- **Data Communication and Networks**

Implement the following programs in C++: -

- **To find the given dotted decimal IP address is valid or not.**
- **To find a class in a given dotted decimal IP address.**
- **To implement Bit stuffing.**
- **To implement Character stuffing.**
- **To implement Parity check.**
- **To implement Checksum.**
- **To implement stop and wait ARQ protocol.**
- **To implement G0-back-N ARQ protocol.**
- **To implement selective-repeat ARQ protocol.**
- **To implement public-key encryption.**
- **To implement cyclic redundancy check (CRC).**
- **To implement vertical redundancy check (VRC).**
- **To implement Dijkstra's Algorithm.**
- **To implement Bellman-Ford Algorithm.**
- **To find all pair shortest path.**

1. **To find the given dotted decimal IP address is valid or not.**

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
int a(char ip[])
{
int dots=0;
char*ptr;
ptr=strtok(ip,".");
while(ptr!=NULL)
{
if(atoi(ptr)<0||atoi(ptr)>255)
return 0;
```

```
ptr=strtok(NULL,".");
if(ptr!=NULL)
dots++;
}
if(dots!=3)
return 0;
return 1;
}
void main()
{
char ip1[100];
int r;
clrscr();
cout<<"Enter IP address:";
cin>>ip1;
if(a(ip1))
cout<<"Given IP address is valid.";
else
cout<<"Given IP address is not valid.";
getch();
}
```

Output: -

RUN 1:

```
Enter the IP address: 192.168.1.1
Given IP address is valid.
```

RUN 2:

```
Enter the IP address: 300.168.1.1
Given IP address is not valid.
```

2. **To find a class in a given dotted decimal IP address.**

```
#include<iostream.h>
#include<stdlib.h>
```

```
#include<string.h>
#include<conio.h>
#include<stdio.h>
#define DELIM"."
int valid_digit(char*ip_str)
{
while(*ip_str)
{
if(*ip_str>='0'&&*ip_str<='9')
++ip_str;
else
return 0;
}
return 1;
}
int a(char*ip_str)
{
int i,num,dots=0;
char*ptr;
if(ip_str==NULL)
return 0;
ptr=strtok(ip_str,DELIM);
if(ptr==NULL)
return 0;
while(ptr)
{
if(!valid_digit(ptr))
return 0;
num=atoi(ptr);
if(num>=0&&num<=255)
{
ptr=strtok(NULL,DELIM);
if(ptr!=NULL)
++dots;
}else
```

```
return 0;
}
if(dots!=3)
return 0;
return 1;
}
void main()
{
char ip1[100],str1[2];
clrscr();
cout<<"Enter IP address:";
cin>>ip1;
if(a(ip1))
{
for(int i=0;i<=2;i++)
{
str1[i]=ip1[i];
}
int str;
str=atoi(str1);
if((str>=0)&&(str<128))
cout<<"Given IP address is class A.";
else if((str>=128)&&(str<192))
cout<<"Given IP address is class B.";
else if((str>=192)&&(str<224))
cout<<"Given IP address is class C.";
else if((str>=224)&&(str<240))
cout<<"Given IP address is class D.";
else
cout<<"Given IP address is class E.";
}
else
cout<<"Given IP address is invalid.";
int r;
getch();
```

```
}
```

Output :-

```
Enter IP address: 22.4.5.7
Given IP address is class A.

Enter IP address: 172.16.0.5
Given IP address is class B.

Enter IP address: 192.168.1.1
Given IP address is class C.

Enter IP address: 239.255.255.250
Given IP address is class D.

Enter IP address: 255.255.255.255
Given IP address is class E.

