1){
        ni++;
        if(ni==5)
        {
        ds[j++]=1;
        i++;
```

```
k++;
ni=0;}
```

```
        else
        ds[j++]=1;
        }
        else
        ds[j++]=0;
        }
        n1=n-24-k;
        if(len*8!=n1){
        printf("\n Error in data length");
        exit(1);
        }
        printf("\n After destuffing ");
        for(i=0;i<n1;i++)
        printf("%d",ds[i]);
  getch();
  }
```

INPUT/OUTPUT:

Enter the stuffed data000000110111111011111011101111101000111110001111110
After destuffing 111111110111101000111100

Exercise:

a) **Implement K-Bit run length code on given data**
b) **Decode the K-Bit run length code**

1. a)Generate CRC code for a given data

frame Algorithm

1. A string of n as is appended to the data unit. The length of predetermined divisor is n+ 1.

2. The newly formed data unit 1. A string of n as is appended to the data unit. The length of predetermined divisor is n+ 1.

i.e. original data + string of n as are divided by the divisor using binary division and remainder is obtained. This remainder is called CRC.

3. Now, string of n Os appended to data unit is replaced by the CRC remainder (which is also of n bit).

4. The data unit + CRC is then transmitted to receiver.

5. The receiver on receiving it divides data unit + CRC by the same divisor & checks the remainder.

6. If the remainder of division is zero, receiver assumes that there is no error in data and it accepts it.

7. If remainder is non-zero then there is an error in data and receiver rejects it.

PROGRAM:

```
#include<stdio.h>
#include<math.h
> main()
{

int i,j,k,m,n,cl;
char a[10],b[100],c[100];
clrscr();
printf("\n ENTER POLYNANOMIAL:");
scanf("%s",a);
printf("\n ENTER THE FRAME:");
scanf("%s",b);
m=strlen(a);
n=strlen(b);

for(i=0;i<m;i++) /* To eliminat first zeros in polynomial */
{
  if(a[i]=='1')
  {
    m=m-i;
    break;
```

```
}
```

```
}

  for(k=0;k<m;k++) /* To Adjust the polynomial */
  a[k]=a[k+i];

  cl=m+n-1;
  for(i=0;i<n;i++) /* To copy the original frame to c[]*/
  c[i]=b[i];
  for(i=n;i<cl;i++) /* To add n-1 zeros at the end of frame */
  c[i]='0';
  c[i]='\0';         /*To make it as a string */

  for(i=0;i<n;i++) /* To set polynomial remainder at end of c[]*/
  if(c[i]=='1')
  {
  for(j=i,k=0;k<m;k++,j++)
  if(a[k]==c[j])
  c[j]='0';
  else
  c[j]='1';
  }
  for(i=0;i<n;i++) /* To copy original data in c[] */
  c[i]=b[i];
  printf("\n THE MESSAGE IS: %s",c);
  getch();
  }
```

INPUT/OUTPUT:

ENTER POLYNANOMIAL:1011

ENTER THE FRAME:10011101

THE MESSAGE IS: 10011101011

ENTER POLYNANOMIAL:00101

ENTER THE FRAME:10101011

THE MESSAGE IS: 1010101101

b) Verify the CRC code

```
#include<stdio.h>
#include<math.h
> main()
{

int i,j,k,m,n,cl;
char a[10],c[100];
clrscr();
printf("\n ENTER POLYNANOMIAL:");
scanf("%s",a);
printf("\n ENTER THE CRC FRAME:");
scanf("%s",c);
m=strlen(a);
cl=strlen(c);

for(i=0;i<m;i++) /* To eliminat first zeros in polynomial */
{
  if(a[i]=='1')
  { m=m-i; break;  }

}

  for(k=0;k<m;k++) /* To Adjust the polynomial */
  a[k]=a[k+i];

     n=cl-m+1;

   for(i=0;i<n;i++) /* To check polynomial remainder is zero or not */
  if(c[i]=='1')
  {
  for(j=i,k=0;k<m;k++,j++)
  if(a[k]==c[j])
  c[j]='0';
  else
  c[j]='1';
  }
  for(i=0;i<cl;i++) /* To copy original data in c[] */
  if(c[i]=='1')
  {
  printf("\n THERE IS SOME ERROR IN MESSAGE :");
  break;
  }
```

