BHAVANA CK

1BM0CS403

CSE-4A

PROGRAM 10:

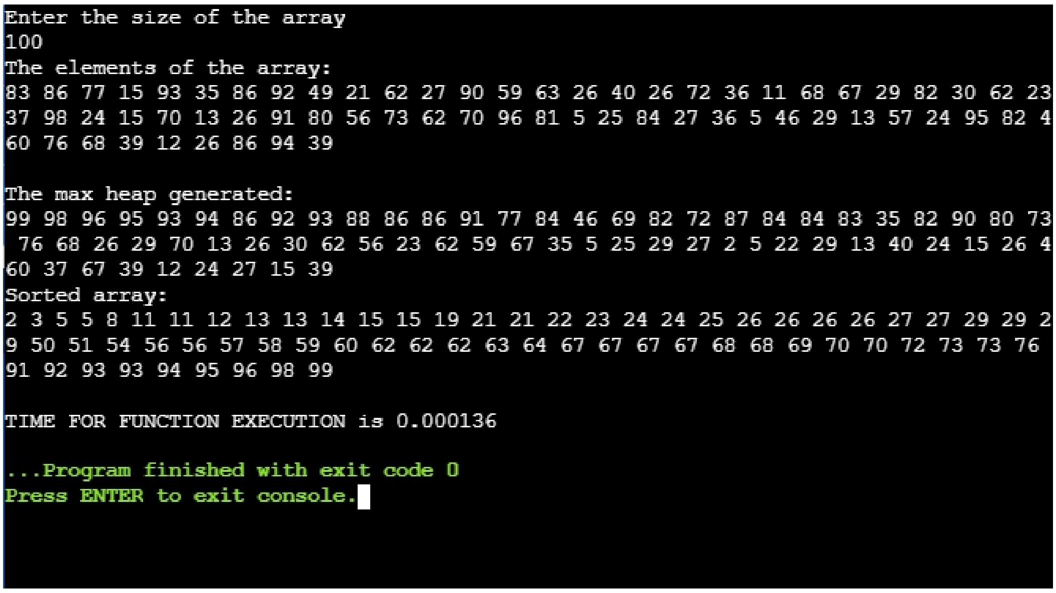
Sort a given set of N integer elements using Heap Sort technique and compute its time taken

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
void swap(int *a, int *b) {
        int temp = *a;
        *a = *b;
        *b = temp;
    }
    void printArray(int arr[], int n) {
        for (int i = 0; i < n; ++i)
            printf("%d ", arr[i]);
        printf("\n");
    }
    void heapify(int arr[], int n, int i)
    {
        int largest = i;
        int left = 2 * i + 1;
        int right = 2 * i + 2;

        if (left < n && arr[left] > arr[largest])
            largest = left;
```

```
      if (right < n && arr[right] > arr[largest])
          largest = right;

      if (largest != i) {
          swap(&arr[i], &arr[largest]);
          heapify(arr, n, largest);
      }
  }

  void heapSort(int arr[], int n) {
      for (int i = n / 2 - 1; i >= 0; i--)
      {
                heapify(arr, n, i);
      }
      printf("The max heap generated:\n");
      printArray(arr, n);
      for (int i = n - 1; i >= 0; i--) {
          swap(&arr[0], &arr[i]);

          heapify(arr, i, 0);
      }
  }

int main()
{
      int n;
   clock_t start, end;
      double cpu_time_used;
   printf("Enter the size of the array\n");
   scanf("%d",&n);
```

```
        int arr[n];
        printf("The elements of the array:\n");
        for(int i=0;i<n;i++)
        arr[i]=rand()%100;
        printArray(arr, n);
        printf("\n");
        start = clock();
        heapSort(arr, n);
        end = clock();
        cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
        printf("Sorted array: \n");
        printArray(arr, n);
        printf("\n");
        printf("TIME FOR FUNCTION EXECUTION is %f",
cpu_time_used);
        return 0;

    }
```

OUTPUT:



```
Enter the size of the array
100
The elements of the array:
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23
37 98 24 15 70 13 26 91 80 56 73 62 70 96 81 5 25 84 27 36 5 46 29 13 57 24 95 82 4
60 76 68 39 12 26 86 94 39

