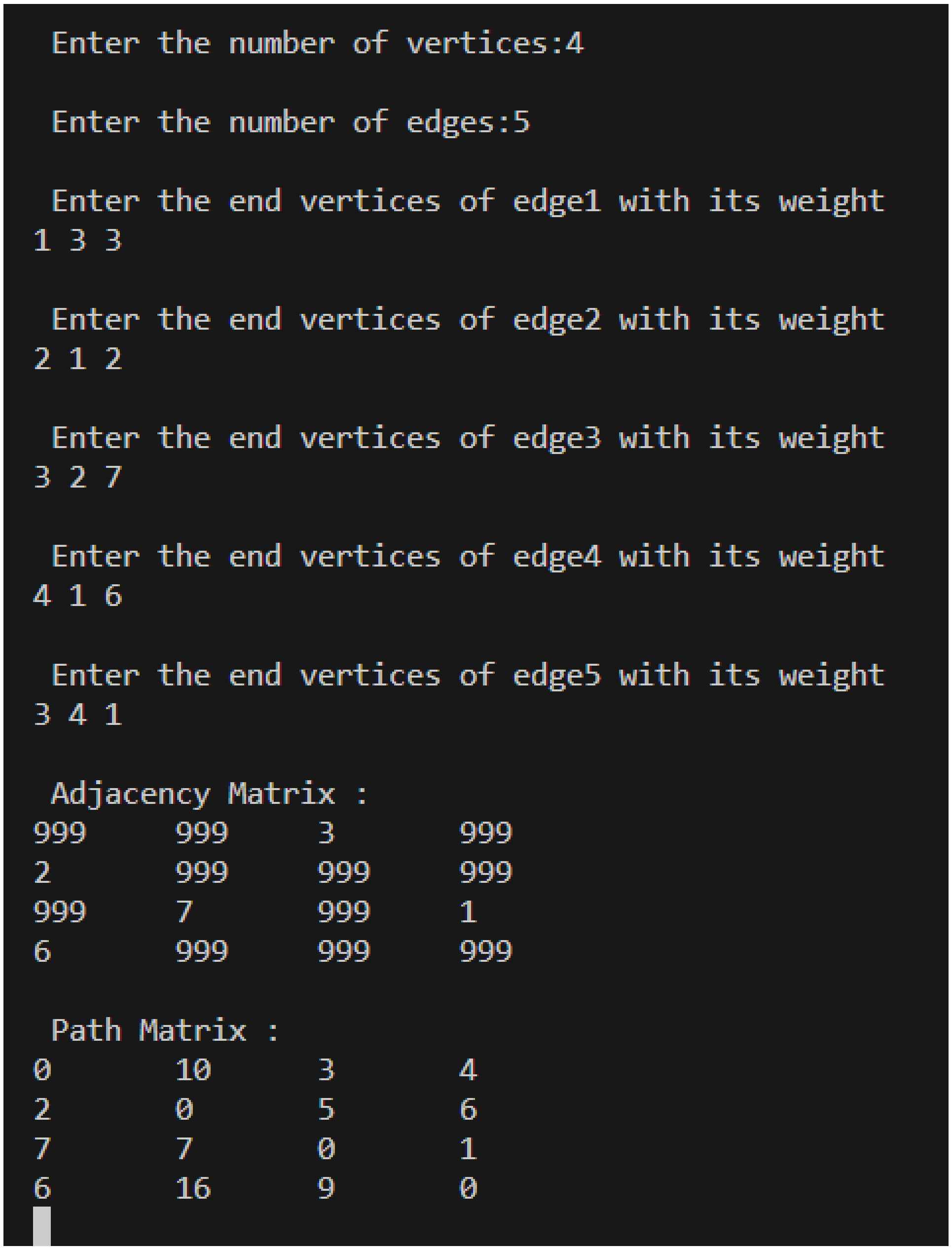
FLOYD’S ALGORITHM

```
#include <stdio.h>
#include <conio.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, e, u, v, i, j;
   printf("\n Enter the number of vertices:");
   scanf("%d", &n);
   printf("\n Enter the number of edges:\n");
   scanf("%d", &e);
   for (i = 1; i <= n; i++)
   {
        for (j = 1; j <= n; j++)
            p[i][j] = 999;
    }
    for (i = 1; i <= e; i++)
    {
        printf("\n Enter the end vertices of edge%d with its weight \n", i);
        scanf("%d%d%d", &u, &v, &w);
        p[u][v] = w;
    }
    printf("\n Matrix of input data:\n");
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
            printf("%d \t", p[i][j]);
        printf("\n");
    }
    floyds(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");
    }
    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]);
        }
    getch();
}
```

OUTPUT:



```
 Enter the number of vertices:4

 Enter the number of edges:5

 Enter the end vertices of edge1 with its weight
1 3 3

 Enter the end vertices of edge2 with its weight
2 1 2

 Enter the end vertices of edge3 with its weight
3 2 7

 Enter the end vertices of edge4 with its weight
4 1 6

 Enter the end vertices of edge5 with its weight
3 4 1

 Adjacency Matrix :
999     999     3       999
2       999     999     999
999     7       999     1
6       999     999     999

 Path Matrix :
0       10      3       4
2       0       5       6
7       7       0       1
6       16      9       0
```