```
if(i==cl)
printf("\n MESSAGE IS CORRECT");
```

```
getch();
}
```

INPUT/OUTPUT :

ENTER POLYNANOMIAL: 1011

ENTER THE CRC FRAME: 10101011101

THERE IS SOME ERROR IN MESSAGE:

ENTER POLYNANOMIAL: 01011

ENTER THE CRC FRAME: 10011101011

MESSAGE IS CORRECT

Exercise:

1. ***a)*** **Implement Hamming code Generation for a given binary code**

   b) Hamming code verification and checking
2. **Implement even and odd parity for given binary code**

2. Implement Dijkstra's algorithm to compute the shortest path through a graph

Algorithm:

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the unvisited set consisting of all the nodes.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6 + 2 = 8. If this distance is less than the previously recorded tentative distance of B, then overwrite that distance. Even though a neighbor has been examined, it is not marked as "visited" at this time, and it remains in the unvisited set.
4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Select the unvisited node that is marked with the smallest tentative distance, and set it as the new "current node" then go back to step 3.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<math.h>
main()
{
int u,v,num,i,j,l,k,s[10],min,cost[10][10],dist[10],path[10],n;
clrscr();
printf("\n ENTER VERTECES:");
scanf("%d",&n);
printf("\n ENTER ADJECENCY MATRIX:\n");
for(i=1;i<=n;i++)
```

```
{
for(j=1;j<=n;j++)
```

```
scanf("%d",&cost[i][j]);
}

for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
    if(i==j)
    cost[i][j]=0;
    else
    if(cost[i][j]==-1
    )
    cost[i][j]=30000;

    printf("\nENTER SOURCE VERTEX:");
    scanf("%d",&v);
    clrscr();

    for(i=1;i<=n;i++)
    {
      s[i]=0;
      path[i]=v;
      dist[i]=cost[v][i];
    }

      dist[v]=0;
      for(num=2;num<=n;num++)
      { min=30000;
           u=0;
           for(i=1;i<=n;i++)
           {
              if(s[i]!=1)
              if(min>dist[i])
             {
              u=i; min=dist[i];
             }
       }


             s[u]=1;
           for(i=1;i<=n;i++)
           { if(s[i]!=1)
              if(dist[i]>(min+cost[u][i]))
              {
                 dist[i]=min+cost[u][i];
                 path[i]=u;
              }
```

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

```
   printf("\nPATH MATRIX:\n");
   printf("\nDISTANCE NODE      PATH\n");
   for(i=1;i<=n;i++)
   { printf("\n %d",dist[i]);
      printf(" %d ",i);
      j=i;
      do
      {
      printf(" --> %d ",path[j]);
          u=path[j];
          j=u;
      }while(u!=v);

   }
  getch();
  }
```

INPUT/OUTPUT:

ENTER VERTECES:8

ENTER ADJECENCY MATRIX:

```
 0 2 -1 -1 -1 -1 6 -1
 2 0 7 -1  2 -1 -1 -1
-1 7 0 3 -1 3 -1 -1
-1 -1 3 0 -1 -1 -1 2
-1 2 -1 -1 0 2 1 -1
-1 -1 3 -1 2 0 -1 2
 6 -1 -1 -1 1 -1 0 4
-1 -1 -1 2 -1 2 4 0
```

ENTER SOURCE VERTEX:1

PATH MATRIX:

DISTANCE NODE PATH

```
 0 1   --> 1
 2 2   --> 1
 9 3   --> 2  --> 1
 10 4  --> 8  --> 6 --> 5 --> 2 --> 1
```

4 5 --> 2 --> 1

6 6 --> 5 --> 2 --> 1
5 7 --> 5 --> 2 --> 1
8 8 --> 6 --> 5 --> 2 --> 1

Exercise:

1. **Write a program for implementing link state routing using Dijkstra's algorithm**
2. **Write a program for Flooding algorithm**

