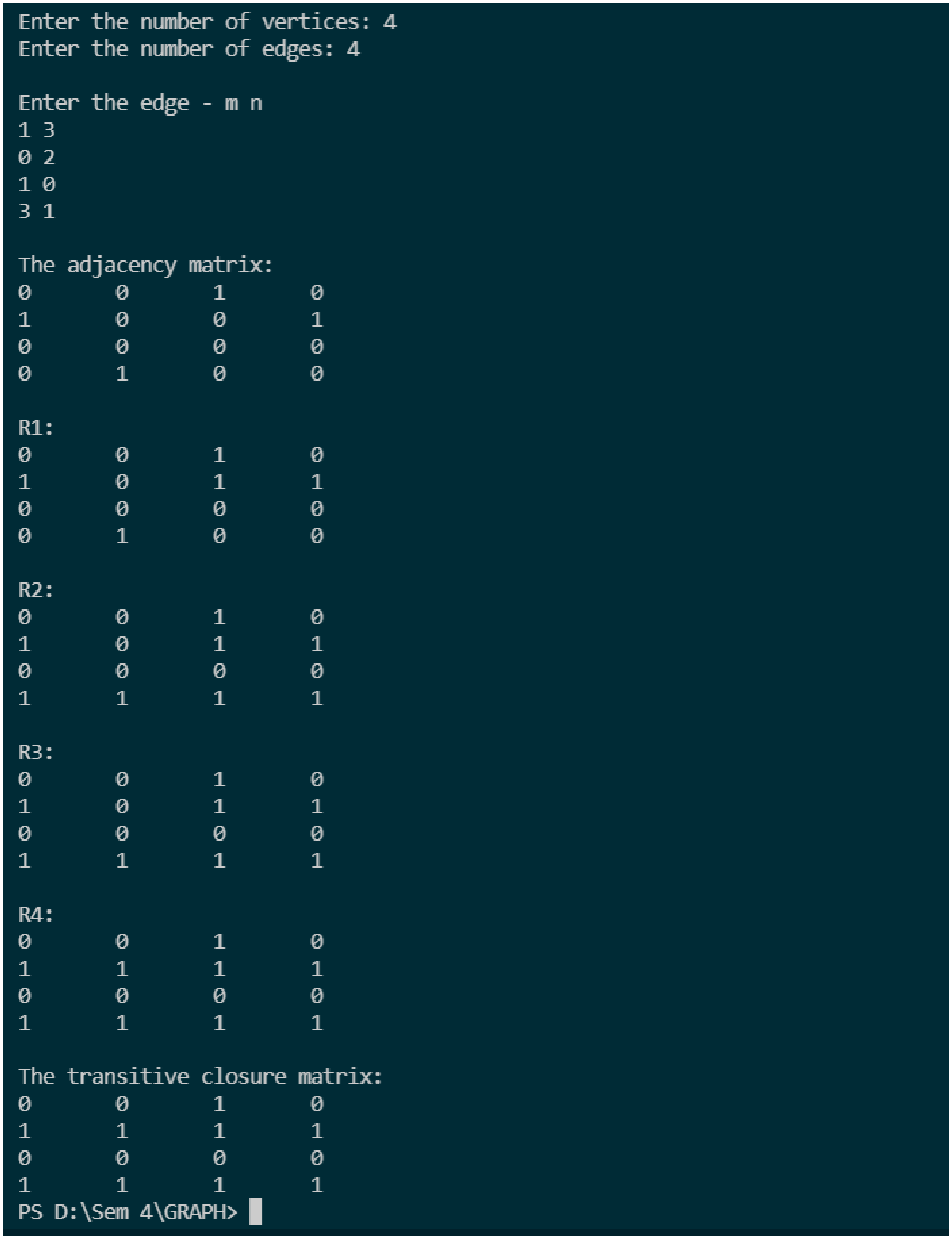
Program statement: Write programs in C for the following, considering a directed graph: 1) Warshall's Algorithm to find the transitive closure