Enter IP address: 239.255.250
Given IP address is invalid.
```

3. To implement Bit stuffing.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
int a[20],b[30],i,j,k,count,n;
clrscr();
cout<<"Bit stuffing"<<endl;
cout<<"----------"<<endl;
cout<<"\nEnter the size of frame:";
cin>>n;
cout<<"Enter the data of frame(interms of 0's & 1's):";
for(i=0;i<n;i++)
```

```
{
cin>>a[i];
}
i=0;count=1;j=0;
do{
if(a[i]==1)
{
b[j]=a[i];
for(k=i+1;a[k]==1&&k<n&&count<5;k++)
{
j++;b[j]=a[k];count++;
if(count==5)
{
j++;b[j]=0;
}
i=k;
}
}
else
{
b[j]=a[i];
}
i++;
j++;
}while(i<n);
cout<<"\nAfter bit stuffing, the frame becomes:"<<endl;
for(i=0;i<j;i++)
cout<<b[i]<<" ";
getch();
}
```

Output :-

```
Bit stuffing
-----------
Enter the size of frame: 8
```

Enter the data of frame (interms of 0's & 1's):

1 1 1 1 1 1 0 1

After bit stuffing, the frame becomes:

1 1 1 1 1 0 1 0 1

4. **To implement Character stuffing.**

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
int i=0,j,n;
char a[20],b[20];
clrscr();
cout<<"Character Stuffing"<<endl;
cout<<"--------------------"<<endl;
cout<<"Enter the string:";
cin>>a;
n=strlen(a);
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
if(a[i]=='d'&&a[i+1]=='l'&&a[i+2]=='e')
{
b[j++]='d';
b[j++]='l';
b[j++]='e';
i=i+3;
}
```

```
else
{
b[j++]=a[i++];
}
}
b[j++]='d';
b[j++]='l';
b[j++]='e';
b[j++]='t';
b[j++]='x';
b[j]='\0';
cout<<"\nAfter character stuffing,the given string
becomes:";
cout<<b;
getch();
}
```

Output: -

```
Character Stuffing
---------------------------------------------------
Enter the string: COMPUTER
After character stuffing, the given string becomes:
dlestxCOMPUTERdletx
```

5. To implement Parity check.

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int i,n,m=0,a[20];
cout<<"Parity Check"<<endl;
cout<<"---------------"<<endl;
cout<<"\nEnter the number of bits:";
cin>>n;
cout<<"\nEnter the bit information(interms of 0's & 1's):";
```

```
for(i=1;i<=n;i++)
{
cin>>a[i];
}
for(i=1;i<=n;i++)
{
if(a[i]==1)
m++;
}
if(m%2==0)
{
cout<<"\nParity is even”;
a[n+1]=0;
}
else
{
cout<<"\nParity is odd”<<endl;
a[n+1]=1;
}
cout<<"Hence the bit information after parity check
becomes:"<<endl;
for(i=1;i<=n;i++)
{
cout<<a[i]<<" ";
}
getch();
}
```

Output: -

```
Parity Check
-----------------------------------------------------
Enter the number of bits: 4
Enter the bit information(interms of 0’s & 1’s): 1 1 0 1
Parity is odd.
Hence the bit information after parity check becomes:
1 1 0 1 1
```

6. To implement Checksum.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<stdio.h>
#define size 100
unsigned short int checksum(char[]);
void valch(unsigned short int check,char message[]);
int main()
{
clrscr();
cout<<"Implementing Checksum"<<endl;
cout<<"----------------------"<<endl;
cout<<"\nEnter any data:";
char message[size]={0};
cin>>message;
unsigned short int check=checksum(message);
getch();
valch(check,message);
getch();
return 0;
}
void valch(unsigned short int check, char message[])
{
unsigned short int t[size],ch=0;
int i,j,n;
cout<<endl<<endl;
cout<<"Checking checksum at reciever side:"<<endl;
for(i=0,j=0;i<strlen(message);i+=2,j++)
{
t[j]=message[i];
t[j]=t[j]<<8;
t[j]=t[j]+message[i+1];
}
n=j;
```

```
for(i=0;i<n;i++)
{
printf("%x\n",t[i]);
ch+=t[i];
}
printf("%x\n",check);
ch+=check;
printf("-----\n");
printf("%x\n",ch);
printf("1\n");
printf("=====\n");
ch++;
printf("%x\n",ch);
ch=~ch;
printf("%4x\n",ch);
if(ch==0)
cout<<"Checksum is valid...!";
else
cout<<"Checksum is invalid...!";
}
unsigned short int checksum(char message[])
{
unsigned short int t[size];
unsigned short int check=0;
int i,j,n;
cout<<"Generating checksum at transmitter side:"<<endl;
for(i=0,j=0;i<strlen(message);i+=2,j++)
{
t[j]=message[i];
t[j]=t[j]<<8;
t[j]=t[j]+message[i+1];
}
n=j;
for(i=0;i<n;i++)
{
```