4. Take an example subnet graph with weights indicating delay between nodes

```
#include<stdio.h>
#include<conio.h>
struct full
{
                                                    char line[10],dest[10];int hops;
}f[20];
  main()
{
 int nv,min,minver,i;
 char sv[2],temp;
 clrscr();
 printf("\nEnter number of vertices:");
 scanf("%d",&nv);
 printf("\n Enter source vertex: ");
 scanf("%s",sv);
 printf("\n Enter full table for source vertex %s :\n",sv);
```

```
for(i=0;i<nv;i++)
scanf("%s %s

%d",f[i].dest,f[i].line,&f[i].hops); printf("\n

HIERARCHIAL TABLE\n\n");

 for(i=0;i<nv;)
 {
 if(sv[0]==f[i].dest[0])
 {
 printf("\n %s %s %d",f[i].dest,f[i].line,f[i].hops);
 i++;
 }
 else
 {
 min=1000;
 minver=0;
 temp=f[i].dest[0];
 while(temp==f[i].dest[0])
 {
 if(min>f[i].hops)
 {
 min=f[i].hops;
 minver=i;
 }
 i++;
 }
 printf("\n %c %s %d ",temp,f[minver].line,f[minver].hops);
```

```
 }
 }
 getch();
}
```

INPUT/OUTPUT:

Enter number of vertices:

8 Enter source vertex :1A

Enter full table for source vertex 1A :
1A - -
1B 1B 1
1C 1C 1
2A 1B 1
2B 1B 2
3A 1C 2
3B 1C 3
4A 1C 3

HIERARCHIAL TABLE

1A - 0
1B 1B 1
1C 1C 1
2 1B 1
3 1C 2
4 1C 3

Exercise:

1. **Implement path vector routing protocol.**
2. **Routing Information Protocol**

5. Now obtain Routing table for each node using distance vector routing algorithm Algorithm:

Input: Graph and a given vertex *src*

Output: Shortest distance to all vertices from *src*. If there is a negative weight cycle, then shortest distances are not calculated, negative weight cycle is reported.

1) This step initializes distances from source to all vertices as infinite and distance to source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.

2) This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph.

…..**a)** Do following for each edge v-u

……If dist[v] > dist[u] + weight of edge uv, then update dist[v] …….dist[v] = dist[u] + weight of edge uv

3) This step reports if there is a negative weight cycle in graph. Do following for each edge u-v

……If dist[v] > dist[u] + weight of edge uv, then “Graph contains negative weight cycle” The idea of step 3 is, step 2 guarantees shortest distances if graph doesn’t contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle

Program:

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
main()
{
int i,j,k,nv,sn,noadj,edel[20],tdel[20][20],min;
char sv,adver[20],ch;
clrscr();

printf("\n ENTER THE NO.OF VERTECES:");
```

```
scanf("%d",&nv);
```

```
printf("\n ENTER THE SOURCE VERTEX NUM,BER AND NAME:");
scanf("%d",&sn);

flushall();
sv=getchar();

printf("\n NETER NO.OF ADJ VERTECES TO VERTEX %c",sv);
scanf("%d",&noadj);

for(i=0;i<noadj;i++)
{
printf("\n ENTER TIME DELAY and NODE NAME:");
scanf("%d %c",&edel[i],&adver[i]);
}
for(i=0;i<noadj;i++)
{
 printf("\n ENTER THE TIME DELAY FROM %c  to ALL OTHER
 NODES: ",adver[i]);
 for(j=0;j<nv;j++)
 scanf("%d",&tdel[i][j]);
}

