2nd LAB CYCLE PROGRAMS:

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 heap(int a[], int n) {

  for (int i = n / 2; i > 0; i--) {

    int k = i, v = a[k], heap = 0;


    while (!heap && (2 * k <= n)) {

      int j = 2 * k;


      if (j < n && a[j] < a[j + 1])

        j++;

      if (v >= a[j])

        heap = 1;

      else {

        a[k] = a[j];

        k = j;

      }

    }


    a[k] = v;

  }

}


void swap(int *xp, int *yp) {

  int temp = *xp;

  *xp = *yp;
```

```
  *yp = temp;
}

void sort(int a[], int n) {
  heap(a, n);

  for (int k = n; k >= 1; k--) {
    int max = a[1], j = 1;

    for (int i = 1; i <= k; i++) {
      if (max < a[i]) {
        max = a[i];
        j = i;
      }
    }

    swap(&a[1], &a[j]);
    swap(&a[1], &a[k]);
  }
}

int main() {

  int n;
  clock_t start_t,end_t;
  double total_t;

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

  int a[n];
```

```
 for (int i = 1; i <= n; i++)
   a[i]=rand()%50;

            printf("The elements are:\n");
                        for(int i=1;i<=n;i++){
                                    printf("%d\t",a[i]);
                        }

 start_t=clock();
 sort(a, n);
 end_t=clock();

 printf("\nSorted:\n ");
 for (int i = 1; i <= n; i++)
   printf("%d ", a[i]);

 total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;

            printf("\nThe time taken is %f\n",total_t);
}
```

main.c

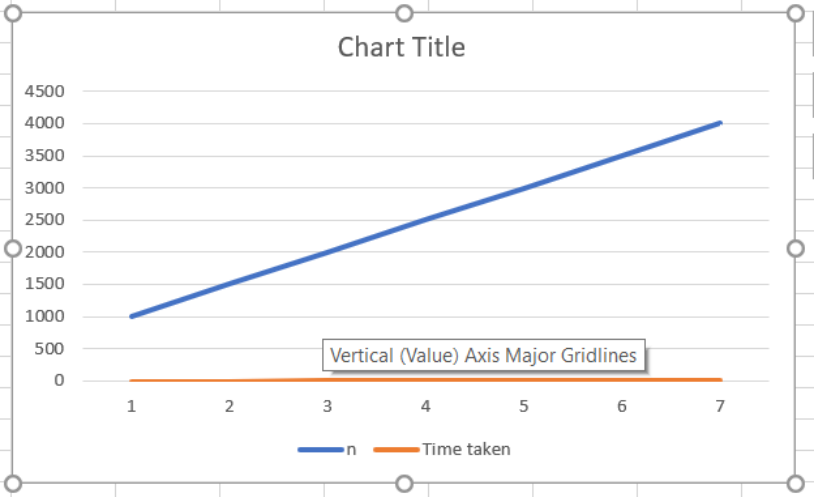
```
int a[n];

for (int i = 1; i <= n; i++)
  a[i]=rand()%50;

  printf("The elements are:\n");
      for(int i=1;i<=n;i++){
          printf("%d\t",a[i]);
      }

start_t=clock();
sort(a, n);
end_t=clock();

printf("\nSorted:\n ");
```

input

```
Enter n: 10
The elements are:
33      36      27      15      43      35      36      42      49      21
Sorted:
 15 21 27 33 35 36 36 42 43 49
The time taken is 0.000003