```
printf("%x\n",t[i]);
check+=t[i];
}
printf("0000\n");
printf("-----\n");
printf("%x\n",check);
printf("1\n");
check+=1;
printf("----\n");
printf("%x\n",check);
check=~check;
printf("%x\n",check);
cout<<"Checksum generated is:";
printf("%x\n",check);
return check;
}
```

Output :-

```
Implementing Checksum
-----------------------------------------
Enter any data: 123
Generating checksum at transmitter side:
 3 1 3 2
 3 3 0 0
 0 0 0 0
-----------------------------------------
 6 4 3 2
 1
----------
 6 4 3 3
 9 b c c
Checksum generated is: 9 b c c

Checking checksum at receiver side:
 3 1 3 2
 9 b c c
----------
```

```
 F F F E
 1
-------------
 F F F F
 0
Checksum is valid...!
```

7. To implement stop and wait ARQ protocol.

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int i,j,f;
char ch;
cout<<"Stop and Wait ARQ protocol"<<endl;
cout<<"--------------------------"<<endl;
cout<<"Enter the total number of frames you want to
send:";
cin>>f;
if(f<=0)
{
cout<<"\nNo,frames have been requested.....!";
}
else
{
i=0;
j=0;
while(i<f)
{
cout<<"\nFrame "<<i<<" is sent....!";
cout<<"\nIs acknowledgement "<<j<<" recieved?(y/n):";
cin>>ch;
if(ch=='y'||ch=='Y')
{
```

```
cout<<"Acknowledgement recieved!"<<endl;
i++;
j++;
}
else
{
cout<<"\nSend again....!"<<endl;
}
}
cout<<"\nInformation sent successfully....!"<<endl;
}
getch();
}
```

Output :-

```
RUN 1:
Stop and Wait ARQ protocol
--------------------------------------------
Enter the total number of frames you want to send: 0
No, frames have been requested.....!

RUN 2:
Stop and Wait ARQ protocol
--------------------------------------------
Enter the total number of frames you want to send: 3
Frame 0 is sent....!
Is acknowledgement 0 is received? (y/n): y
Acknowledgement received!

Frame 1 is sent....!
Is acknowledgement 1 is received? (y/n): y
Acknowledgement received!

Frame 2 is sent....!
Is acknowledgement 2 is received? (y/n): n
```

```
Send again.....!

Frame 2 is sent....!
Is acknowledgement 2 is received? (y/n): y
Acknowledgement received!

Information sent successfully.....!
```

8. To implement Go-back-N ARQ protocol.

```
#include<dos.h>
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void cal();
void main()
{
int i,n,f,c,ans=0;
clrscr();
randomize();
abc:
cout<<"\nEnter the number of frames to send:";
cin>>n;
f=random(n+2);
cout<<"\nFrames are going to transmitted.";
for(i=1;i<=n;i++)
{
cout<<"\nFrame "<<i<<" sending";
cal();
}
for(i=1;i<=n;i++)
{
if(i==f)
{
cout<<"\n\tFrame is lost "<<f<<" Resend it.";
cal();
```

```
cout<<"\n\tAcknowledgement is not recieved."<<i;
cal();
ans=1;
}
else
{
cout<<"\nAcknowledgement Recieved."<<i;
cal();
}
}
if(ans)
{
ans=0;
cout<<"\nResend frames.";
cout<<"\nR U ready to resend all the frames once again(1-
yes or 0-exit):";
int resend;
cin>>resend;
if(resend==0)
goto xyz;
for(i=1;i<=n;i++)
{
cout<<"\nFrame "<<i<<" sending";
cal();
}
for(i=1;i<=n;i++)
{
cout<<"\nAcknowledgement Recieved "<<i<<" after
resending";
cal();
}
}
else
cout<<"\nUr data successfully sended";
```

```
cout<<"\nDo u want another frames(0-exit and 1-
continue...)?:";
cin>>c;
if(c==1)
goto abc;
else
goto xyz;
xyz:
cout<<"\nThank you";
cal();
getch();
}
void cal()
{
for(int i=0;i<3;i++)
{
sleep(i);
cout<<".";
}
}
```