printf("\n DELAY VIA--VERTEX \n ");
 for(i=0;i<nv;i++)
{
   min=1000;
   ch=0;
  for(j=0;j<noadj;j++)
  if(min>(tdel[j][i]+edel[j]))
  {
   min=tdel[j][i]+edel[j];
   ch=adver[j];
  }
   if(i!=sn-1)
    printf("\n%d
   %c",min,ch); else
    printf("\n0     -");
  }
 getch();
}
```

INPUT/OUTPUT:

ENTER THE NO.OF VERTECES:12

ENTER THE SOURCE VERTEX NUMBER AND NAME:10 J

ENTER NO.OF ADJ VERTECES TO VERTEX 4

ENTER TIME DELAY and NODE NAME:8 A

ENTER TIME DELAY and NODE NAME:10 I

ENTER TIME DELAY and NODE NAME:12 H

ENTER TIME DELAY and NODE NAME:6 K

ENTER THE TIME DELAY FROM A to ALL OTHER NODES:
0 12 25 40 14 23 18 17 21 9 24 29

ENTER THE TIME DELAY FROM I to ALL OTHER NODES:
24 36 18 27 7 20 31 20 0 11 22 33

ENTER THE TIME DELAY FROM H to ALL OTHER NODES:
20 31 19 8 30 19 6 0 14 7 22 9

ENTER THE TIME DELAY FROM K to ALL OTHER NODES:
21 28 36 24 22 40 31 19 22 10 0 9

DELAY VIA--VERTEX
8 a
20 a
28 i
20 h
17 i
30 i

18 h
12 h
10 i
0 -
6 k
15 k

Exercise:

1. **Routing table for each node using hierarchical routing algorithm**

2. ***Open Shortest Path First***

6. Take an example subnet of hosts. Obtain broadcast tree for it.

/*PROGRAM TO IMPLEMENT BROADCAST ROUTING ALGORITHM*/

Algorithm:

A router creates a data packet and then sends it to each host one by one. In this case, the router creates multiple copies of single data packet with different destination addresses. All packets are sent as unicast but because they are sent to all, it simulates as if router is broadcasting.

This method consumes lots of bandwidth and router must destination address of each node.

Secondly, when router receives a packet that is to be broadcasted, it simply floods those packets out of all interfaces. All routers are configured in the same way.

Program:

```
#include<stdio.h>
int a[10][10],n;
void main()
{
int i,j,root;
clrscr();
printf("Enter no.of nodes:");
scanf("%d",&n);
printf("Enter adjacent matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
printf("Enter connecting of %d-->%d::",i,j);
scanf("%d",&a[i][j]);
}
printf("Enter root node:");
scanf("%d",&root);
adj(root);
}
```

```
adj(int k)
{
int i,j;
```

```
printf("Adjacent node of root node::\n");
printf("%d\n\n",k);
for(j=1;j<=n;j++)
{
if(a[k][j]==1 ||
a[j][k]==1)
printf("%d\t",j);
}
printf("\n");
for(i=1;i<=n;i++
)
{
if((a[k][j]==0) && (a[i][k]==0) &&
(i!=k)) printf("%d",i);
}
}
```

OUTPUT

****