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

Heapsort

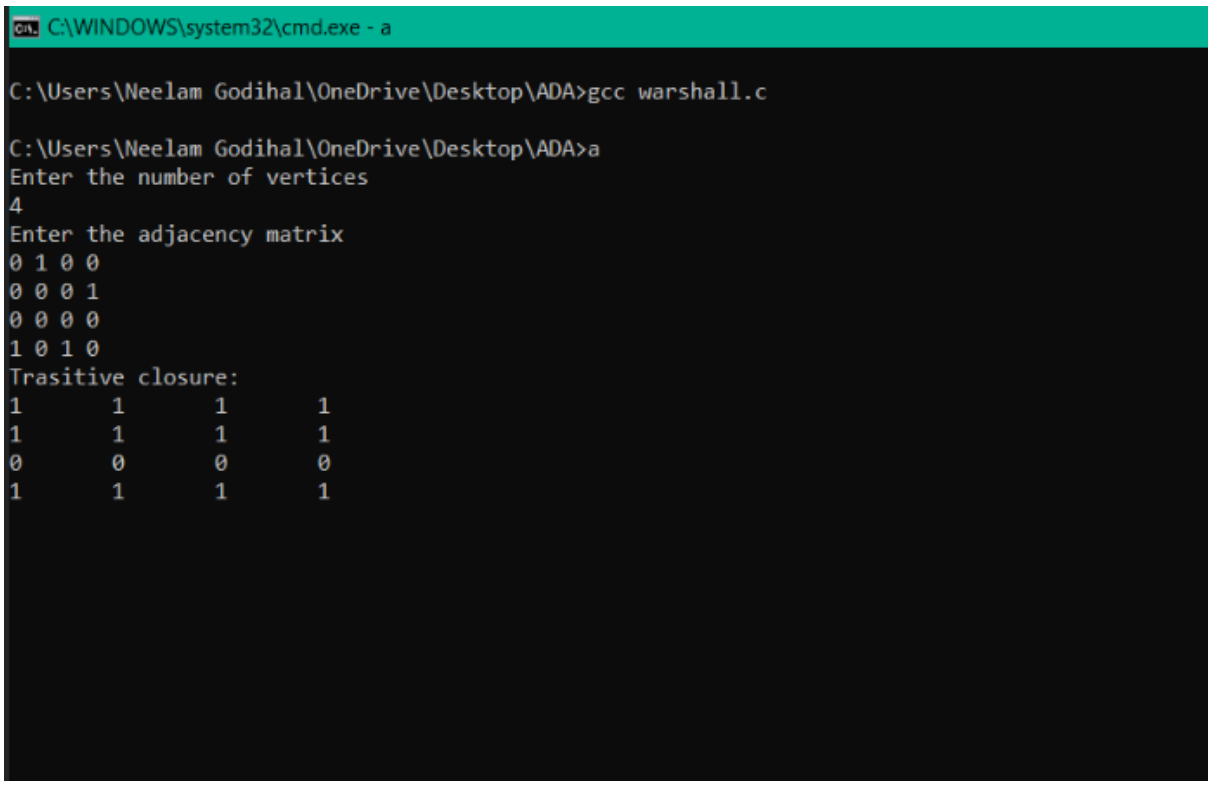
n	Time taken
1000	0.001838
1500	0.003481
2000	0.00621
2500	0.010015
3000	0.012577
3500	0.018979
4000	0.024831



11)Implement Warshall's algorithm using dynamic programming.

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

int n,a[10][10],p[10][10];
void warshall(int n,int a[10][10],int p[10][10])
{
	int i,j,k;

	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			p[i][j]=a[i][j];

	for(k=0;k<n;k++)
		for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			if((p[i][j]==0) && (p[i][k]==1 && p[k][j]==1))
				p[i][j]=1;

}
void main()
{
	int i,j;
	//clrscr();
	printf("Enter the number of vertices\n");
	scanf("%d",&n);
	printf("Enter the adjacency matrix\n");
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
		{
		scanf("%d",&a[i][j]);
```