The max heap generated:
99 98 96 95 93 94 86 92 93 88 86 86 91 77 84 46 69 82 72 87 84 84 83 35 82 90 80 73
 76 68 26 29 70 13 26 30 62 56 23 62 59 67 35 5 25 29 27 2 5 22 29 13 40 24 15 26 4
60 37 67 39 12 24 27 15 39
Sorted array:
2 3 5 5 8 11 11 12 13 13 14 15 15 19 21 21 22 23 24 24 25 26 26 26 26 27 27 29 29 2
9 50 51 54 56 56 57 58 59 60 62 62 62 63 64 67 67 67 67 68 68 69 70 70 72 73 73 76
91 92 93 93 94 95 96 98 99

TIME FOR FUNCTION EXECUTION is 0.000136

...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 11:

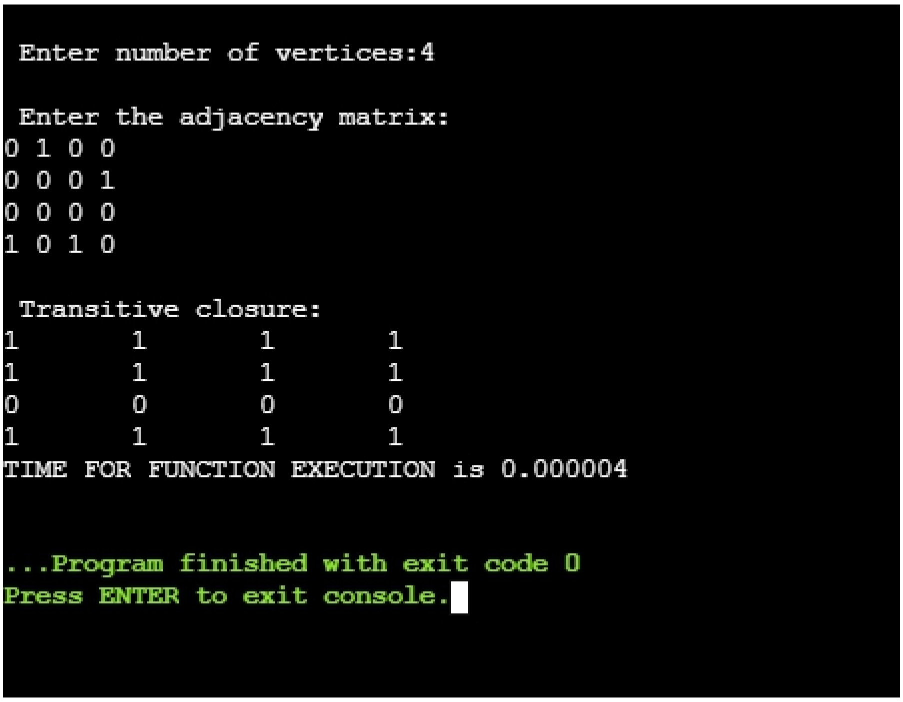
Implement Warshall’s algorithm using dynamic programming.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include <time.h>
int a[20][20];
int max(int,int);
void warshal(int p[20][20],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 i,j,n;
        clock_t start, end;
        double cpu_time_used;
        printf("\n Enter number of vertices:");
```

```
       scanf("%d",&n);

       for(i=1;i<=n;i++)
       {
            for(j=1;j<=n;j++)
            {
                  a[i][j]=0;
            }
       }

       printf("\n Enter the adjacency matrix:\n");
       for(i=1;i<=n;i++)
       {
              for(j=1;j<=n;j++)
              {
              scanf("%d",&a[i][j]);
              }
       }

       start = clock();
       warshal(a,n);
       end = clock();

    printf("\n Transitive closure: \n");
    for (i=1;i<=n;i++) {
        for (j=1;j<=n;j++)
             printf("%d\t",a[i][j]);
        printf("\n");
    }
    cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
```

```
        printf("TIME FOR FUNCTION EXECUTION is %f\n",
cpu_time_used);
        getch();
}
```

OUTPUT:



```
 Enter number of vertices:4

 Enter the adjacency matrix:
0 1 0 0
0 0 0 1
0 0 0 0
1 0 1 0

 Transitive closure:
1       1       1       1
1       1       1       1
0       0       0       0
1       1       1       1
TIME FOR FUNCTION EXECUTION is 0.000004


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 12:

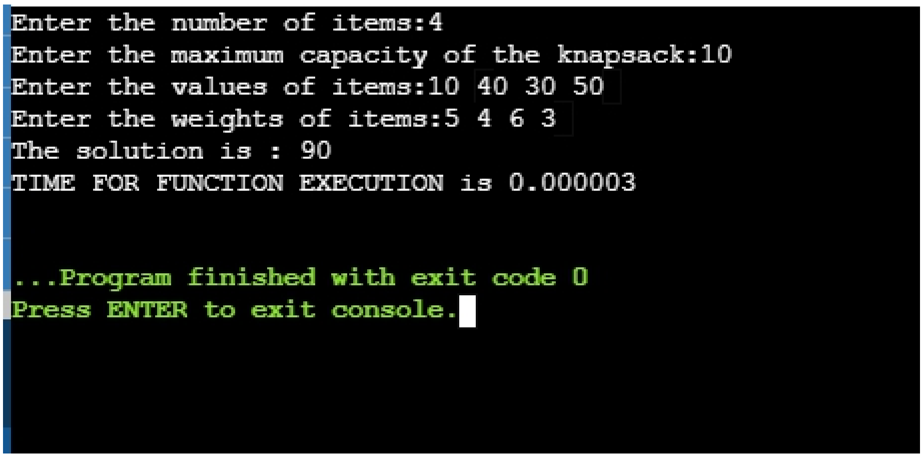
Implement 0/1 Knapsack problem using dynamic programming.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include <time.h>
int max(int a, int b) {
	if(a>b){
		return a;
	} else {
		return b;
	}
}
int knapsack(int W, int wt[], int val[], int n) {
	int i, w;
	int knap[n+1][W+1];
	for (i = 0; i <= n; i++) {
		for (w = 0; w <= W; w++) {
			if (i==0 || w==0)
				knap[i][w] = 0;
			else if (wt[i-1] <= w)
				knap[i][w] = max(val[i-1] + 
knap[i-1][w-wt[i-1]], knap[i-1][w]);
			else
				knap[i][w] = knap[i-1][w];
		}
	}
```

```
    return knap[n][W];
}
int main()
{
    int W;
    int n;
    clock_t start, end;
    double cpu_time_used;
    printf("Enter the number of items:");
    scanf("%d",&n);
    int val[n];
    int wt[n];
    printf("Enter the maximum capacity of the
knapsack:");
    scanf("%d",&W);
    printf("Enter the values of items:");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&val[i]);
    }
    printf("Enter the weights of items:");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&wt[i]);
    }
    start = clock();
    int sol=knapsack(W, wt, val, n);
    end = clock();
    printf("The solution is : %d\n", sol);
    cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
```

```
	printf("TIME FOR FUNCTION EXECUTION is %f\n",
cpu_time_used);
	getch();
	return 0;
}
```

OUTPUT:



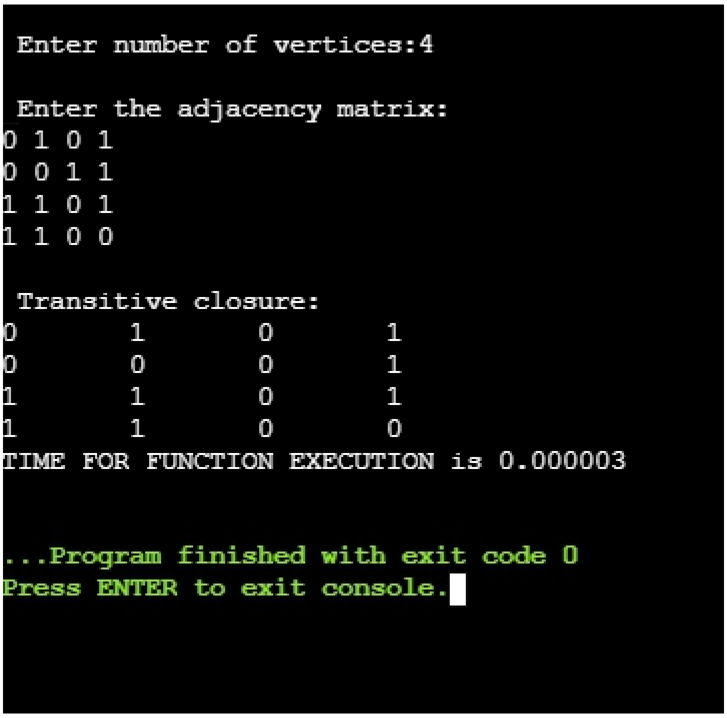
```
Enter the number of items:4
Enter the maximum capacity of the knapsack:10
Enter the values of items:10 40 30 50
Enter the weights of items:5 4 6 3
The solution is : 90
TIME FOR FUNCTION EXECUTION is 0.000003