```
Enter  no.of  nodes:5
Enter          adjacent
matrix
Enter connecting of 1-->1::0
Enter connecting of 1-->2::1
Enter connecting of 1-->3::1
Enter connecting of 1-->4::0
Enter connecting of 1-->5::0
Enter connecting of 2-->1::1
Enter connecting of 2-->2::0
Enter connecting of 2-->3::1
Enter connecting of 2-->4::1
Enter connecting of 2-->5::0
Enter connecting of 3-->1::1
Enter connecting of 3-->2::1
Enter connecting of 3-->3::0
Enter connecting of 3-->4::0
Enter connecting of 3-->5::0
Enter connecting of 4-->1::0
Enter connecting of 4-->2::1
Enter connecting of 4-->3::0
Enter connecting of 4-->4::0
Enter connecting of 4-->5::1
Enter connecting of 5-->1::0
Enter connecting of 5-->2::0
Enter connecting of 5-->3::0
```

Enter connecting of 5-->4::1
Enter connecting of 5-->5::0
Enter root node:2
Adjacent node of root node:: 2

1 3 4
5

Exercise: 1. Implement Core-Based Tree(CBT) protocol obtain broadcast tree for it.

2. Implement an optimal algorithm for Broadcasting multiple messages in trees.

7. Take a 64 bit plain text and encrypt the same using DES algorithm.

/* Program to implement DES */

Algorithm:

1.) Firstly, we need to process the key.

1.1 Get a 64-bit key from the user. (Every 8th bit is considered a parity bit. For a key to have correct parity, each byte should contain an odd number of "1" bits.)

1.2 Calculate the key schedule.

1.2.1 Perform the following permutation on the 64-bit key. (The parity bits are discarded, reducing the key to 56 bits. Bit 1 of the permuted block is bit 57 of the original key, bit 2 is bit 49, and so on with bit56 being bit 4 of the original key.)

1.2.2 Split the permuted key into two halves. The first 28 bits are called C[0] and the last 28 bits are called D[0].

1.2.3 Calculate the 16 subkeys. Start with i = 1.

1.2.3.1 Perform one or two circular left shifts on both C[i-1] and D[i-1] to get C[i] and D[i], respectively.

1.2.3.3 Loop back to 1.2.3.1 until K[16] has been calculated.

2 Process a 64-bit data block.

2.1 Get a 64-bit data block. If the block is shorter than 64 bits, it should be padded as appropriate for the application.

2.2 Perform the following permutation on the data block. Initial Permutation (IP)

2.3 Split the block into two halves. The first 32 bits are called L[0], and the last 32 bits are called R[0].

2.4 Apply the 16 subkeys to the data block. Start with i = 1.

2.4.1 Expand the 32-bit R[i-1] into 48 bits according to the bit-selection

2.4.2 Exclusive-or E(R[i-1]) with K[i].

2.4.3 Break E(R[i-1]) xor K[i] into eight 6-bit blocks. Bits 1-6 are B[1], bits 7-12 are B[2], and so on with bits 43-48 being B[8].

2.4.4 Substitute the values found in the S-boxes for all B[j]. Start with j = 1. All values in the S- boxes should be considered 4 bits wide.

2.4.4.1 Take the 1st and 6th bits of B[j] together as a 2-bit value (call it m) indicating the row in S[j] to look in for the substitution.

2.4.4.2 Take the 2nd through 5th bits of B[j] together as a 4-bit value (call it n) indicating the column in S[j] to find the substitution.

2.4.4.3 Replace B[j] with

S[j][m][n]. Substitution Box 1

(S[1])

2.4.4.4 Loop back to 2.4.4.1 until all 8 blocks have been replaced.

2.4.5 Permute the concatenation of B[1] through B[8] as indicated

below. Permutation P

2.4.6 Exclusive-or the resulting value with L[i-1]. Thus, all together, your R[i] = L[i-1] xor P(S[1](B[1])...S[8](B[8])), where B[j] is a 6-bit block of E(R[i-1]) xor K[i]. (The function for R[i] is written as, R[i] = L[i-1] xor f(R[i-1], K[i]).)

2.4.7 L[i] = R[i-1].

2.4.8 Loop back to 2.4.1 until K[16] has been applied.

2.5 Perform the following permutation on the block

R[16]L[16]. Final Permutation (IP**-1)

Program:

```
#include<stdio.h>

#include<conio.h>

void main()

{

int k[15],k1[15],k2[15],i,j,p[15],p1[15],p2[15];
```

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

int p8[10]={6,3,7,4,8,5,10,9};
```

```
int t1,t2,t3,t4,t[4],b[10],s,h,a;
int ip[8]={2,6,3,1,4,8,5,7};
int ep[8]={4,1,2,3,2,3,4,1};
int
ip1[8]={4,1,3,5,7,2,8,6};
int p4[4]={2,3,4,1};
int s0[4][4]={{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
int s1[4][4]={{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
clrscr();
printf("\n\tSimplified-DES\n");
printf("\n");
printf("\nEnter the plain text of 8 bits length::\n");
for(i=0;i<8;i++)
scanf("%d",&p2[i]);
printf("\n\nEnter the key of 10 bits length::\n");
for(i=0;i<10;i++)
scanf("%d",&k[i]);
printf("\n\nKey Generation::\n");
for(i=0;i<10;i++)
{
j=p10[i];
p[i]=k[j-1];
```

```
}

t1=p[0];
```

```
t2=p[5];
for(i=0;i<4;i++
) p[i]=p[i+1];
p[i]=t1;
for(i=5;i<9;i++
) p[i]=p[i+1];
p[i]=t2;
for(i=0;i<8;i++
)
{
j=p8[i];
k1[i]=p[j-1];
}
t1=p[0];
t2=p[1];
t3=p[5];
t4=p[6];
for(i=0;i<3;i++)
{
p[i]=p[i+2];
}
p[i]=t1;
```

```
i++;

p[i]=t2;
```

```
for(i=5;i<8;i++
) p[i]=p[i+2];
p[i]=t3;
i++;
p[i]=t4;
for(i=0;i<8;i++
)
{
j=p8[i];
k2[i]=p[j-1];
}
printf("\nkey k1::");
for(i=0;i<8;i++)
printf("%d",k1[i]);
printf("\nkey k2::");
for(i=0;i<8;i++)
printf("%d",k2[i]);
for(i=0;i<8;i++)
p[i]=p2[i];
for(a=0;a<2;a++)
{
if(a==0)
```

