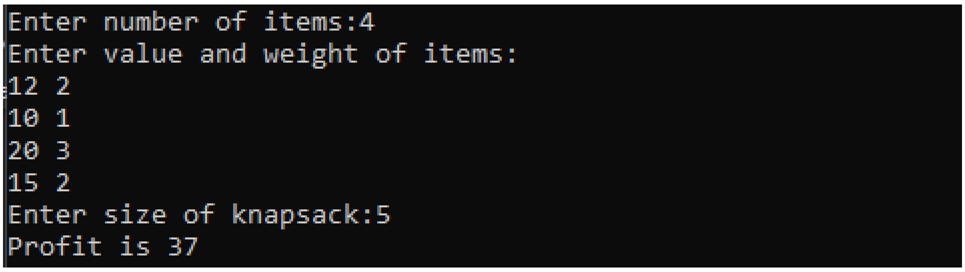
Lab-6

1) Implement 0/1 Knapsack problem using dynamic programming.

```
#include<stdio.h>
int max(int a, int b) { return (a > b)? a : b; }
int knapSack(int W, int wt[], int val[], int n)
{
   int i, w;
   int K[n+1][W+1];
   for (i = 0; i <= n; i++)
   {
       for (w = 0; w <= W; w++)
       {
           if (i==0 || w==0)
               K[i][w] = 0;
           else if (wt[i-1] <= w)
                 K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);
           else
                 K[i][w] = K[i-1][w];
       }
   }
   return K[n][W];
}
int main()
{
    int i, n, val[20], wt[20], W;

    printf("Enter number of items:");
    scanf("%d", &n);

    printf("Enter value and weight of items:\n");
    for(i = 0;i < n; ++i){
     scanf("%d%d", &val[i], &wt[i]);
    }


    printf("Enter size of knapsack:");
    scanf("%d", &W);


    printf("Profit is %d", knapSack(W, wt, val, n));
    return 0;
}
```

OUTPUT

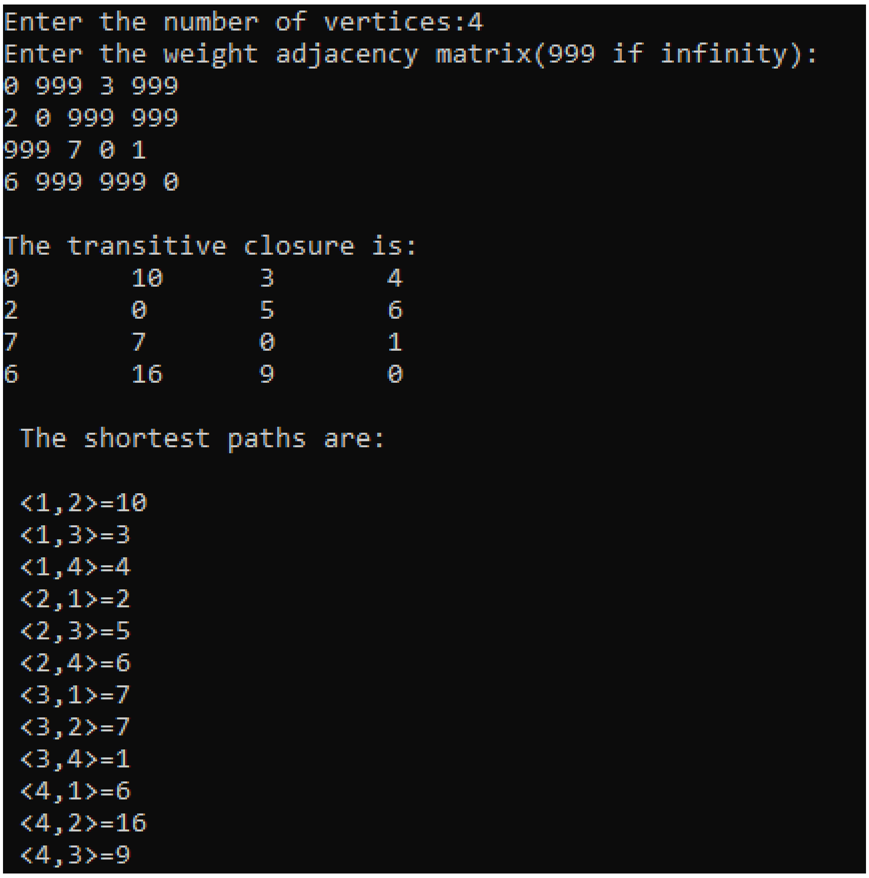


```
Enter number of items:4
Enter value and weight of items:
12 2
10 1
20 3
15 2
Enter size of knapsack:5
Profit is 37
```

2) Implement All Pair Shortest paths problem using Floyd’s algorithm.

```
#include<stdio.h>
#include<conio.h>
int c[10][10],d[10][10],n;
void floyd()
{
    int i,j,k;
    for(i=1;i<=n;i++)
    {


        for(j=1;j<=n;j++)
        {
            d[i][j]=c[i][j];
        }
    }
    for(k=1;k<=n;k++)
    {
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                d[i][j]=(d[i][j]<d[i][k]+d[k][j])?d[i][j]:(d[i][k]+d[k][j]);
            }
        }

    }
}
void main()
{
    int i,j;
    printf("Enter the number of vertices:");
    scanf("%d",&n);
    printf("Enter the weight adjacency matrix(999 if infinity):\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&c[i][j]);
        }
    }
    floyd();
    printf("\nThe transitive closure is:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
           printf("%d\t",d[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&&d[i][j]!=999)
                          printf("\n <%d,%d>=%d",i,j,d[i][j]);
           }
}
```

OUTPUT



```
Enter the number of vertices:4
Enter the weight adjacency matrix(999 if infinity):
0 999 3 999
2 0 999 999
999 7 0 1
6 999 999 0

The transitive closure is:
0       10      3       4
2       0       5       6
7       7       0       1
6       16      9       0

 The shortest paths are:

 <1,2>=10
 <1,3>=3
 <1,4>=4
 <2,1>=2
 <2,3>=5
 <2,4>=6
 <3,1>=7
 <3,2>=7
 <3,4>=1
 <4,1>=6
 <4,2>=16
 <4,3>=9
```