```
            }
        }
        warshall(n,a,p);
        printf("Trasitive closure:\n");
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
            printf("%d\t",p[i][j]);
            }
        printf("\n");
        }
        getch();
}
```



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

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

void knapsack();
int max(int,int);
int i,j,n,m,p[10],w[10],v[10][10];

void main()
{
        //clrscr();
        printf("\nEnter the no. of items:\t");
        scanf("%d",&n);
        printf("\nEnter the weight of the each item:\n");
        for(i=1;i<=n;i++)
        {
         scanf("%d",&w[i]);
        }
        printf("\nEnter the profit of each item:\n");
        for(i=1;i<=n;i++)
        {
         scanf("%d",&p[i]);
        }
        printf("\nEnter the knapsack's capacity:\t");
        scanf("%d",&m);
        knapsack();
        getch();
}

void knapsack()
{
 int x[10];
```

```
for(i=0;i<=n;i++)
{
 for(j=0;j<=m;j++)
 {
  if(i==0||j==0)
  {
   v[i][j]=0;
  }
  else if(j-w[i]<0)
  {
   v[i][j]=v[i-1][j];
  }
  else
  {
   v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);
  }
 }
}
printf("\nThe output is:\n");
for(i=0;i<=n;i++)
{
 for(j=0;j<=m;j++)
 {
  printf("%d\t",v[i][j]);
 }
 printf("\n\n");
}
printf("\nThe optimal solution is %d",v[n][m]);
printf("\nThe solution vector is:\n");
for(i=n;i>=1;i--)
{
 if(v[i][m]!=v[i-1][m])
```

```
  {
   x[i]=1;
   m=m-w[i];
  }
  else
  {
   x[i]=0;
  }
 }
 for(i=1;i<=n;i++)
 {
  printf("%d\t",x[i]);
 }
}

int max(int x,int y)
{
 if(x>y)
 {
  return x;
 }
 else
 {
  return y;
 }
}
```

```
C:\WINDOWS\system32\cmd.exe - a
0       0       0       0       ^C
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc knapsack.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a

Enter the no. of items: 4

Enter the weight of the each item:
2
1
3
2

Enter the profit of each item:
12
10
20
15

Enter the knapsack's capacity:  5

The output is:
0       0       0       0       0       0

0       0       12      12      12      12

0       10      12      22      22      22

0       10      12      22      30      32

0       10      15      25      30      37


The optimal solution is 37
The solution vector is:
1       1       0       1
```

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

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

int a[10][10],n;
void floyds();
int min(int,int);

void main()
{
 int i,j;
 int source,dest;

 printf("\nEnter the no. of vertices:\t");
 scanf("%d",&n);
 printf("\nEnter the cost matrix:\n");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   scanf("%d",&a[i][j]);
  }
 }
 floyds();
 printf("-------------------------------------------------------");
 printf("\nEnter the source:");
 scanf("%d",&source);
 printf("Enter the dest:");
 scanf("%d",&dest);
 printf("Shortest distance between %d vertice and %d vertice is %d.", source, dest, a[source][dest]);
 getch();
}
```

```
void floyds()
{
 int i,j,k;
 for(k=1;k<=n;k++)
 {
  for(i=1;i<=n;i++)
  {
   for(j=1;j<=n;j++)
   {
    a[i][j]=min(a[i][j],a[i][k]+a[k][j]);
   }
  }
 }
 printf("\nAll pair shortest path matrix is:\n");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   printf("%d\t",a[i][j]);
  }
  printf("\n\n");
 }
}

int min(int x,int y)
{
 if(x<y)
 {
  return x;
 }
 else
```

```
 {
  return y;
 }
}
```

```
C:\WINDOWS\system32\cmd.exe - a
^C
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc floyd.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a

Enter the no. of vertices:      4

Enter the cost matrix:
0 9999 3 9999
2 0 9999 9999
9999 7 0 1
6 9999 9999 0

All pair shortest path matrix is:
0       10      3       4

2       0       5       6

7       7       0       1

6       16      9       0
```

14) Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.