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

```
{
j=ip[i];
p1[i]=p[j-1];
}
}
for(i=0;i<4;i++)
{
if(a==0)
k[i]=p1[i+4];
if(a==1)
{
k[i]=p[i+4];
for(i=0;i<8;i++
) b[i]=p[i];
}
}
for(i=0;i<8;i++)
{
j=ep[i];
p[i]=k[j-1];
}
for(i=0;i<8;i++)
```

{

```
if(a==0)
{
if(p[i]==k1[i])
k[i]=0;
else
k[i]=1;
}
if(a==1)
{
if(p[i]==k2[i])
k[i]=0;
else
k[i]=1;
}
}
j=0;
for(i=0;i<8;i=i+4)
{
if(k[i]==0&&k[i+3]==0)
{
t[j]=0;
j++;
```

```
}
```

```
if(k[i]==0&&k[i+3]==1)
{
t[j]=1;
j++;
}
if(k[i]==1&&k[i+3]==0)
{
t[j]=2;
j++;
}
if(k[i]==1&&k[i+3]==1)
{
t[j]=3;
j++;
}
if(k[i+1]==0&&k[i+2]==0)
{
t[j]=0;
j++;
}
if(k[i+1]==0&&k[i+2]==1)
{
```

```
t[j]=1;
```

```
j++;
}
if(k[i+1]==1&&k[i+2]==0)
{
t[j]=2;
j++;
}
if(k[i+1]==1&&k[i+2]==1)
{
t[j]=3;
j++;
}
}
s=s0[t[0]][t[1]];
h=s1[t[2]][t[3]];
if(s==0)
{
k[0]=0;
k[1]=0;
}
if(s==1)
{
```

```
k[0]=0;
```

```
k[1]=1;
}
if(s==2)
{
k[0]=1;
k[1]=0;
}
if(s==3)
{
k[0]=1;
k[1]=1;
}
if(h==0)
{
k[2]=0;
k[3]=0;
}
if(h==1)
{
k[2]=0;
k[3]=1;
}
```

if(h==2)

```
{
k[2]=1;
k[3]=0;
}
if(h==3)
{
k[2]=1;
k[3]=1;
}
for(i=0;i<4;i++)
{
j=p4[i];
p[i]=k[j-1];
}
for(i=0;i<4;i++)
{
if(a==0)
{
if(p1[i]==p[i])
k[i]=0;
else
k[i]=1;
```

```
}
```

```
if(a==1)
{
if(b[i]==p[i])
k[i]=0;
else
k[i]=1;
}
}
if(a==0)
{
for(i=0;i<4;i++
) p[i]=p1[i+4];
for(i=0;i<4;i++
) p[i+4]=k[i];
}
if(a==1)
{
for(i=4;i<8;i++
) k[i]=b[i];
for(i=0;i<8;i++
)
{
```

```
j=ip1[i];

p[i]=k[j-1];
```

```
}

}

}

printf("\n\nThe cipher text::");

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

printf("%d",p[i]);

getch(); }
```

OUTPUT
****

Simplified-DES

Enter the plain text of 8 bits length::

1

0

1

0

1

0

1

0

Enter the key of 10 bits length::

1

0

1

0

1

0

1

0

1

0

Key Generation::

key k1::11100100

key k2::01010011

The cipher text::01100110

Exercise:

1. **Take a plain text and implement AES algorithm**
2. **Implement a Blowfish algorithm.**

