3a. Design and implement C/C++ Program to solve All-Pairs Shortest Paths problem using Floyd's algorithm.

Algorithm:

Floyd(W[1..n,1..n])

// Implements Floyd‟s algorithm for the all-pairs shortest paths problem
// Input : The weight matrix W of a graph with no negative-length cycle
// Output : The distance matrix of shortest path's lengths

D ← W
for k←1 to n do
for i ← 1 to n do
for j ← 1 to n do
D[i,j] ← min {D[i, j], D[i,k]+D[k, j]}
return D

Program:

```
#include<stdio.h>
int min(int,int);
void floyds(int p[10][10],int n)
{
        int i,j,k;
        for(k=1;k<=n;k++)
        {
                for(i=1;i<=n;i++)
                {
                        for(j=1;j<=n;j++)
                        {
                                if(i==j)
                                        p[i][j]=0;
                                else
                                        p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
                        }
                }
        }
}
int min(int a,int b)
{
         if(a<b)
                return(a);
```

```
        else
                return(b);
}
void main()
{
                int p[10][10],w,n,u,v,i,j;;
                printf("\n Enter the number of vertices:");
                scanf("%d",&n);
                printf("\n Matrix of input data:\n");
                for(i=1;i<=n;i++)
                {
                        for(j=1;j<=n;j++)
                                scanf("%d",&p[i][j]);
                                if(p[i][j]==0)
                                p[i][j]=999;

                }
                floyds(p,n);
                 printf("\n The shortest paths are:\n");
                 for(i=1;i<=n;i++)
                 for(j=1;j<=n;j++)
                {
                        if(i!=j)
                        printf("\n <%d,%d>=%d",i,j,p[i][j]);
                  }
}
```

OUTPUT:

1.

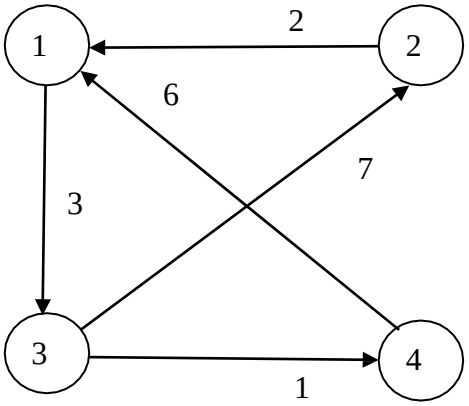
```
 Enter the number of vertices:4

 Matrix of input data:
0 3 999 7
8 0 2 999
5 999 0 1
2 999 999 0

 The shortest paths are:
<1,2>=3 <1,3>=5 <1,4>=6
<2,1>=5 <2,3>=2 <2,4>=3
<3,1>=3 <3,2>=6 <3,4>=1
<4,1>=2 <4,2>=5 <4,3>=7
```

2.

Enter the number of vertices
4



Enter the Cost Matrix (999 for infinity)

0	999	3	999
2	0	999	999
999	7	0	1
6	999	999	0

The All Pair Shortest Path Matrix is:

0	10	3	4
2	0	5	6
7	7	0	1
6	16	9	0

3b. Design and implement C/C++ Program to find the transitive closure using Warshal's algorithm.

Algorithm:

Require: Adjacency matrix M_{nXn} of digragh D

Ensure: D^t the transitive closure of D

```
for k ⟵ 1 until n do
    for i ⟵ 1 until n do
        for j ⟵ 1 until n do
            M [i,j] ⟵ M[i,j] or (M[i,j] and M[k,j])
        end for
    end for
end for
```

Program:

```
#include<stdio.h>
int max(int, int);
void warshal(int p[10][10], int n)
{
    int i, j, k;
    for (k = 1; k <= n; k++)
        for (i = 1; i <= n; i++)
            for (j = 1; j <= n; j++)
                p[i][j] = max(p[i][j], p[i][k] && p[k][j]);

}

int max(int a, int b)
{
    if (a > b)
        return (a);
    else
        return (b);
}

void main()
{
    int p[10][10] = { 0 }, n, e, u, v, i, j;
    printf("\n Enter the number of vertices:");
    scanf("%d", &n);
    printf("\n Enter the Matrix of input data: \n");
```

```
    for (i = 1; i <= n; i++)
       {
                  for (j = 1; j <= n; j++)
                 scanf("%d", &p[i][j]);
       }
    warshal(p, n);
    printf("\n Transitive closure: \n");
    for (i = 1; i <= n; i++)
       {
                  for (j = 1; j <= n; j++)
                 printf("%d\t", p[i][j]);
                  printf("\n");
       }
}
```

OUTPUT:

1.

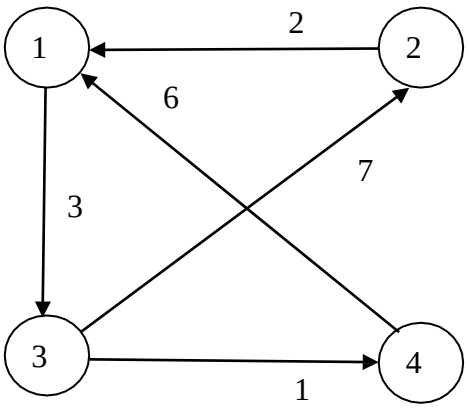
```
 Enter the number of vertices:4

 Enter the Matrix of input data:
0 1 0 0
0 0 0 1
0 0 0 0
0 1 0 1

 Transitive closure:
0       1       0       1
0       1       0       1
0       0       0       0
0       1       0       1
```

2.

Enter the number of vertices
4



Enter the Cost Matrix (999 for infinity)

0	999	3	999
2	0	999	999
999	7	0	1
6	999	999	0

The All Pair Shortest Path Matrix is:

0	10	3	4
2	0	5	6
7	7	0	1
6	16	9	0