```
#include<stdio.h>

#include<conio.h>

#include<process.h>


void prims();

int c[10][10],n;


void main()

{

 int i,j;

 printf("\nEnter the no. of vertices:\t");

 scanf("%d",&n);

 printf("\nEnter the cost matrix:\n");

 for(i=1;i<=n;i++)

 {

  for(j=1;j<=n;j++)

  {

   scanf("%d",&c[i][j]);

  }

 }

 prims();

 getch();

}


void prims()

{

 int i,j,u,v,min;

 int ne=0,mincost=0;

 int elec[10];

 for(i=1;i<=n;i++)
```

```
{
 elec[i]=0;
}
elec[1]=1;
while(ne!=n-1)
{
 min=9999;
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   if(elec[i]==1)
   {
    if(c[i][j]<min)
    {
     min=c[i][j];
     u=i;
     v=j;
    }
   }
  }
 }
 if(elec[v]!=1)
 {
  printf("\n%d----->%d=%d\n",u,v,min);
  elec[v]=1;
  ne=ne+1;
  mincost=mincost+min;
 }
 c[u][v]=c[v][u]=9999;
}
printf("\nMincost=%d",mincost);
```

}

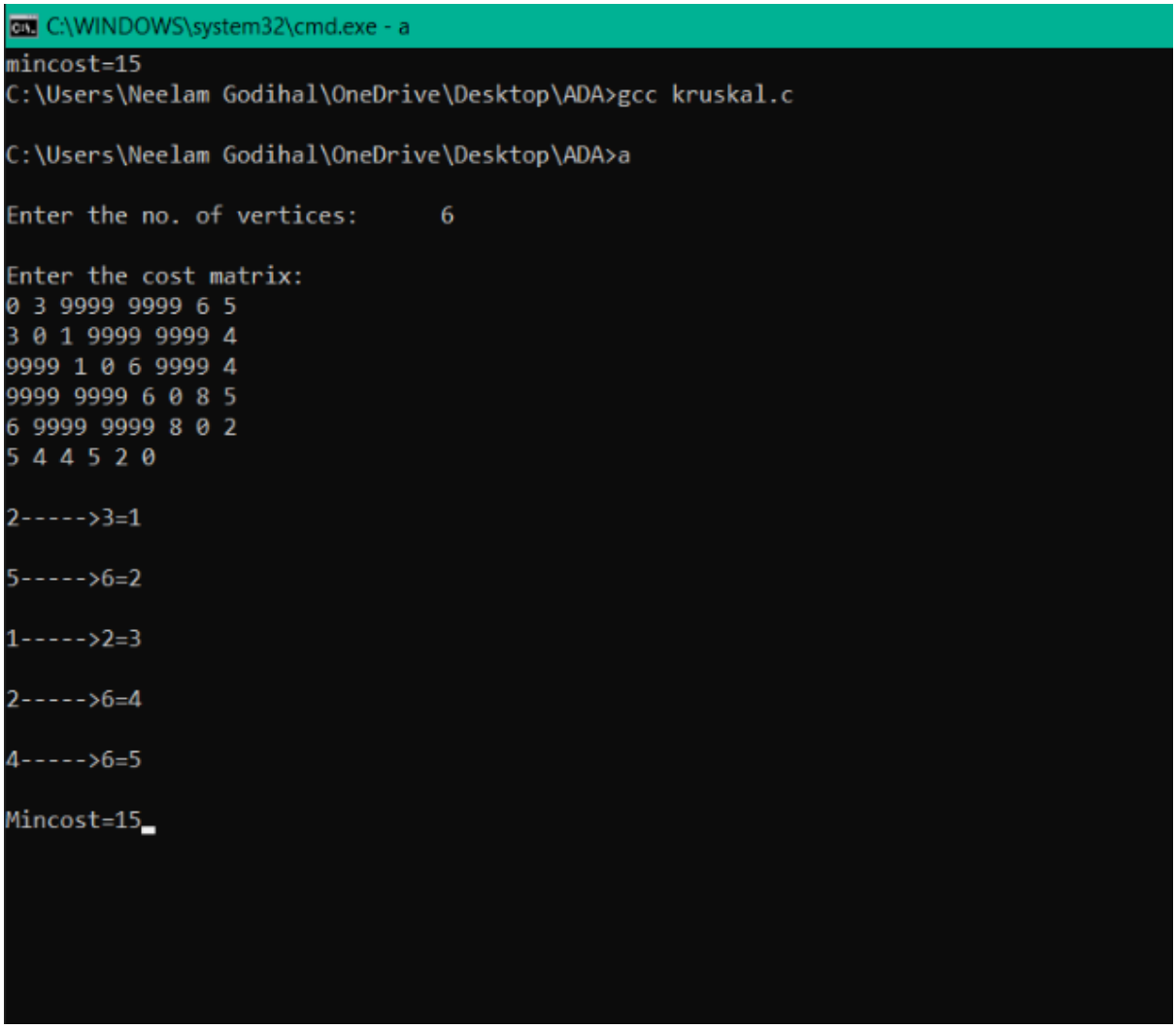
```
C:\WINDOWS\system32\cmd.exe - a
collect2.exe: error: ld returned 1 exit status