8. **Write a program to break the above DES coding.**

```
#include<stdio.h>

#include<conio.h>
```

```
#include<string.h>

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

p8[]={6,3,7,4,8,5,10,9},

p4[]={2,4,3,1};

int ip[]={2,6,3,1,4,8,5,7},

ipinv[]={4,1,3,5,7,2,8,6},

ep[]={4,1,2,3,2,3,4,1};

int s0[][4]={{1,0,3,2,},{3,2,1,0},{0,2,1,3,},{3,1,3,2}};

int s1[][4]={{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};

void permute(char op[],char ip[],int p[], int n)

{

int i;

for(i=0;i<n;i++

)

op[i]=ip[p[i]-1];

op[i]='\0';

}

void circularls(char pr[],int n)

{

int i;

char ch=pr[0];

for(i=0;i<n-1;i++
```

```
) pr[i]=pr[i+1];

pr[i]=ch;
```

```
}
void keygen(char k1[],char k2[],char key[])
{
char keytemp[11];
permute(keytemp,key,p10,10);
circularls(keytemp,5);
circularls(keytemp+5,5);
permute(k1,keytemp,p8,8);
circularls(keytemp,5);
circularls(keytemp,5);
circularls(keytemp+5,5);
circularls(keytemp+5,5);
permute(k2,keytemp,p8,8);
}
void xor(char op[],char ip[])
{
int i;
for(i=0;i<strlen(op)&&i<strlen(ip);i++
) op[i]=(op[i]-'0')^(ip[i]-'0')+'0';
}
void sbox(char op[],char ip[],int s[][4])
{
```

```
int value;
```

```
value=s[(ip[0]-'0')*2+(ip[3]-'0')][(ip[1]-'0')*2+(ip[2]-'0')];

op[0]=value/2+'0';

op[1]=value%2+'0';

op[2]='\0';

}

void fk(char op[],char ip[],char k[])

{

char l[5],r[5],tmp[9],tmp1[9],tmp2[9];

strncpy(l,ip,4);

l[4]='\0';

strncpy(r,ip+4,4);

r[4]='\0';

permute(tmp,r,ep,8);

xor(tmp,k);

sbox(tmp1,tmp,s0);

sbox(tmp2,tmp+4,s1);

strcat(tmp1,tmp2);

permute(tmp,tmp1,p4,4);

xor(tmp,l);

strcat(tmp,r);

strcpy(op,tmp);

}
```

```
void sw(char pr[])
```

```
{
char tmp[9];
strncpy(tmp,pr+4,4);
strncpy(tmp+4,pr,4);
tmp[8]='\0';
strcpy(pr,tmp);
}
void main()
{
char key[11],k1[9],k2[9],plain[9],cipher[9],tmp[9];
clrscr();
printf("enter 10 bit key:");
gets(key);
if(strlen(key)!=10) printf("invalid key length !!");
else
{
keygen(k1,k2,key);
printf("sub key k1::");
puts(k1);
printf("subkey k2::");
puts(k2);
printf("enter 8 bit plain text:");
```

```
gets(plain);
```

```
if(strlen(plain)!=8) printf("invalid length plain text !!");

permute(tmp,plain,ip,8);

fk(cipher,tmp,k1);

sw(cipher);

fk(tmp,cipher,k2);

permute(cipher,tmp,ipinv,8);

printf("cipher teaxt is::");

puts(cipher);

/* decryption process*/

permute(tmp,cipher,ip,8);

fk(plain,tmp,k2);

sw(plain);

fk(tmp,plain,k1);

permute(plain,tmp,ipinv,8);

printf("decrypted text is::");

puts(plain);

}

getch();

}
```

Exercise:1. Using AES algorithm decrypt the cipher

2. Implement HMAC algorithm

9. Using RSA algorithm encrypt a text data and Decrypt the same.

a) RSA encryption algorithm

Algorithm:

RSA encrypts messages through the following algorithm, which is divided into 3 steps:

1. Key Generation

I. Choose two distinct prime numbers p and q.

II. Find n such that n = pq.
n will be used as the modulus for both the public and private keys.

III. Find the totient of n,

ϕ(n) ϕ(n)=(p-1)(q-1).

IV. Choose an e such that 1 < e < ϕ(n), and such that e and ϕ(n) share no divisors other than 1 (e and ϕ(n) are relatively prime).
e is kept as the public key exponent.

V. Determine d (using modular arithmetic) which satisfies the congruence

relation de ≡ 1 (mod ϕ(n)).

In other words, pick d such that de - 1 can be evenly divided by (p-1)(q-1), the totient, or ϕ(n). This is often computed using the Extended Euclidean Algorithm, since e and ϕ(n) are relatively prime and d is to be the modular multiplicative inverse of e.
d is kept as the private key exponent.

The public key has modulus n and the public (or encryption) exponent e. The private key has modulus n and the private (or decryption) exponent d, which is kept secret.

