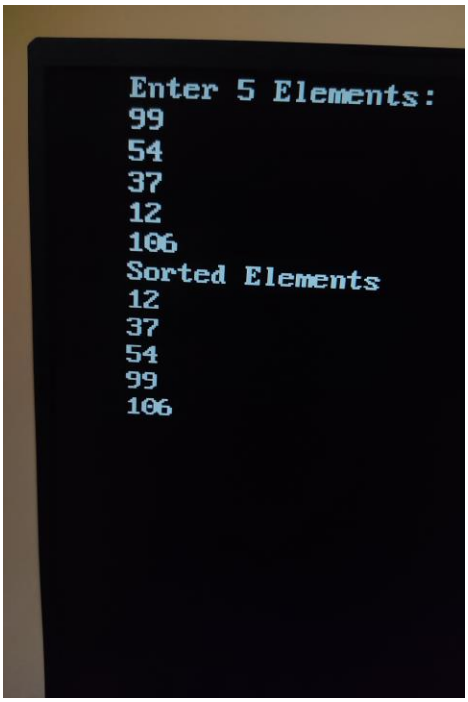
DAA Short prog

1. Write a program to sort a list of N elements using Selection Sort Technique.

```c
#include<stdio.h>
#include<conio.h>

void main() {
    int arr[5], i, j, pos, temp;
    clrscr();

    printf("Enter 5 elements:\n");
    for(i = 0; i < 5; i++)
        scanf("%d\n", &arr[i]);

    for(i = 0; i < 4; i++) {
        pos = i;
        for(j = i + 1; j < 5; j++) {
            if(arr[j] < arr[pos])
                pos = j;
        }
        if(pos != i) {
            temp = arr[i];
            arr[i] = arr[pos];
            arr[pos] = temp;

        }
    }

    printf("Sorted elements:\n");
    for(i = 0; i < 5; i++)
        printf("%d\n", arr[i]);

    getch();
}
```

OUTPUT;



2 .Write a program to perform Travelling Sales man Problem.

```
#include<stdio.h>
#include<conio.h>
int tsp_g[10][10] = {
    {0,30,33,10,45},
    {56,0,9,15,18},
    {29,13,0,5,12},
    {33,28,16,0,3},
    {1,4,30,24,0}
};
int visited[10],n=6,cost = 0;
void tsp(int c) {
   int k,next = 999, min = 999;
   visited[c] = 1;
   printf("%d\t",c + 1);

   for(k = 0; k <n; k++) {
      if((tsp_g[c][k]!=0) && (visited[k]==0))
      {
     if(tsp_g[c][k] < min) {
    min = tsp_g[c][k];
    }
    next = k;
```

```
      }
   }
   if(min != 999) {
      cost=cost + min;
      tsp(next);
   } else {
       printf("1");
       cost += tsp_g[c][0];
   }
}

void main() {
    clrscr();
    printf("\nshortest path:\t");
    tsp(0);
    printf("\nMinimum Cost: %d\t", cost);
    getch();
    }
```

OUTPUT;