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc prim.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a

Enter the no. of vertices:      6

Enter the cost matrix:
0 3 9999 9999 6 5
3 0 1 9999 9999 4
9999 1 0 6 9999 4
9999 9999 6 0 8 5
6 9999 9999 8 0 2
5 4 4 5 2 0

1----->2=3

2----->3=1

2----->6=4

6----->5=2

6----->4=5

mincost=15
```

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

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

void kruskals();
int c[10][10],n;

void main()
{
 int i,j;
 //clrscr();
 printf("\nEnter the no. of vertices:\t");
 scanf("%d",&n);
 printf("\nEnter the cost matrix:\n");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   scanf("%d",&c[i][j]);
  }
 }
 kruskals();
 getch();
}

void kruskals()
{
 int i,j,u,v,a,b,min;
 int ne=0,mincost=0;
 int parent[10];
 for(i=1;i<=n;i++)
```

```
{
 parent[i]=0;
}
while(ne!=n-1)
{
 min=9999;
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   if(c[i][j]<min)
   {
    min=c[i][j];
    u=a=i;
    v=b=j;
   }
  }
 }
 while(parent[u]!=0)
 {
  u=parent[u];
 }
 while(parent[v]!=0)
 {
  v=parent[v];
 }
 if(u!=v)
 {
  printf("\n%d----->%d=%d\n",a,b,min);
  parent[v]=u;
  ne=ne+1;
  mincost=mincost+min;
```

```
  }
  c[a][b]=c[b][a]=9999;
 }
 printf("\nMincost=%d",mincost);
}
```



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>

void dijkstras();
int c[10][10],n,src;

void main()
{
 int i,j;
 //clrscr();
 printf("\nEnter the no of vertices:\t");
 scanf("%d",&n);
 printf("\nEnter the cost matrix:\n");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   scanf("%d",&c[i][j]);
  }
 }
 printf("\nEnter the source node:\t");
 scanf("%d",&src);
 dijkstras();
 getch();
}

void dijkstras()
{
  int vis[10],dist[10],u,j,count,min;
  for(j=1;j<=n;j++)
```

```
{
 dist[j]=c[src][j];
}
for(j=1;j<=n;j++)
{
 vis[j]=0;
}
dist[src]=0;
vis[src]=1;
count=1;
while(count!=n)
{
 min=9999;
 for(j=1;j<=n;j++)
 {
  if(dist[j]<min&&vis[j]!=1)
  {
   min=dist[j];
   u=j;
  }
 }
 vis[u]=1;
 count++;
 for(j=1;j<=n;j++)
 {
  if(min+c[u][j]<dist[j]&&vis[j]!=1)
  {
   dist[j]=min+c[u][j];
  }
 }
}
printf("\nThe shortest distance is:\n");
```

```
  for(j=1;j<=n;j++)
  {
   printf("\n%d----->%d=%d",src,j,dist[j]);
  }
}
```

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc dij.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a

Enter the no of vertices:       6

Enter the cost matrix:
0 3 9999 9999 6 5
3 0 1 9999 9999 4
9999 1 0 6 9999 4
9999 9999 6 0 8 5
6 9999 9999 8 0 2
5 4 4 5 2 0

Enter the source node:  1

The shortest distance is:

1----->1=0
1----->2=3
1----->3=4
1----->4=10
1----->5=6
1----->6=5
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a

Enter the no of vertices:       6

Enter the cost matrix:
0 3 9999 9999 6 5
3 0 1 9999 9999 4
9999 1 0 6 9999 4
9999 9999 6 0 8 5
6 9999 9999 8 0 2
5 4 4 5 2 0

Enter the source node:  2

The shortest distance is:

2----->1=3
2----->2=0
2----->3=1
2----->4=7
2----->5=6
2----->6=4
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>
```

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>


int count,w[10],d,x[10];


void subset(int cs, int k, int r)

{

	int i;

	x[k]=1;

	if(cs+w[k]==d)

	{

		printf("\nSubset solution = %d\n", ++count);

		for(i=0;i<=k;i++)

		{

			if(x[i]==1)

			printf("%d ", w[i]);

		}

	}

	else

	if(cs+w[k]+w[k+1]<=d)

	subset(cs+w[k], k+1, r-w[k]);

	if((cs+r-w[k]>=d) && (cs+w[k+1])<=d)

	{

		x[k]=0;

		subset(cs,k+1,r-w[k]);

	}

}
```

```
void  main()
{
        int sum=0,i,n;
        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]);

        printf("Enter the required sum\n");
        scanf("%d", &d);
        for(i=0;i<n;i++)
        sum+=w[i];
        if(sum<d)
        {
                printf("No solution exists!\n");
                return;
        }
        printf("The solution is:\n");
        count=0;
        subset(0,0,sum);
        getch();
}
```

C:\WINDOWS\system32\cmd.exe - a

```
78
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc subset.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a
Enter the number of elements
4
Enter the elements in ascending order
5
7
8
10
Enter the required sum
15
The solution is:

Subset solution = 1
5 10
Subset solution = 2
7 8
```

18)Implement “N-Queens Problem” using Backtracking.

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

int canplace(int r,int c[50])
{
   int i;
   for(i=0;i<r;i++)
   {
      if(c[i]==c[r] || abs(c[i]-c[r])==abs(i-r))
            return 0;
   }
   return 1;
}

void display(int c[50],int n)
{
    int i,j;
    char cb[10][10];
    for(i=0;i<n;i++)
       for(j=0;j<n;j++)
             cb[i][j]='-';
   for(i=0;i<n;i++)
       cb[i][c[i]]='q';
   printf("-----------------------------\n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
            printf("%c\t ",cb[i][j]);
      }
      printf("\n");
```

```
   }
}

void nqueen(int n)
{
   int r,c[50];
   c[0]=-1;
   r=0;
   while(r>=0)
   {
      c[r]++;
      while(c[r]<n && !canplace(r,c))
            c[r]++;
     if(c[r]<n)
     {
            if(r==n-1)
            {
               display(c,n);
               printf("\n");
            }
            else
            {
               r++;
               c[r]=-1;
            }
     }
     else
            r--;
  }
}

void main()
```

```
{
    int n;
    printf("Enter the number of queens\n");
    scanf("%d",&n);
    nqueen(n);
    getch();
}
```

```
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc n_queens.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a
Enter the number of queens
4
----------------------------
-        q       -       -
-        -       -       q
q        -       -       -
-        -       q       -

----------------------------
-        -       q       -
q        -       -       -
-        -       -       q
-        q       -       -


C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>_
```