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 13:

Implement All Pair Shortest paths problem using Floyd's algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include <time.h>
double inf=INFINITY;
int a[20][20];
int min(int,int);
void warshal(int p[20][20],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 i,j,n;
        clock_t start, end;
        double cpu_time_used;
        printf("\n Enter number of vertices:");
        scanf("%d",&n);

        printf("\n Enter the adjacency matrix:\n");
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                scanf("%d",&a[i][j]);
                }
        }

        start = clock();
        warshal(a,n);
        end = clock();

     printf("\n Transitive closure: \n");
     for (i=1;i<=n;i++) {
          for (j=1;j<=n;j++)
                printf("%d\t",a[i][j]);
          printf("\n");
     }
    cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
        printf("TIME FOR FUNCTION EXECUTION is %f\n",
cpu_time_used);
        getch();
}
```

OUTPUT:



```
 Enter number of vertices:4

 Enter the adjacency matrix:
0 1 0 1
0 0 1 1
1 1 0 1
1 1 0 0

 Transitive closure:
0       1       0       1
0       0       0       1
1       1       0       1
1       1       0       0
TIME FOR FUNCTION EXECUTION is 0.000003


...Program finished with exit code 0
Press ENTER to exit console.
```

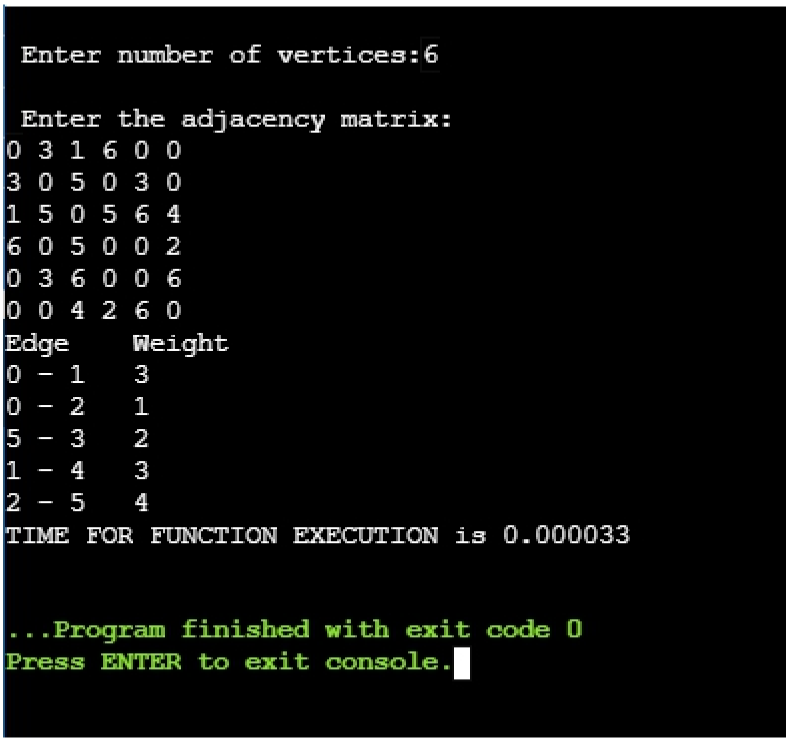
PROGRAM 14:

Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.

```
#include<stdio.h>
#include<conio.h>
#include <limits.h>
#include <time.h>
int a[20][20];
void printMST(int parent[],int n)
{
        printf("Edge \tWeight\n");
        for (int i = 1; i < n; i++)
                printf("%d - %d \t%d \n", parent[i], i,
a[i][parent[i]]);
}
int findMinVertex(int visited[],int weight[],int n)
{
        int minVertex = -1; // Initialized to -1 means there is
no vertex till now
          for (int i = 0; i < n; i++)
          {
              if (!visited[i] && (minVertex == -1 || weight[i] <
weight[minVertex]))
              {
              minVertex = i;
              }
          }
          return minVertex;
}
```