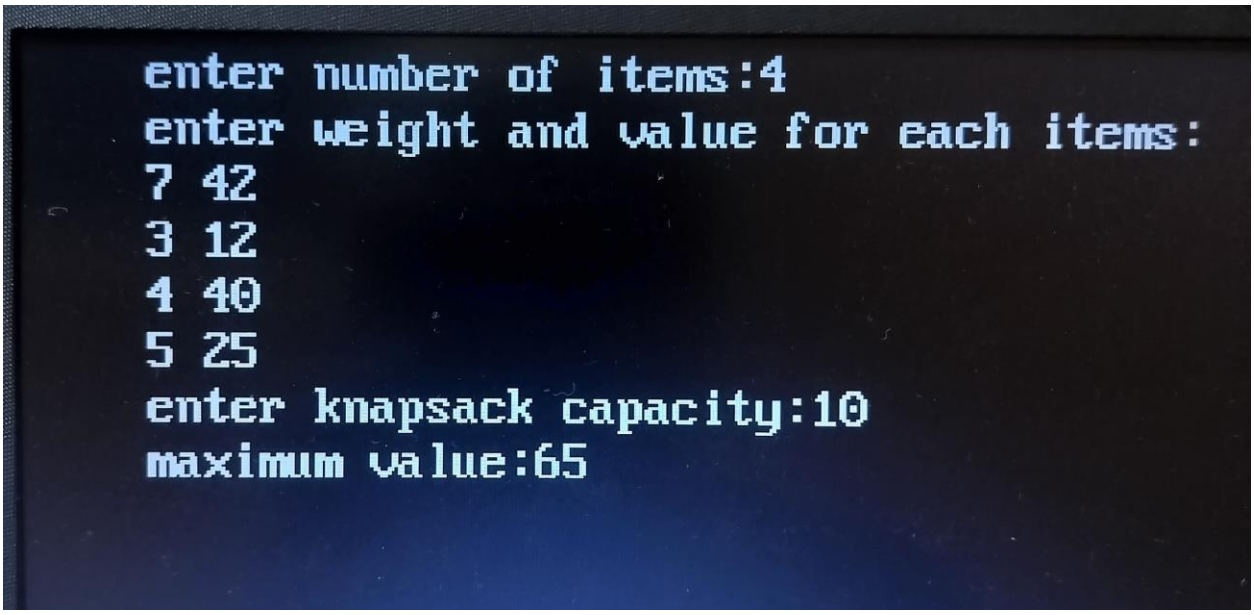
```
shortest path:  1       5       4       3       2       1
Minimum Cost: 99
```

3. Write program to implement Dynamic Programming algorithm for the 0/1 Knapsack problem.

```
#include<stdio.h>
#include<conio.h>

int max(int a, int b) { return (a > b) ? a : b; }

int knapSack(int W, int wt[], int val[], int n) {
    if (n == 0 || W == 0) return 0;
    if (wt[n - 1] > W)
        return knapSack(W, wt, val, n - 1);
    return max(val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),
               knapSack(W, wt, val, n - 1));
}

void main() {
```

```
    int n, W, wt[10], val[10], i;
    clrscr();

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

    printf("Enter WEIGHT and VALUE for each item:\n");
    for(i = 0; i < n; i++)
        scanf("%d %d", &wt[i], &val[i]);

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

    printf("Maximum value: %d", knapSack(W, wt, val, n));
    getch();
}
```

OUTPUT:



4. Write program to implement the DFS and BFS algorithm for a graph.

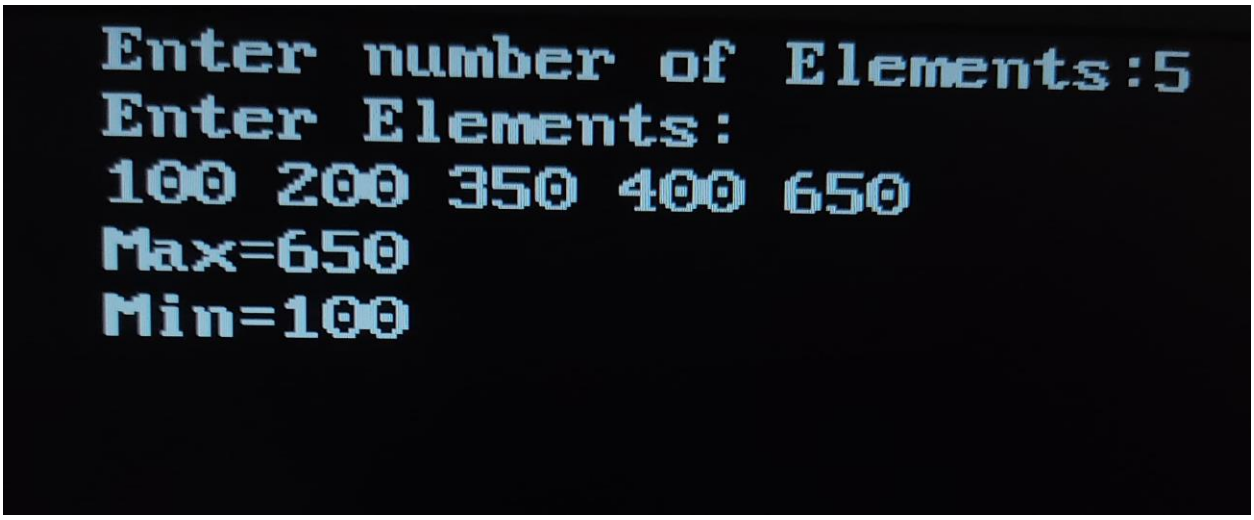
OUTPUT:

5 .Write a program to find minimum and maximum value in an array using divide and conquer.

```c
#include<stdio.h>
#include<conio.h>

void maxmin(int a[], int i, int j, int *max, int *min) {
    int mid, max1, min1;
    if (i == j) *max = *min = a[i];
    else if (i == j - 1) {
        *max = (a[i] > a[j]) ? a[i] : a[j];
        *min = (a[i] < a[j]) ? a[i] : a[j];
    } else {
        mid = (i + j) / 2;
        maxmin(a, i, mid, max, min);
        maxmin(a, mid + 1, j, &max1, &min1);
        *max = (*max > max1) ? *max : max1;
        *min = (*min < min1) ? *min : min1;
    }
}

void main() {
    int a[10], n, max, min, i;
    clrscr();

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

    printf("Enter elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    maxmin(a, 0, n - 1, &max, &min);
```

```
    printf("Max = %d\nMin = %d", max, min);
    getch();
}
```

OUTPUT:



6. Write a test program to implement Divide and Conquer Strategy. Eg: Quick sort algorithm for sorting list of integers in ascending order.

```
#include<stdio.h>
#include<conio.h>

int partition(int a[], int start, int end) {
    int pivot = a[end], i = start - 1, j, t;
    for (j = start; j < end; j++) {
        if (a[j] < pivot) {
            t = a[++i];
            a[i] = a[j];
            a[j] = t;
        }
    }
    t = a[i + 1];
    a[i + 1] = a[end];
    a[end] = t;
    return i + 1;
}
```

```
void quick(int a[], int start, int end) {
    if (start < end) {
        int p = partition(a, start, end);
        quick(a, start, p - 1);
        quick(a, p + 1, end);
    }
}

void main() {
    int a[] = {24, 9, 29, 14, 19, 27}, i, n = 6;
    clrscr();

    printf("\nBefore sorting:\n");
    for (i = 0; i < n; i++) printf("%d\t", a[i]);

    quick(a, 0, n - 1);

    printf("\nAfter sorting:\n ");
    for (i = 0; i < n; i++) printf("%d\t", a[i]);

    getch();
}
```

OUTPUT:

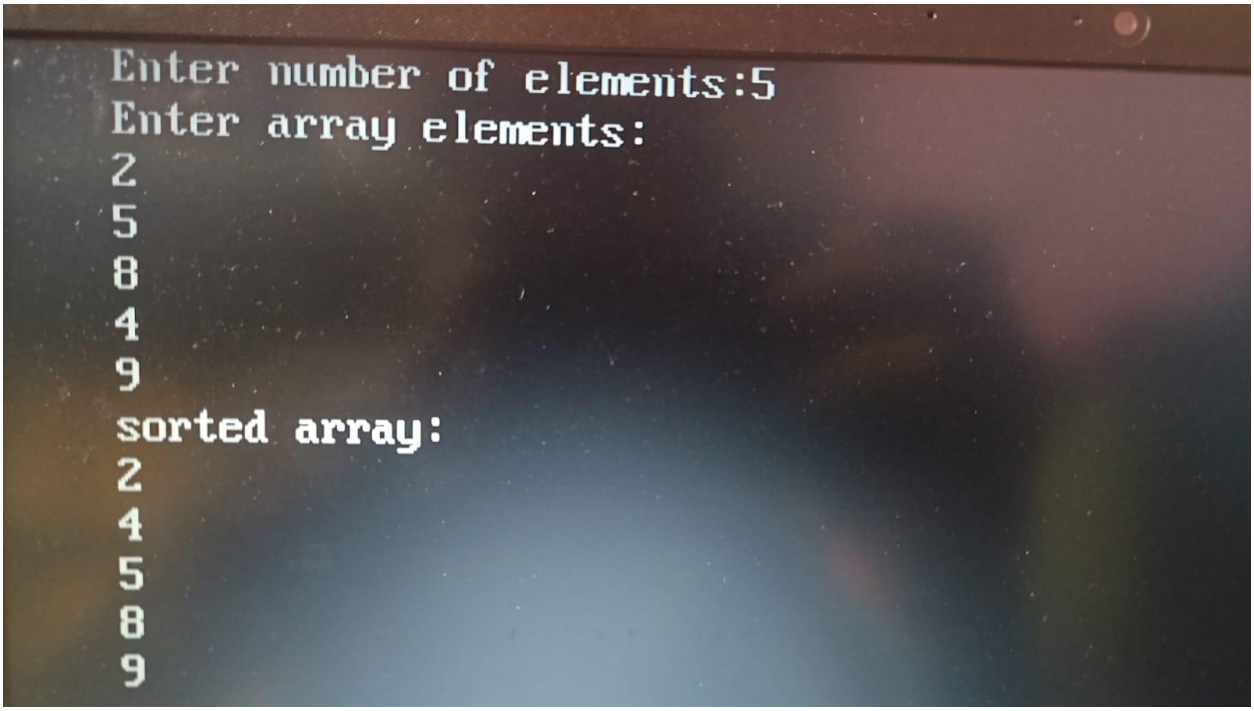
```
Before sorting:
24      9       29      14      19      27
After sorting:
9       14      19      24      27      29
```

7. Write a program to implement Merge sort algorithm for sorting a list of integers in ascending order.

```
#include<stdio.h>
#include<conio.h>
#define MAX 50
```

```c
void mergesort(int a[], int low, int high);
void merge(int a[], int low, int mid, int high);

void main() {
    int a[MAX], n, i;
    clrscr();
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter array elements:\n");
    for (i = 0; i < n; i++)
    scanf("%d", &a[i]);

    mergesort(a, 0, n - 1);

    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
    printf("%d\n", a[i]);
    getch();
}

void mergesort(int a[], int low, int high) {
    if (low < high) {
        int mid = (low + high) / 2;
        mergesort(a, low, mid);
        mergesort(a, mid + 1, high);
        merge(a, low, mid, high);
    }
}

void merge(int a[], int low, int mid, int high) {
    int i = low, j = mid + 1, k = low, temp[MAX];
    while (i <= mid && j <= high) temp[k++] = (a[i] < a[j]) ? a[i++] : a[j++];
    while (i <= mid) temp[k++] = a[i++];
    while (j <= high) temp[k++] = a[j++];
    for (i = low, i <= high; i++, k++)
    a[i] = temp[i];
}
```

OUTPUT:



8 .Write C program that accepts the vertices and edges for a graph and stores it as an adjacency matrix.

```c
#include <stdio.h>
#include <conio.h>

void createAdjMatrix(int Adj[10][10], int edges[][2], int M) {
    int i, x, y;
    for (i = 0; i < M; i++) {
        x = edges[i][0];
        y = edges[i][1];
        Adj[x][y] = Adj[y][x] = 1; // Add edges for both directions
    }
}

void printAdjMatrix(int Adj[10][10], int N) {
    int i, j;
    for (i = 1; i <= N; i++) {
        for (j = 1; j <= N; j++) {
            printf("%d ", Adj[i][j]);
        }
        printf("\n");
    }
```

```
}

void main() {
    int edges[][2] = { {1, 2}, {2, 3}, {4, 5}, {1, 5} };
    int Adj[10][10] = {0}; // Initialize adjacency matrix to 0
    int N = 5, M = 4;

    //clrscr();
    createAdjMatrix(Adj, edges, M);
    printAdjMatrix(Adj, N);
    //getch();
}
```