Source code:

```
// A C Program for Warshall algorithm
// representation of graphs using adjacency matrix
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void print(int v, int a[v][v])
{
    int i,j;
    for( i=0; i<v; i++)
    {
        for( j=0; j<v; j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }
}

void war(int v, int a[v][v])
{
    int i,j,k=0;
    while( k<v )
    {
        for( i=0; i<v; i++ )
        {
            for( j=0; j<v; j++ )
            {
                if( a[i][k]==1 && a[k][j]==1 )
                    a[i][j]=1;
            }
        }
        printf("\nR%d:\n",(k+1));
        print(v,a);
        k++;
    }
```

```c
    printf("\nThe transitive closure matrix:\n");
    print(v,a);
}

int main()
{
    int v,e;
    printf("Enter the number of vertices: ");
    scanf("%d",&v);
    printf("Enter the number of edges: ");
    scanf("%d",&e);
    int a[v][v];
    int i,j,p;
    for( i=0; i<v; i++ )
    {
        for( j=0; j<v; j++ )
        {
            a[i][j]=0;
        }
    }
    printf("\nEnter the edge - m n\n");
    for( p=0; p<e; p++ )
    {
        scanf("%d%d", &i, &j);
        a[i][j]=1;
    }
    printf("\nThe adjacency matrix:\n");
    print(v, a);
    war(v, a);
}
```

Output:



```
Enter the number of vertices: 4
Enter the number of edges: 4

Enter the edge - m n
1 3
0 2
1 0
3 1

The adjacency matrix:
0       0       1       0
1       0       0       1
0       0       0       0
0       1       0       0

R1:
0       0       1       0
1       0       1       1
0       0       0       0
0       1       0       0

R2:
0       0       1       0
1       0       1       1
0       0       0       0
1       1       1       1

R3:
0       0       1       0
1       0       1       1
0       0       0       0
1       1       1       1

R4:
0       0       1       0
1       1       1       1
0       0       0       0
1       1       1       1

The transitive closure matrix:
0       0       1       0
1       1       1       1
0       0       0       0
1       1       1       1
PS D:\Sem 4\GRAPH>
```

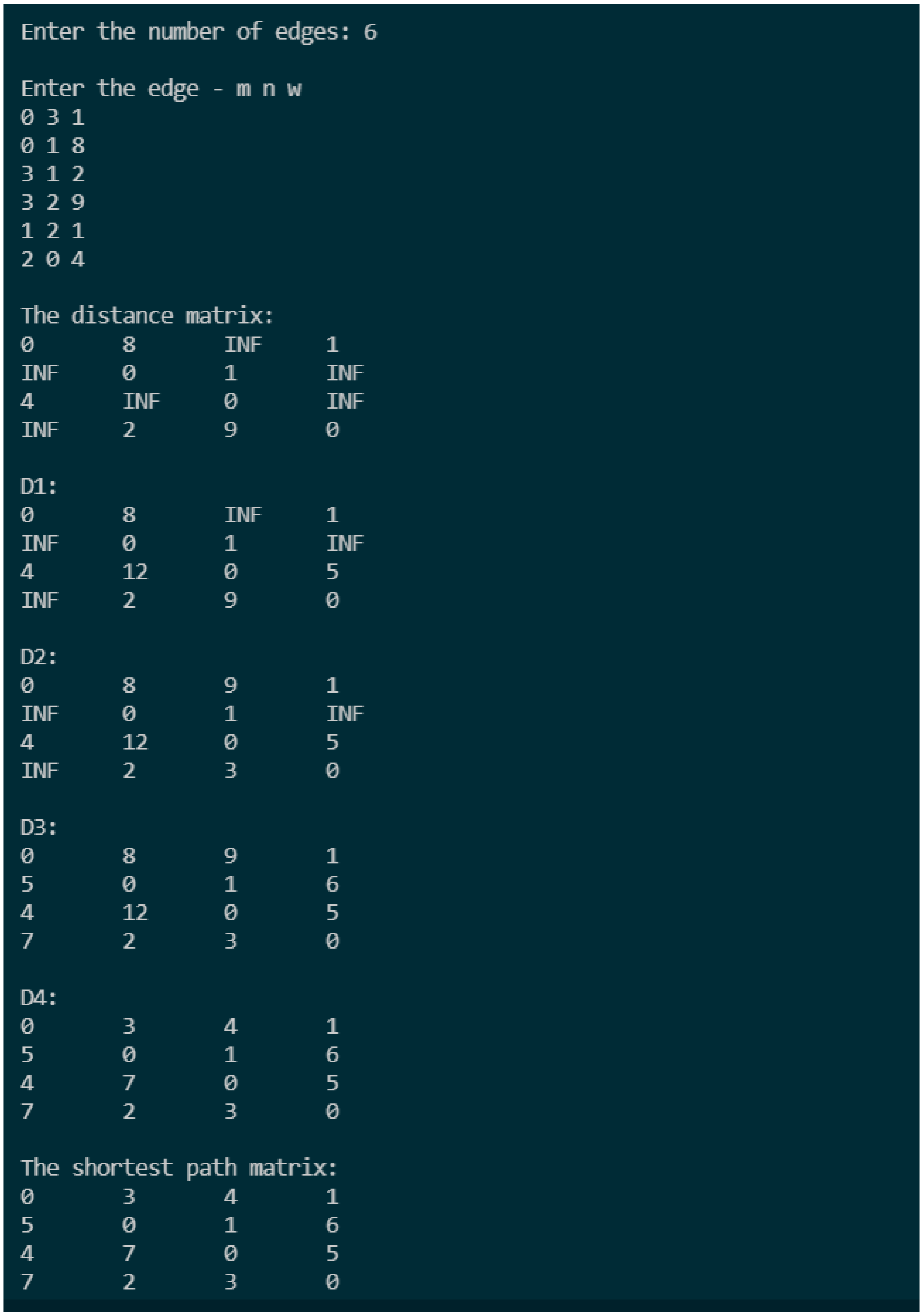
Program statement: 2) Floyd's Algorithm to find All-pairs shortest-path in a weighted graph.