```
void prim(int n)
{
    int parent[n];
    int weight[n];
    int visited[n];
    for(int i=0;i<n;i++)
    {
        visited[i]=0;
        weight[i]=INT_MAX;
    }
    weight[0]=0;
    parent[0]=-1;
    for(int count=0;count<n-1;count++)
    {
        int
minVertex=findMinVertex(visited,weight,n);
        visited[minVertex]=1;
            for (int j = 0; j < n; j++)
            {
                if(a[minVertex][j] != 0 && !visited[j])
                {
                     if(a[minVertex][j] < weight[j])
                     {
                     // updating weight array and
parent array
                     weight[j] = a[minVertex][j];
                     parent[j] = minVertex;
                     }
                }
            }
    }
    printMST(parent,n);
}

void main() {
    int i,j,n;
    clock_t start, end;
    double cpu_time_used;
    printf("\n Enter number of vertices:");
    scanf("%d",&n);

    printf("\n Enter the adjacency matrix:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
        scanf("%d",&a[i][j]);
        }
    }

    start = clock();
    prim(n);
    end = clock();

cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
    printf("TIME FOR FUNCTION EXECUTION is %f\n",
cpu_time_used);
    getch();
}
```

OUTPUT:



```
 Enter number of vertices:6

 Enter the adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0
Edge    Weight
0 - 1   3
0 - 2   1
5 - 3   2
1 - 4   3
2 - 5   4
TIME FOR FUNCTION EXECUTION is 0.000033


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 15:

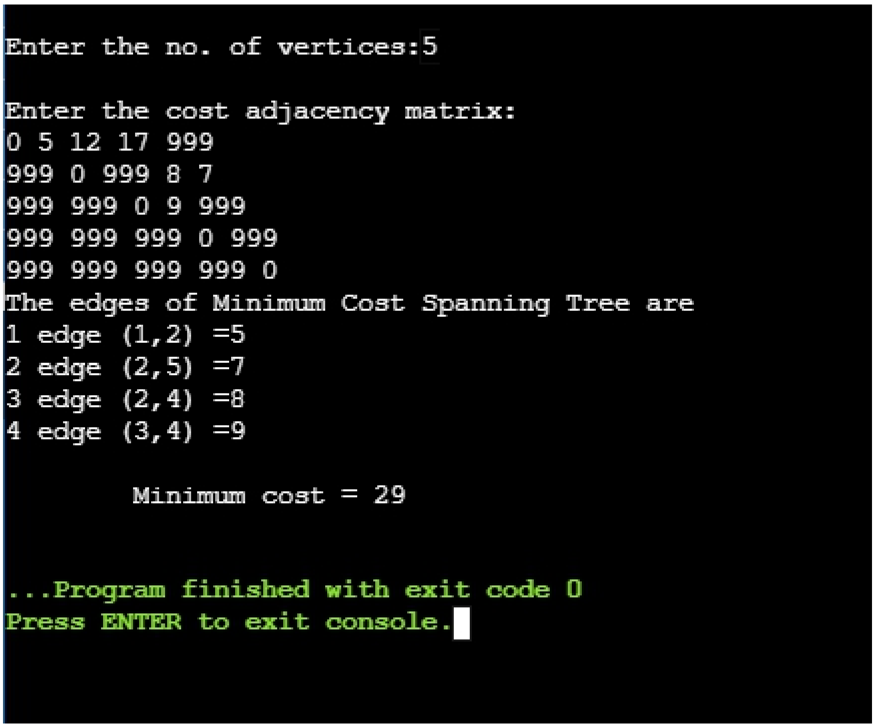
Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.

```
#include <stdio.h>
#include <stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int i)
    {
     while(parent[i])
     i=parent[i];
     return i;
    }
  int uni(int i,int j)
    {
     if(i!=j)
     {
     parent[j]=i;
     return 1;
     }
     return 0;
    }
    void main()
    {
     printf("\nEnter the no. of vertices:");
     scanf("%d",&n);
     printf("\nEnter the cost adjacency matrix:\n");
     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("The edges of Minimum Cost Spanning Tree
are\n");
	while(ne < n)
	{
	for(i=1,min=999;i<=n;i++)
	{
	for(j=1;j <= n;j++)
	{
	if(cost[i][j] < min)
	{
	min=cost[i][j];
	a=u=i;
	b=v=j;
	}
	}
	}
	u=find(u);
	v=find(v);
	if(uni(u,v))
	{
	printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);
	mincost +=min;
	}
	cost[a][b]=cost[b][a]=999;
```

```
    }
    printf("\n\tMinimum cost = %d\n",mincost);

  }
```

OUTPUT:



```
Enter the no. of vertices:5

Enter the cost adjacency matrix:
0 5 12 17 999
999 0 999 8 7
999 999 0 9 999
999 999 999 0 999
999 999 999 999 0
The edges of Minimum Cost Spanning Tree are
1 edge (1,2) =5
2 edge (2,5) =7
3 edge (2,4) =8
4 edge (3,4) =9

        Minimum cost = 29


...Program finished with exit code 0
Press ENTER to exit console.
```

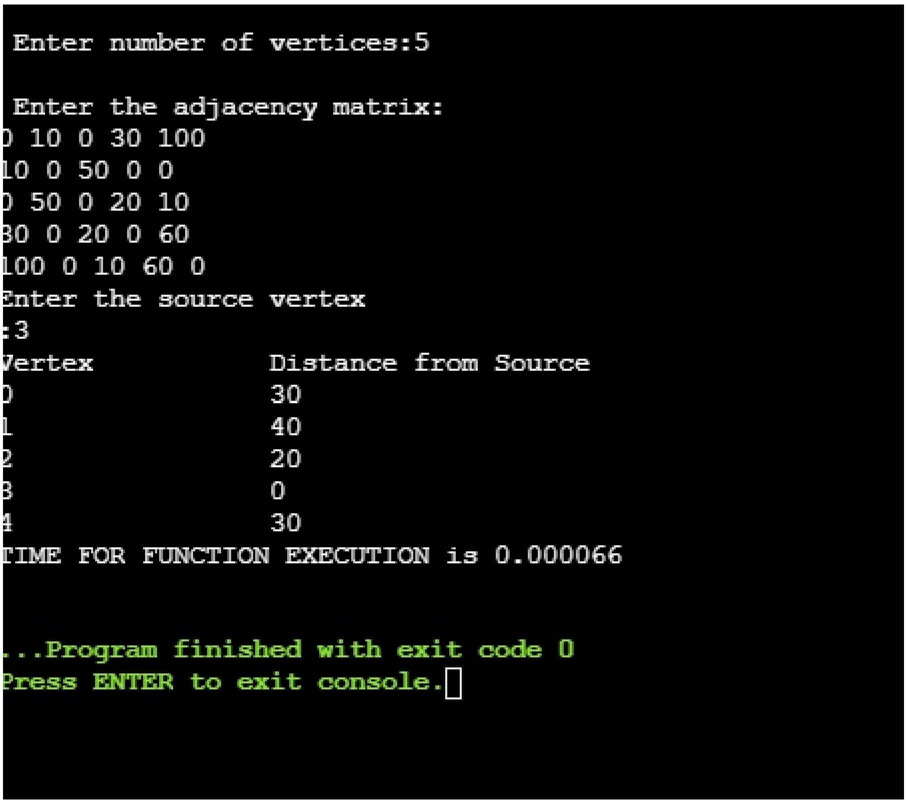
PROGRAM 16:

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm.

```
#include<stdio.h>
#include<conio.h>
#include <limits.h>
#include <time.h>
int a[20][20];
void printSolution(int dist[],int n)
{
        printf("Vertex \t\t Distance from Source\n");
        for (int i = 0; i < n; i++)
                printf("%d \t\t %d\n", i, dist[i]);
}
int findMinVertex(int visited[],int dist[],int n)
{
        int minVertex = -1;
          for (int i = 0; i < n; i++)
          {
              if (!visited[i] && (minVertex == -1 || dist[i] <
dist[minVertex]))
              {
                    minVertex = i;
              }
          }
          return minVertex;
}

void dijsktra(int n,int src)
{
        int visited[n];
        int dist[n];
        for(int i=0;i<n;i++)
        {
                visited[i]=0;
                dist[i]=INT_MAX;
        }
        dist[src]=0;
        for(int i=0;i<n-1;i++)
        {
                int minVertex=findMinVertex(visited,dist,n);
                visited[minVertex]=1;
                for(int j=0;j<n;j++)
                {
                        if(a[minVertex][j]!=0 && !visited[j])
                        {

if(a[minVertex][j]+dist[minVertex]<dist[j])
                                {

dist[j]=a[minVertex][j]+dist[minVertex];
                                }
                        }
                }
        }
        printSolution(dist,n);
}
```

```
void main() {
        int i,j,n;
        clock_t start, end;
        double cpu_time_used;
        int src;
        printf("\n Enter number of vertices:");
        scanf("%d",&n);

        printf("\n Enter the adjacency matrix:\n");
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
                scanf("%d",&a[i][j]);
                }
        }
        printf("Enter the source vertex\n:");
        scanf("%d",&src);
        start = clock();
        dijsktra(n,src);
        end = clock();

    cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
        printf("TIME FOR FUNCTION EXECUTION is %f\n",
cpu_time_used);
        getch();
}
```

OUTPUT:



```
 Enter number of vertices:5

 Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0
Enter the source vertex
:3
Vertex           Distance from Source
0                30
1                40
2                20
3                0
4                30
TIME FOR FUNCTION EXECUTION is 0.000066


...Program finished with exit code 0
Press ENTER to exit console.
```

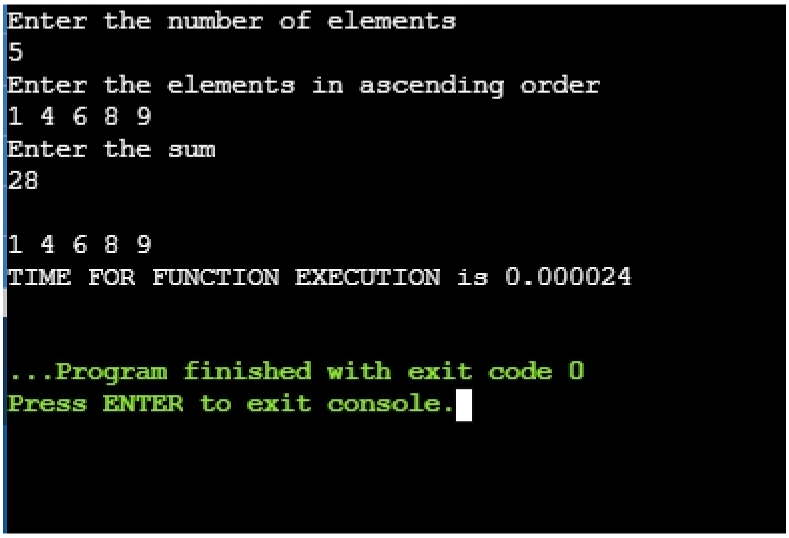
PROGRAM 17:

Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set S = {s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t have a solution.

```
#include<stdio.h>
#include<conio.h>
#include <limits.h>
#include <time.h>
int w[20],x[20],m,flag=0;;

void sum_of_subsets(int s,int k,int r)
{
    x[k]=1;
    if(s+w[k]==m)
    {
        for(int i=0;i<=k;i++)
        {
            if(x[i]==1)
            printf("%d ",w[i]);
            flag=1;
        }
        printf("\n");
    }
    else if(s+w[k]+w[k+1]<=m)
    sum_of_subsets(s+w[k],k+1,r-w[k]);
    if((s+r-w[k]>=m)&&(s+w[k+1]<=m))
    {
        x[k]=0;
        sum_of_subsets(s,k+1,r-w[k]);
    }
}


int main()
{
    int i,n,r=0;
    clock_t start, end;
       double cpu_time_used;
    printf("Enter the number of elements\n");
    scanf("%d",&n);
    printf("Enter the elements in ascending order\n");
    for(i=0;i<n;i++){
    scanf("%d",&w[i]);
    r=r+w[i];}
    printf("Enter the sum\n");
    scanf("%d",&m);
    printf("\n");
    start=clock();
    sum_of_subsets(0,0,r);
    end=clock();
    if(flag==0)
    printf("No solution\n");
    cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
```

```
        printf("TIME FOR FUNCTION EXECUTION is %f\n", 
cpu_time_used);
        getch();
    return 0;
}
```

OUTPUT:



```
Enter the number of elements
5
Enter the elements in ascending order
1 4 6 8 9
Enter the sum
28

1 4 6 8 9
TIME FOR FUNCTION EXECUTION is 0.000024


...Program finished with exit code 0
Press ENTER to exit console.
```