OUTPUT:

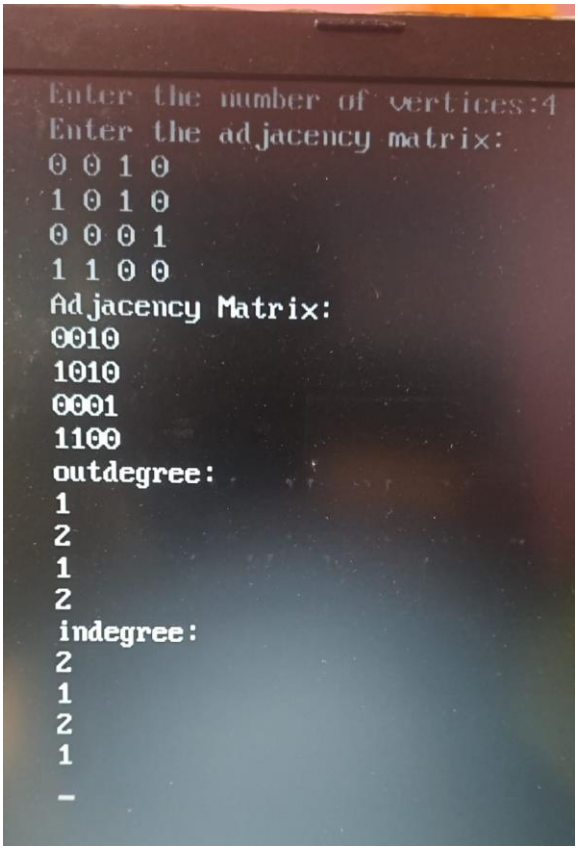
```
01001
10100
01000
00001
10010
```

9. Implement function to print In-Degree , Out Degree and to display that adjacency matrix.

```
#include <stdio.h>
#include <conio.h>

void main() {
    int n, i, j, adj[10][10], indegree[10] = {0}, outdegree[10] = {0};
    clrscr();

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

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

```
                indegree[j]++;
            }
        }
    }

    printf("Adjacency Matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", adj[i][j]);
        }
        printf("\n");
    }

    printf("\nOutdegree:\n");
    for (i = 0; i < n; i++) printf("%d\t", outdegree[i]);
    printf("\nIndegree:\n");
    for (i = 0; i < n; i++) printf("%d\t", indegree[i]);