2. Encryption

I. Person A transmits his/her public key (modulus n and exponent e) to Person B, keeping his/her private key secret.

II. When Person B wishes to send the message "M" to Person A, he first converts M to an integer such that 0 < m < n by using agreed upon reversible protocol known as a

padding scheme.

III. Person B computes, with Person A's public key information, the ciphertext c

corresponding to $c \equiv m^e \pmod n$.

IV. Person B now sends message "M" in ciphertext, or c, to Person A.

3. Decryption

I. Person A recovers m from c by using his/her private key exponent, d, by the

computation $m \equiv c^d \pmod n$.

II. Given m, Person A can recover the original message "M" by reversing the padding scheme.
This procedure works since
$c \equiv m^e \pmod n$,
$c^d \equiv (m^e)^d \pmod n$,
$c^d \equiv m^{de} \pmod n$.

By the symmetry property of mods we have that

$m^{de} \equiv m^{de} \pmod n$.

Since $de = 1 + k\phi(n)$, we can write

$m^{de} \equiv m^{1 + k\phi(n)} \pmod n$,
$m^{de} \equiv m(m^k)^{\phi(n)} \pmod n$,
$m^{de} \equiv m \pmod n$.

From Euler's Theorem and the Chinese Remainder Theorem, we can show that this is true for all m and the original message

$c^d \equiv m \pmod n$, is obtained.

Program:

```
#include<stdio.h>
main()
{
  int k,b,bin[20];
  int i;
  long int c,m,e,d,n;
  char ch;
  char in_file[20],out_file[20];
  FILE *in,*out;
```

```
  clrscr();
  printf("\n Enter any input text file name : ");
  gets(in_file);
  printf("\n Enter file name to store enc output : ");
  gets(out_file);
  in = fopen(in_file,"r");
  out =
  fopen(out_file,"w");

  printf("\n Enter values of e and n : ");
  scanf("%ld%ld",&e,&n);

  i=-1;
  b=e;
  while(b>0)
  {
     bin[++i] = b%2;
     b=b/2;
  }
  k=i;
  do
  {
     m = fgetc(in);
     d = 1;

     for( i=k; i>=0; i--)
     {
      d = (d*d) % n;
      if (bin[i] == 1)
         d = (d*m) % n;
     }

     fputc(d,out);
  }
  while(!feof(in));
  printf("\n File is encrfypted successfully.    ");
  getch();
}
```

INPUT/OUTPUT:

Enter any input text file name : inp.txt

Enter file name to store enc output : out.txt

Enter values of e and n :
7 187

File is encrfypted successfully....

C:\TURBOC2>type
inp.txt
abcdefghijklmnop123&*()

C:\TURBOC2>type out.txt
\§▒ÉTwë│`òp0âB¢l↓v◄/☼t.

b) RSA Decryption algorithm

```
#include<stdio.h>
main()
{
  int k,b,bin[20];
  int i;
  long int c,m,e,d,n;
  char ch;
  char in_file[20],out_file[20];
  FILE *in,*out;

  clrscr();
  printf("\n Enter any ciphertext file name : ");
  gets(in_file);
  printf("\n Enter file name to store dec output : ");
  gets(out_file);
  in = fopen(in_file,"r");
  out =
  fopen(out_file,"w");

  printf("\n Enter values of d and n : ");
  scanf("%ld%ld",&e,&n);

  i=-1;
  b=e;
  while(b>0)
  {
     bin[++i] = b%2;
     b=b/2;
  }
  k=i;
```

```
do
{
   m = fgetc(in);
   d = 1;
```

```
    for( i=k; i>=0; i--)
    {
     d = (d*d) % n;
     if (bin[i] == 1)
        d = (d*m) % n;
    }
   fputc(d,out);
  }
  while(!feof(in));
  printf("\n File is decrypted successfully.   ");
  getch();
 }
```

INPUT/OUTPUT:

Enter any ciphertext file name : out.txt

Enter file name to store dec output :

inp1.txt Enter values of d and n : 23 187

File is decrypted successfully....

C:\TURBOC2>type out.txt
\§ÉTwë│`òp0âB¢I↓v◄/☼t.

C:\TURBOC2>type inp1.txt
abcdefghijklmnop123&*()U

Exercise:

1. **Implement Diffie- Hellman cryptosystem using RSA algorithm.**
2. **Implement SHA-512 algorithm**