Source code:

```
// A C Program for Floyd Warshall algorithm
// representation of graphs using adjacency matrix
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void print(int v, int a[v][v])
{
    int i,j;
    for( i=0; i<v; i++)
    {
        for( j=0; j<v; j++)
        {
            if(a[i][j]==9999)
                printf("INF\t");
            else
                printf("%d\t",a[i][j]);
        }
        printf("\n");
    }
}

void fw(int v, int a[v][v])
{
    int i,j,k=0;
    while( k<v )
    {
        for( i=0; i<v; i++ )
        {
            for( j=0; j<v; j++ )
            {
                if( i==j )
                    continue;
                else
                    a[i][j]=fmin(a[i][j],(a[i][k]+a[k][j]));
            }
```

```c
        }
        printf("\nD%d:\n",(k+1));
        print(v,a);
        k++;
    }
    printf("\nThe shortest path matrix:\n");
    print(v,a);
}

int main()
{
    int v,e;
    printf("Enter the number of vertices: ");
    scanf("%d",&v);
    printf("Enter the number of edges: ");
    scanf("%d",&e);
    int a[v][v];
    int i,j,p,w;
    for( i=0; i<v; i++ )
    {
        for( j=0; j<v; j++ )
        {
            if( i!=j )
                a[i][j]=9999;
            else
                a[i][j]=0;
        }
    }
    printf("\nEnter the edge - m n w\n");
    for( p=0; p<e; p++ )
    {
        scanf("%d%d%d", &i, &j, &w);
        if( a[i][j] > 0 && a[i][j] < w)
            continue;
        else{
            a[i][j]=w;
        }
    }
    printf("\nThe distance matrix:\n");
    print(v, a);
    fw(v, a);
}
```

Output:



```
Enter the number of edges: 6

Enter the edge - m n w
0 3 1
0 1 8
3 1 2
3 2 9
1 2 1
2 0 4

The distance matrix:
0       8       INF     1
INF     0       1       INF
4       INF     0       INF
INF     2       9       0

D1:
0       8       INF     1
INF     0       1       INF
4       12      0       5
INF     2       9       0

D2:
0       8       9       1
INF     0       1       INF
4       12      0       5
INF     2       3       0

D3:
0       8       9       1
5       0       1       6
4       12      0       5
7       2       3       0

D4:
0       3       4       1
5       0       1       6
4       7       0       5
7       2       3       0

The shortest path matrix:
0       3       4       1
5       0       1       6
4       7       0       5
7       2       3       0
```