    getch();
}
```

OUTPUT:

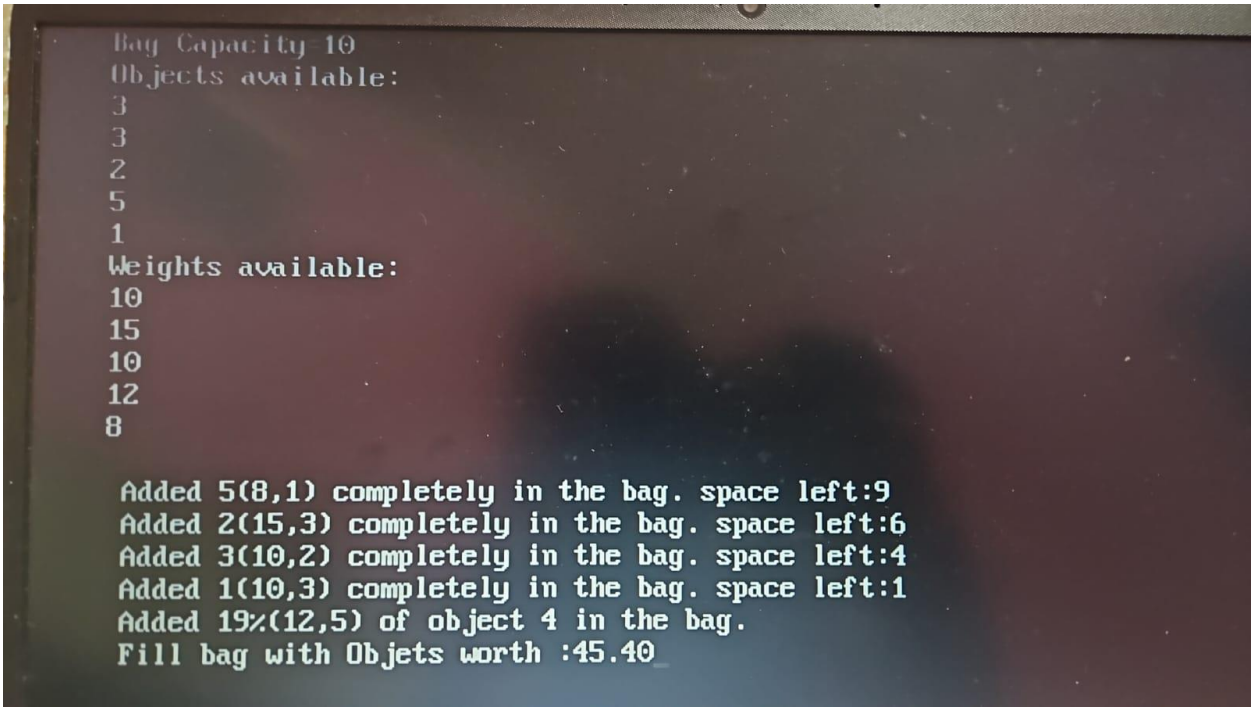


10. Write a program to perform Knapsack Problem using Greedy Solution.

```
#include<stdio.h>
#include<conio.h>
int n=5;
int p[10]={3,3,2,5,1};
int w[10]={10,15,10,12,8};
int W=10;
void main()
{
int cur_w,i,maxi,used[10]={0};
float tot_v=0;
clrscr();
printf("Bag Capacity=%d\n",W);
printf("Objects available:\n");
for(i=0; i<n; i++)
printf("%d\n",p[i]);
printf("Weights available:\n");
for(i=0; i<n; i++)
printf("%d\n",w[i]);
```