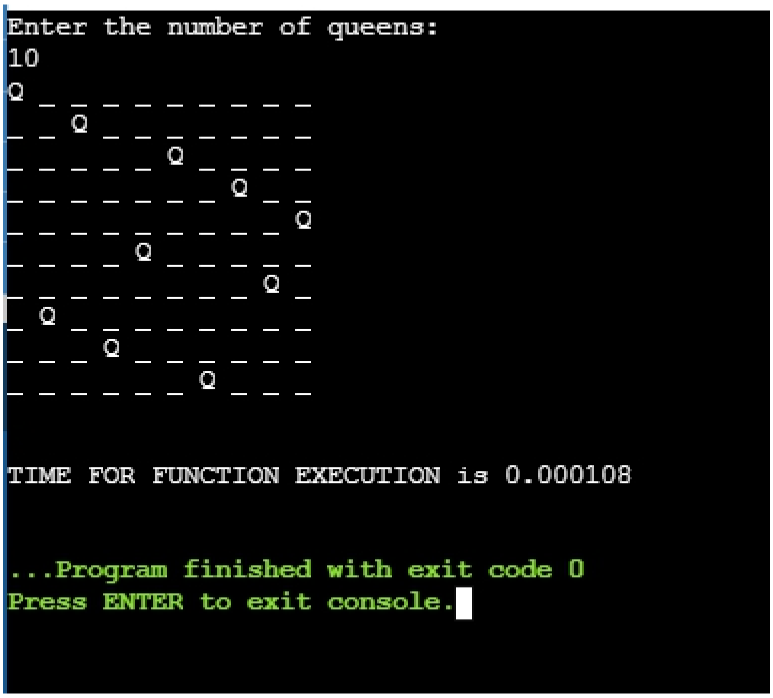
PROGRAM 18:

Implement “N-Queens Problem” using Backtracking

```
#include<stdio.h>
#include<conio.h>
#include <limits.h>
#include <time.h>
int board[10][10];
int isSafe(int board[][10],int i,int j,int n)
{
	for(int row=0;row<i;row++)
	{
		if(board[row][j]==1)
		{
			return 0;
		}
	}

	int x=i;
	int y=j;
	while(x>=0 && y>=0)
	{
		if(board[x][y]==1)
		{
			return 0;
		}
		x--;
		y--;
	}
```

```
        x=i;
        y=j;
        while(x>=0 && y<n)
        {
                if(board[x][y]==1)
                {
                        return 0;
                }
                x--;
                y++;
        }
        return 1;
}
int solveNQueen(int board[][10],int i, int n)
{
        if(i==n)
        {
                for(int i=0;i<n;i++)
                {
                        for(int j=0;j<n;j++)
                        {
                                if(board[i][j]==1)
                                {
                                        printf("Q ");
                                }
                                else
                                {
                                        printf("_ ");
                                }
                        }
                        printf("\n");
                }
                printf("\n\n");
                return 1;
        }
        for(int j=0;j<n;j++)
        {
                if(isSafe(board,i,j,n))
                {
                        board[i][j]=1;
                        int nextQueenPossible=
solveNQueen(board,i+1,n);
                        if(nextQueenPossible==1)
                        {
                            return 1;
                        }
                        board[i][j]=0;
                }

        }
        return 0;
}
int main()
{
        int n;
        clock_t start, end;
        double cpu_time_used;
        printf("Enter the number of queens:\n");
        scanf("%d",&n);
        for(int i=0;i<n;i++)
        {
                for(int j=0;j<n;j++)
                {
                        board[i][j]=0;
                }
        }
        start=clock();
        int val=solveNQueen(board,0,n);
        end=clock();
        cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
        printf("TIME FOR FUNCTION EXECUTION is %f\n",
cpu_time_used);
        getch();
        return 0;
}
```

OUTPUT:



```
Enter the number of queens:
10
Q _ _ _ _ _ _ _ _ _
_ _ Q _ _ _ _ _ _ _
_ _ _ _ _ Q _ _ _ _
_ _ _ _ _ _ _ Q _ _
_ _ _ _ _ _ _ _ _ Q
_ _ _ _ Q _ _ _ _ _
_ _ _ _ _ _ _ _ Q _
_ Q _ _ _ _ _ _ _ _
_ _ _ Q _ _ _ _ _ _
_ _ _ _ _ _ Q _ _ _


TIME FOR FUNCTION EXECUTION is 0.000108


...Program finished with exit code 0
Press ENTER to exit console.
```