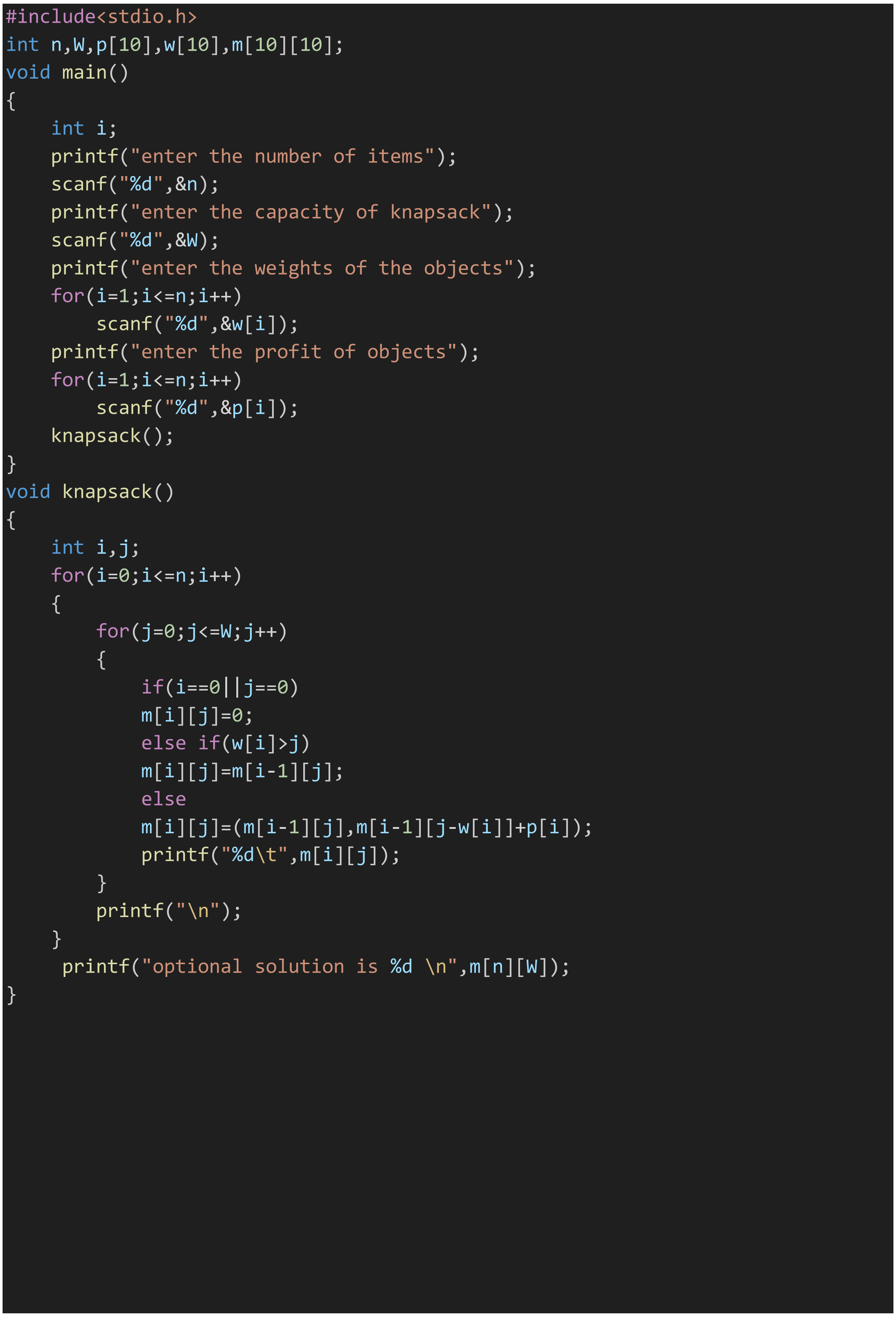

```c
#include<stdio.h>
int n,W,p[10],w[10],m[10][10];
void main()
{
    int i;
    printf("enter the number of items");
    scanf("%d",&n);
    printf("enter the capacity of knapsack");
    scanf("%d",&W);
    printf("enter the weights of the objects");
    for(i=1;i<=n;i++)
        scanf("%d",&w[i]);
    printf("enter the profit of objects");
    for(i=1;i<=n;i++)
        scanf("%d",&p[i]);
    knapsack();
}
void knapsack()
{
    int i,j;
    for(i=0;i<=n;i++)
    {
        for(j=0;j<=W;j++)
        {
            if(i==0||j==0)
            m[i][j]=0;
            else if(w[i]>j)
            m[i][j]=m[i-1][j];
            else
            m[i][j]=(m[i-1][j],m[i-1][j-w[i]]+p[i]);
            printf("%d\t",m[i][j]);
        }
        printf("\n");
    }
     printf("optional solution is %d \n",m[n][W]);
}
```

```c
#include<stdio.h>
void main()
{
    int i,j,s,dist[10],cost[10][10],n;
    printf("enter the num of vertex");
    scanf("%d",&n);
    printf("enter the cost matrix");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
            {
                cost[i][j]=999;
            }
        }
    }
    printf("enter the source vertex");
    scanf("%d",&s);
    dijk(n,cost,dist,s);
    for(i=1;i<=n;i++)
    {
        if(s!=i)
        {
            printf("%d->%d=%d\n",s,i,dist[i]);
        }
    }
}
void dijk(int n,int cost[10][10],int dist[10],int s)
{
    int i,v,min,count=1,visted[10];
    for(i=1;i<=n;i++)
    {
        visted[i]=0;
        dist[i]=cost[s][i];
    }
 visted[s]=1;
 dist [s]=0;
 while (count<=n)
 {
    min=999;
    for(i=1;i<=n;i++)
    {
        if(dist[i]<min && visted[i]!=1)
        {
            min=dist[i];
            v=i;
        }
    }
    visted[v]=1;
    count++;
    for(i=1;i<=n;i++)
    {
        if(dist[i]>dist[v]+cost[v][i])
        {
            dist[i]=dist[v]+cost[v][i];
        }
    }
 }
}
```

topology

```
#include<stdio.h>
int n,indegree[10],a[10][10];
void main()
{
    int i,j;
    printf("enter the number of vertices");
    scanf("%d",&n);
    printf("enter the matrix");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    topology();
}
void find_indegree()
{
    int i,j,sum;
    for (j = 1; j < n; j++)
    {
        sum=0;
        for(i=1;i<=n;i++)
        {
            sum=sum+a[i][j];
        }
        indegree[j]=sum;
    }
}
void topology()
{
    int i,u,v,k=0,top=-1,s[10],t[10];
    find_indegree();
    for(i=1;i<=n;i++)
    {
        if(indegree[i]==0)
        {
            s[++top]=i;

        }
    }
    while (top!=-1)
    {
        u=s[top--];
        t[k++]=u;
        for(v=1;v<=n;v++)
        {
            if(a[u][v]==1)
            {
                indegree[v]--;
                if(indegree[v]==0)
                s[++top]=v;
            }
        }
    }
    for(i=0;i<n;i++)
    {
        printf("%d\t",t[i]);
    }
}
```

Krushkal

```c
#include<stdio.h>
int ne=1,cost[10][10];
void main()
{
    int i,j,parent[10],u,v,n,a,b,min,mincost=0;
    printf("enter the no of vertices\n");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
    parent[i]=0;
    }
    printf("enter the cost matrix");

    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);

        }
    }
    while (ne<=n)
    {
        min=999;
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                if(cost[i][j]<min)
                {
                    min=cost[i][j];
                    a=u=i;
                    b=v=j;

                }
            }
        }
        while(parent[u])
        {
            u=parent[u];

        }
        while(parent[v])
        {
            v=parent[v];
        }
        if(u!=v)
        {
            printf("edge %d(%d->%d)mincost is%d\n",ne++,u,v,min);
            mincost=mincost+min;
            cost[a][b]=cost[b][a]=999;
        }
    }
    printf("minimumcost is%d",mincost);
}
```

Prims

```
#include<stdio.h>
void main()
{
    int i,j,ne,n,source,target,min,mincost=0;
    int cost[10][10],visited[10],parent[10];
    printf("enter the number of vertices");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        parent[i]=0;

    }
    printf("enter the cost matrix");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&cost[i][j]);
        }
    }
    for(i=0;i<n;i++)
    {
        visited[i]=0;
    }
        printf("enter the source vertex");
        scanf("%d",&source);
            visited[source]=1;
            ne=1;
            while(ne<n)
            {
                min=999;
                for(i=0;i<n;i++)
                {
                    if(visited[i])
                    {
                        for(j=0;j<n;j++)
                        {
                            if(cost[i][j]<min && !visited[j])
                            {
                                min=cost[i][j];
                                source=i;
                                target=j;
                            }
                        }
                    }
                }
                visited[target]=1;
                printf("\n edge %d(%d->%d)%d",ne++,source,target,min);
            mincost=mincost+min;
        }
        printf("mincost of spanning tree %d \n",mincost);
 }
```

Floyds

```
#include<stdio.h>
void main()
{
    int i,j,k,cost[10][10],n;
    printf("enter the number of vertices");
    scanf("%d",&n);
    printf("enter the cost matric");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
        }
    }
    for(k=1;k<=n;k++)
    {
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                if((cost[i][k]+cost[k][j])<cost[i][j])
                cost[i][j]=cost[i][k]+cost[k][j];
            }
        }
    }
    printf("the edge with cost\n");
```

```c
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            printf("edge %d to %d ,cost is=%d\n",i,j,cost[i][j]);
        }
    }
}
```

Warshal

```c
#include<stdio.h>
void main()
{
    int a[10][10],T[10][10];
    int n,i,j,k;
    printf("enter the number of vertices");
    scanf("%d",&n);
    printf("enter the adjacency matix\n");    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    for(k=0;k<n;k++)
    {
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
                a[i][j]=a[i][j]||(a[i][k] && a[k][j]);
            }
        }
    }
                printf("Transitive closure is \n");
                for(i=0;i<=n;i++)
                {
                    for(j=0;j<=n;j++)
                    {
                        printf("%d",a[i][j]);
                        printf("\n");
                    }
                }
}
```

Selection

```
#include<stdio.h>
int a[10],n,i,j,min,temp;
void main()
{
    printf("enter the size of an array");
    scanf("%d",&n);
    printf("enter the elements of array");
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);
    printf("array elements are");
    for(i=0;i<n;i++)
    printf("%d",a[i]);
    selectionsort();
    printf("elements after sorting");
    for(i=0;i<n;i++)
    printf("%d",a[i]);
}
void selectionsort()
{
    for(i=0;i<n;i++)
    {
        min=i;
        for(j=0;j<n;j++)
        {
            if(a[j]>a[min])
            {
                min=j;
            }

        temp=a[min];
        a[min]=a[i];
        a[i]=temp;
        }
    }
}
```

Merge

```
#include<stdio.h>
int a[10],n,low,mid,high,i;
void mergesort(int low,int high)
{
```

```c
    if(low!=high)
    {
        mid=(low+high)/2;
        mergesort(low,mid);
        mergesort(mid+1,high);
        merge(low,mid,high);
    }
}
void merge(int low,int mid,int high)
{
    int i,j,k,temp[10];
    i=low;
    j=mid+1;
    k=low;
    while(i<=mid && j<=high)
    {
        if(a[i]<=a[j])
        temp[k++]=a[i++];
        else
        temp[k++]=a[j++];

    }
    while(i<=mid)
    {
        temp[k++]=a[i++];
    }
    while(j<=high)
    {
        temp[k++]=a[j++];
    }
    for(i=low;i<=high;i++)
    {
        a[i]=temp[i];
    }

}
void main()
{
    printf("enter the size of an array");
    scanf("%d",&n);
    printf("enter the elements of array");
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);
    mergesort(0,n-1);
    printf("array elements after sorting");
    for(i=0;i<n;i++)
    printf("%d",a[i]);
}
```

Quick

```c
#include <stdio.h>
void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition(int arr[], int low, int high)
{
    int pivot = arr[low];
    int i = low;
    int j = high;
    while (i < j)
    {
        while (arr[i] <= pivot && i <= high - 1)
        {
            i++;
        }
        while (arr[j] > pivot && j >= low + 1)
        {
            j--;
        }
        if (i < j)
        {
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[low], &arr[j]);
    return j;
}
void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
    int partitionIndex = partition(arr, low, high);
    quickSort(arr, low, partitionIndex - 1);

    quickSort(arr, partitionIndex + 1, high);
    }
}
int main()
{
    int n,arr10],i;
    printf("enter the size of an array \n");
    scanf("%d",&n);
    printf("enter the lements are an array\n");
    for(i=0;i<n;i++)
    scanf("%d",&arr[i]);
    printf("Original array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    quickSort(arr, 0, n - 1);
    printf("\nSorted array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```