```
cur_w=W;
while(cur_w>0)
{
maxi=-1;
for(i=0;i<n;i++)
{
if(!used[i]&&(maxi==-1||(float)w[i]/p[i]>(float)w[maxi]/p[maxi]))
{maxi=i;
}
}
used[maxi]=1;
cur_w-=p[maxi];
tot_v +=w[maxi];
if(cur_w>=0)
{
printf("\n Added %d(%d,%d) completely in the bag. space
left:%d",maxi+1,w[maxi],p[maxi],cur_w);
}
else
{
printf("\n Added %d%%(%d,%d) of object %d in the
bag.",(int)((1+(float)cur_w/p[maxi])*100), w[maxi],p[maxi],maxi+1);
tot_v -=w[maxi];
tot_v +=(1 + (float)cur_w/p[maxi])*w[maxi];
}
}
printf("\n Fill bag with Objets worth :%.2f",tot_v);
getch();
}
```

OUTPUT:

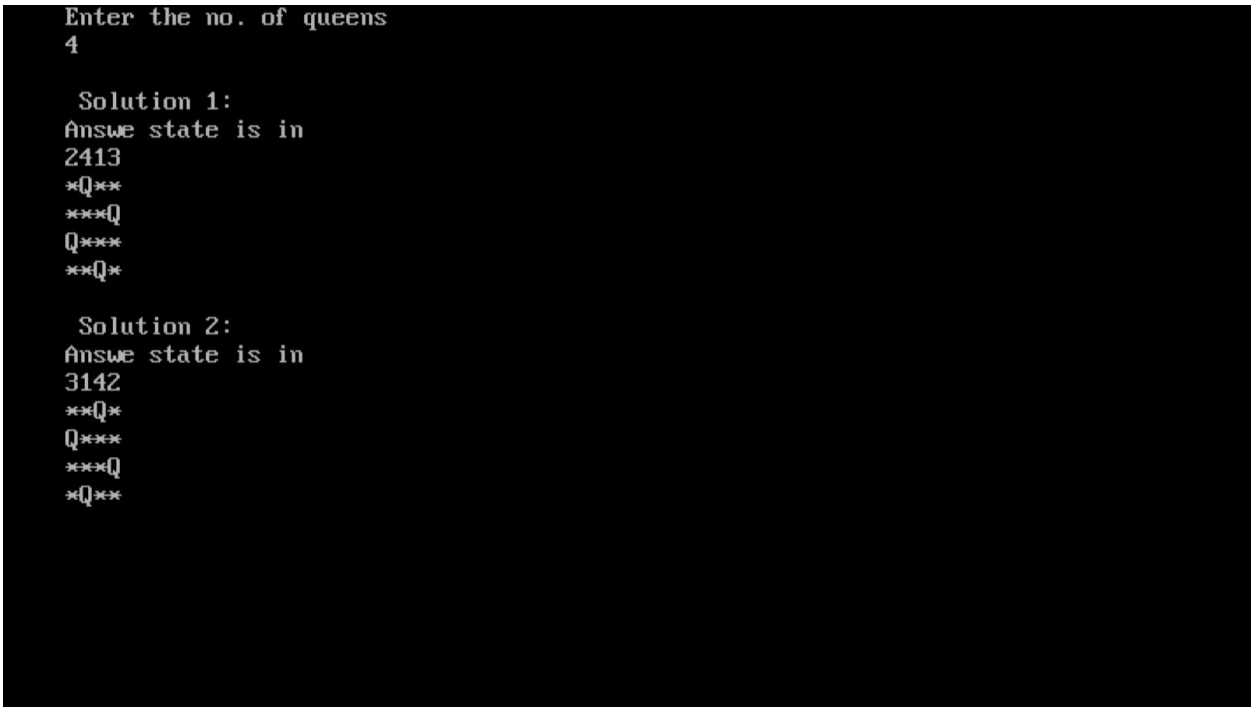


11. Write program to implement backtracking algorithm for solving problems like Nqueens.

```
#include<stdio.h>
#include<conio.h>
int x[25];
int place(int k);
void main()
{
void nqueens(int);
int n;
clrscr();
printf("Enter the no. of queens\n");
scanf("%d",&n);
nqueens(n);
getch();
}

void nqueens(int n)
{
int i,j,k=1,q=1;
x[1]=0;
while(k>0)
```

```
{
x[k]++;
while(x[k]<=n && !place(k)) x[k]++;
if (x[k]<=n)
{
if(k==n)
{
printf("\n Solution %d:\n",q);
q++;
printf("Answe state is in \n");
for(i=1;i<=n;i++)
{
printf("%d",x[i]);
}
printf("\n");
for(i=1;i<=n;i++)
{
for( j=1;j<=n;j++)
printf("%c",x[i]==j?'Q':'*');
printf("\n");
}
}
else
{
k++;
x[k]=0;
}
}
else k--;
}
}

int place(int k)
{
int i;
for(i=1; i<k;i++)
{
if(x[i]==x[k]||abs(x[i]-x[k])==abs(i-k))
{
return 0;
}
}
return -1;
}
```

OUTPUT:



12. Write a program to implement the backtracking algorithm for the sum of subsets problem.

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.>
void subset_sum(int *arr, int size, int d, int sum, int
idx)
{ int i;
    if (sum == d) {
        printf("Found subset: ");
        for (i = 0; i < idx; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return;
    }
        if (idx >= size || sum > d) {
        return;
    }
    subset_sum(arr, size, d, sum + arr[idx], idx + 1);
```

```
    subset_sum(arr, size, d, sum, idx + 1);
}
void main()
{
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int d = 9;
    clrscr();
    subset_sum(arr, size, d, 0, 0);
    getch();
}
```