Output:-

```
Enter the number of frames to send: 3

Frames are going to transmitted.
Frame 1 sending...
Frame 2 sending...
Frame 3 sending...
Acknowledgement Received 1...
        Frame is lost 2 Resend it...
        Acknowledgement is not received 2...
Acknowledgement Received 3...

Resend frames.
```

R U ready to resend all the frames once again(1-yes or 0-exit): 1
Frame 1 sending...
Frame 2 sending...
Frame 3 sending...
Acknowledgement Received 1 after resending...
Acknowledgement Received 2 after resending...
Acknowledgement Received 3 after resending...
Do u want another frames (0-exit and 1-continue...)?: 0

Thank you...

9. To implement Selective Repeat ARQ protocol.

```
#include<dos.h>
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void cal();
void main()
{
int i,n,f,c,ans=0;
clrscr();
cout<<"Selective Repeat ARQ Protocol"<<endl;
cout<<"-----------------------------"<<endl;
randomize();
abc:
cout<<"\nEnter the number of frames to send:";
cin>>n;
f=random(n+2);
cout<<"\nFrames are going to be transmitted...";
for(i=1;i<=n;i++)
{
cout<<"\nFrame"<<i<<" sending...";
cal();
}
```

```
for(i=1;i<=n;i++)
{
if(i==f)
{
cout<<"\n\tAcknowledgement"<<i<<" is not recieved...";
cal();
ans=1;
}
else
{
cout<<"\nAcknowledgement"<<i<<"recieved...";
cal();
}
}
if(ans)
{
ans=0;
again:
cout<<"\nR U ready to resend the selected frames(1-yes
or 0-exit):";
int resend;
cin>>resend;
cout<<"\nSending the frame...";
cal();
if(f==resend)
{
cout<<"\nAcknowledgement "<<f<<" recieved
successfully...!";
}
else
{
cout<<"\nUr sended frame is rejected.Resend it...";
cal();
goto again;
}
```

```
}
else
cout<<"\nUr data successfully sended...!";
cout<<"\nDo you want another frames to be transmitted(0-
exit and 1-continue...)?:";
cin>>c;
if(c==1)
goto abc;
else
goto xyz;
xyz:
cout<<"\nThank you!!!";
cal();
getch();
}
void cal()
{
for(int i=0;i<4;i++)
{
sleep(i);
cout<<".";
}
}
```

Output: -

```
Selective Repeat ARQ Protocol
-----------------------------------------------------
Enter the number of frames to send: 4
Frames are going to be transmitted...
Frame 1 sending...
Frame 2 sending...
Frame 3 sending...
Frame 4 sending...
   Acknowledgement 1 is not received...
Acknowledgement 2 received...
```

Acknowledgement 3 received...
Acknowledgement 4 received...
R U ready to resend the selected frames (1-yes or 0-exit):
1

Sending the frame...
Acknowledgement 1 received successfully...!
Do you want another frames to be transmitted (0-exit and 1-continue): 0

Thank you!!!

10. To implement public-key encryption.

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
long int p,q,fn,e,s=0,d=1,n;
char m[100];
unsigned long int C=0,c;
clrscr();
cout<<"\nEnter the value of p & q:";
cin>>p>>q;
fn=(p-1)*(q-1);
cout<<"\NEnter the encryption key:";
cin>>e;
do
{
s=(d*e)%fn;
d++;
}
while(s!=1);
```

```
d=d-1;
cout<<"\nPublic key("<<e<<0<<")";
cout<<"\nPrivate key("<<d<<0<<")";
cout<<"\nEnter message:";
cin>>m;
randomize();
C=random(100000);
cout<<C;
cout<<"\nEnter clipertext:";
cin>>c;
cout<<"\nPlane text:";
if(C==c)
cout<<m;
getch();
}
```

Output: -

```
Enter the value of p & q: 5 15
Enter the encryption key: 5
Public key(50)
Private key(450)
Enter message: anu
4294951781
Enter cliper text: 4294951781
Plane text: anu
```

11. To implement Cyclic Redundancy Check (CRC).

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
void main()
{
int i,j,n,g,a,arr[20],gen[20],temp[20];
clrscr();
cout<<"Cyclic Redundancy Check"<<endl;
```

```
cout<<"----------------------"<<endl;
cout<<"\nEnter the number of data bits:";
cin>>n;
cout<<"Enter data bits(0 or 1):";
for(i=0;i<n;i++)
{
cin>>arr[i];
}
cout<<"\nEnter the size of generator bits:";
cin>>g;
cout<<"\nEnter generator bits:";
for(i=0;i<g;i++)
{
cin>>gen[i];
}
for(i=n;i<n+g-1;i++)
{
arr[i]=0;
}
a=n+g-1;
for(i=0;i<a;i++)
{
temp[i]=arr[i];
}
for(i=0;i<n;i++)
{
if(temp[i]==1)
{
for(j=0;j<g;j++)
{
temp[i+j]=temp[i+j]^gen[j];
}
}
}
cout<<"\nCRC bits are:";
```

```
for(i=n;i<a;i++)
{
cout<<temp[i];
}
cout<<"\nFinal data to be transmitted:";
for(i=0;i<n;i++)
cout<<arr[i];
for(i=n;i<a;i++)
cout<<temp[i];
getch();
}
```

Output: -

```
Cyclic Redundancy Check
-----------------------------------------------
Enter the number of data bits: 8
Enter data bits: 1 0 1 0 0 0 0 1
Enter the size of generator: 4
Enter generator bits: 1 0 0 1
CRC bits are: 1 1 1
Final data to be transmitted: 1 0 1 0 0 0 0 1 1 1 1
```

12. To implement Vertical Redundancy Check (VRC).

```
#include<iostream.h>
#include<conio.h>
void main()
{
int i,j,row,col,a[20][20];
cout<<"Vertical Redundancy Check"<<endl;
cout<<"----------------------"<<endl;
cout<<"Enter the number of bits in row:";
cin>>row;
cout<<"\nEnter the number of bits in col:";
cin>>col;
```

```
cout<<"\nEnter the bit information(interms of 0's &
1's):"<<endl;
for(i=1;i<=row;i++)
{
for(j=1;j<=col;j++)
{
cin>>a[i][j];
}
}
for(i=1;i<=row;i++)
{
int m=0;
for(j=1;j<=col;j++)
{
if(a[i][j]==1)
{
m++;
}
}
int k=m%2;
if(k==0)
{
a[i][col+1]=0;
}
else
{
a[i][col+1]=1;
}
}
cout<<"After VRC,above bits are represented as:"<<endl;
for(i=1;i<=row;i++)
{
for(j=1;j<=col+1;j++)
{
cout<<a[i][j]<<" ";
```

```
}
cout<<endl;
}
getch();
}
```

Output: -

```
Vertical Redundancy Check
-----------------------------------------
Enter the number of bits in row: 3
Enter the number of bits in col: 4
Enter the bit information (interms of 0's & 1's):
1 1 0 1
1 1 1 0
1 1 0 0
After VRC, above bits are represented are:
1 1 0 1 1
1 1 1 0 1
1 1 0 0 0
```

13. To implement Dijkstra's Algorithm.

```
#include<iostream.h>
#include<conio.h>
#define INFINITY 99
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main()
{
int g[MAX][MAX],i,j,n,u;
clrscr();
cout<<"Dijkstra's Algorithm"<<endl;
cout<<"--------------------"<<endl;
cout<<"\nEnter the total number of vertices:";
cin>>n;
cout<<"\nEnter the cost matrix:\n";
```

```
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
cin>>g[i][j];
cout<<"\nEnter the initial vertex:";
cin>>u;
dijkstra(g,n,u);
getch();
return 0;
}
void dijkstra(int G[MAX][MAX],int n,int startnode)
{
int cost[MAX][MAX],distance[MAX],pred[MAX];
int visited[MAX],count,mindistance,nextnode,i,j;
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(G[i][j]==0)
cost[i][j]=INFINITY;
else
cost[i][j]=G[i][j];
for(i=1;i<=n;i++)
{
distance[i]=cost[startnode][i];
pred[i]=startnode;
visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
count=1;
while(count<n-1)
{
mindistance=INFINITY;
for(i=1;i<=n;i++)
if(distance[i]<mindistance&&!visited[i])
{
mindistance=distance[i];
```

```
nextnode=i;
}
visited[nextnode]=1;
for(i=1;i<=n;i++)
if(!visited[i])
if(mindistance+cost[nextnode][i]<distance[i])
{
distance[i]=mindistance+cost[nextnode][i];
pred[i]=nextnode;
}
count++;
}
cout<<"\nShortest path is:"<<endl;
for(i=1;i<=n;i++)
if(i!=startnode)
{
cout<<"V "<<startnode<<"to V "<<i<<",cost is=>
"<<distance[i]<<endl;
}
}
```

Output: -

```
Dijkstra's Algorithm
------------------------------------
Enter the total number of vertices: 3

Enter the cost matrix:
0 2 4
3 0 1
6 1 0

Enter the initial vertex: 1

Shortest path is:
V1 to V2, cost is => 2
```

V1 to V3, cost is => 3

14. To implement Bellman – Ford Algorithm.

```
#include<iostream.h>
#include<conio.h>
#define MAX 100
#define INF 9999
struct Edge
{
int src,dest,weight;
};
void main()
{
int V,E;
cout<<"Enter number of vertices:";
cin>>V;
cout<<"Enter number of edges:";
cin>>E;
Edge edges[MAX];
int i,j;
cout<<"Enter source,destination and weight of each
edge:\n";
for(i=0;i<E;i++)
{
cin>>edges[i].src>>edges[i].dest>>edges[i].weight;
}
int source;
cout<<"Enter source vertex:";
cin>>source;
int dist[MAX];
for(i=0;i<V;i++)
{
dist[i]=INF;
}
dist[source]=0;
```

```
for(i=1;i<=V-1;i++)
{
for(j=0;j<E;j++)
{
if(dist[edges[j].src]!=INF &&
dist[edges[j].src]+edges[j].weight<dist[edges[j].dest])
{
dist[edges[j].dest]=dist[edges[j].src]+edges[j].weight;
}
}
}
for(j=0;j<E;j++)
{
if(dist[edges[j].src]!=INF &&
dist[edges[j].src]+edges[j].weight<dist[edges[j].dest])
{
cout<<"Negative weight cycle exists!";
getch();
return;
}
}
cout<<"\nVertex  Distance\n";
for(i=0;i<V;i++)
{
cout<<i<<"\t"<<dist[i]<<endl;
}
getch();
}
```

Output: -

```
Enter number of vertices: 5
Enter number of edges; 8
Enter source, destination and weight of each edge:
0 1 -1
0 2 4
```

```
1 2 3
1 3 2
1 4 2
3 2 5
3 1 1
4 3 -3
Enter source vertex: 0
Vertex Distance
0       0
1       -1
2       2
3       -2
4       1
```

15. To find all pair shortest path.

```
#include<iostream.h>
#include<conio.h>
#define INF 999
void main()
{
int n,i,j,k;
int cost[10][10],dist[10][10];
clrscr();
cout<<"Enter the number of vertices:";
cin>>n;
cout<<"Enter the cost matrix:\n";
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
cin>>cost[i][j];
dist[i][j]=cost[i][j];
}
}
```

```
for(k=0;k<n;k++)
{
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if(dist[i][k]+dist[k][j]<dist[i][j])
{
dist[i][j]=dist[i][k]+dist[k][j];
}
}
}
}
cout<<"\nThe shortest distance matrix is:\n";
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
cout<<dist[i][j]<<" ";
}
cout<<endl;
}
getch();
}
```

Output: -

```
Enter the number of vertices: 3
Enter the cost matrix:
0 4 11
6 0 2
3 999 0
```

The shortest distance matrix is:

0 4 6

5 0 2

3 7 0