OUTPUT:

```
Found subset: 1 2 3 4 5
Found subset: 1 2 3 4
Found subset: 1 2 3 4 5
_
```

13. Write program to implement greedy algorithm for job sequencing with deadlines.

```
#include <stdio.h>
#include <conio.h>
#define MAX 10

typedef struct Job {
    char id[5];
    int deadline;
    int profit;
} Job;

void jobSequencing(Job jobs[], int n);

void main() {
    Job jobs[] = {
        {"j1", 2, 60},
        {"j2", 1, 100},
        {"j3", 3, 20},
        {"j4", 2, 40},
        {"j5", 1, 20},
```

```
    };
    int n = 5, i, j;
    Job temp;

    clrscr();
    // Sort jobs by profit in descending order
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (jobs[j].profit < jobs[j + 1].profit) {
                temp = jobs[j];
                jobs[j] = jobs[j + 1];
                jobs[j + 1] = temp;
            }
        }
    }

    printf("Job\tDeadline\tProfit\n");
    for (i = 0; i < n; i++) {
        printf("%s\t%d\t\t%d\n", jobs[i].id, jobs[i].deadline, jobs[i].profit);
    }

    jobSequencing(jobs, n);
    getch();
}

void jobSequencing(Job jobs[], int n) {
    int timeslot[MAX] = {0}, maxProfit = 0, i, j;
    printf("\nSelected Jobs:\n");
    for (i = 0; i < n; i++) {
        for (j = jobs[i].deadline; j > 0; j--) {
            if (timeslot[j] == 0) {
                timeslot[j] = i + 1; // Mark timeslot as filled
                printf("%s ", jobs[i].id);
                maxProfit += jobs[i].profit;
                break;
            }
        }
    }
    printf("\nMax Profit: %d\n", maxProfit);
}
```

OUTPUT:

14. Write program to implement Dynamic Programming algorithm for the Optimal Binary Search Tree Problem.

```c
#include<stdio.h>
#include<conio.h>
#include<limits.h>
int sum(int frq[] , int i, int j);
int optimalsearchtree(int key[], int frq[], int n);
int optcost(int frq[],int i,int j)
{
    int min, r, cost, fsum,m,n,max;
    if(j<i)
    return 0;
    if(j==i)
    return frq[i];
    fsum=sum(frq,i,j);
    min=max;
    for(r=1;r<=j;++r)
    {
        cost=optcost(frq,i,r-1)+optcost(frq,r+1,j);
        if(cost<min)
        min=cost;
    }
    return min+fsum;
}

int optimalsearchtree(int key[], int frq[], int n)
{
 return optcost(frq,0,n-1);
}
int sum(int frq[], int i, int j)
{
    int s=0,k;
```

```
    for(k=i;k<=j;k++)
    s=s+frq[k];
    return s;
}
void main()
{
    int n;
    int key[]={34, 8};
    int frq[]={10,12};
    clrscr();
    n=sizeof(key)/sizeof(key[0]);
    printf("cost of optimal bst is %d",optimalsearchtree(key,frq,n));
    getch();
}
```

OUTPUT:

Cost of optimal bst is :32

15. Write a program that implements Prims algorithm to generate minimum cost spanning Tree.

OUTPUT:

16. Write a program that implements Kruskals algorithm to generate minimum cost spanning tree.

```
#include <stdio.h>
#include <conio.h>

#define INF 999

int cost[10][10], parent[10], n;

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() {
    int i, j, u, v, a, b, ne = 1, min, mincost = 0;

    clrscr();
    printf("Enter the number 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] = INF;
        }

    printf("\nEdges of the Minimum Spanning Tree:\n");
    while (ne < n) {
        min = INF;

        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;
                }
            }
        }

        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] = INF;
    }

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

OUTPUT: