Write a c code to implement drijistras algorithm

```
#include <stdbool.h>

#include <stdio.h>

#define MAX 999

int V;

int parents[50], noParent=-1;


int minDistance(int dist[], bool picked[])

{

	int min = MAX, min_index;


	for (int v = 0; v < V; v++)

		if (picked[v] == false && dist[v] <= min)

			min = dist[v], min_index = v;


	return min_index;

}

void printPath(int vertx,int parents[V])

{

	if (vertx == noParent) {

		return;

	}

	printPath(parents[vertx], parents);

	printf("%d  ",vertx);

}

void printSolution(int dist[])

{

	printf("\nVertex \t\t Distance from source\t\tPath\n");

	for (int i = 0; i < V; i++){
```

```
        printf("  %d \t\t\t  %d \t\t\t", i, dist[i]);
                        printPath(i,parents);
                        printf("\n");
            }
}

void dijkstra(int graph[V][V], int src)
{
            int dist[V];

            bool picked[V];
            for (int i = 0; i < V; i++)
                        dist[i] = MAX, picked[i] = false;

            dist[src] = 0;
    parents[0]=noParent;
            for (int count = 0; count < V - 1; count++) {
                        int u = minDistance(dist, picked);

                        picked[u] = true;

                        for (int v = 0; v < V; v++){
          if (!picked[v] && graph[u][v]
                                    && dist[u] != MAX
                                    && dist[u] + graph[u][v] < dist[v]){
                                       dist[v] = dist[u] + graph[u][v];
                                    parents[v]=u;
                                    }
                        }
```

```
        }

        printSolution(dist);
}

int main()
{
   printf("Enter the number of vertices\n");
   scanf("%d",&V);
        int graph[V][V],j;
        printf("Enter the matrix\n");
        for(int i=0;i<V;i++){
     for(j=0;j<V;j++){
        scanf("%d",&graph[i][j]);
     }
        }

        dijkstra(graph, 0);

        return 0;
```

C:\Users\lenovo\Desktop\ADA\drijistras.exe

```
Enter the number of vertices
6
Enter the matrix
0 25 35 99 100 99
99 0 27 14 99 99
99 99 0 299 99 99
99 99 99 0 99 21
99 99 50 99 0 99
99 99 99 99 48 0

Vertex          Distance from source          Path
  0                     0                     0
  1                     25                    0  1
  2                     35                    0  2
  3                     39                    0  1  3
  4                     100                   0  4
  5                     60                    0  1  3  5

Process returned 0 (0x0)   execution time : 42.781 s
Press any key to continue.
```

}

Write a c code to for N-Queens problem

```
#include <stdio.h>

#define MAX 10

int x[MAX];


int place(int k) {

   int i;

   for (i = 1; i < k; i++) {

      if (x[i] == x[k] || i - x[i] == k - x[k] || i + x[i] == k + x[k]) {

         return 0;

      }

   }

   return 1;

}


void write(int n) {

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

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

         if (j == x[i])

            printf("Q%d\t",i);

         else

            printf("-\t");

      }

      printf("\n");

   }

   printf("\n\n");

}


void nqueens(int n) {
```

```
    int k = 1;
    x[k] = 0;

    while (k != 0) {
        x[k] = x[k] + 1;

        while (x[k] <= n && !place(k)) {
            x[k] = x[k] + 1;
        }

        if (x[k] <= n) {
            if (k == n) {
                write(n);
            } else {
                k = k + 1;
                x[k] = 0;
            }
        } else {
            k = k - 1;
        }
    }
}

int main() {
    int n;
    printf("Enter the value of N: ");
    scanf("%d", &n);
    nqueens(n);
    return 0;
```

}

C:\Users\lenovo\Desktop\ADA\N-Queens.exe

```
Enter the value of N: 4
-       Q1      -       -
-       -       -       Q2
Q3      -       -       -
-       -       Q4      -


-       -       Q1      -
Q2      -       -       -
-       -       -       Q3
-       Q4      -       -



Process returned 0 (0x0)   execution time : 1.543 s
Press any key to continue.
```

Write a c-code to implement drijistra's Algorithm.

```
# include <stdio.h>
# include <conio.h>
# define infinity 9999
# define Max 10.
void drijistra(int G[Max][Max], int n, int startnode);
int main()
{
int G[Max][Max], i, j, n, u;
Printf("Enter no. 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", &G[i][j]);
Printf("\n Enter the starting node:");
scanf("%d", &u);
dijistra(G, n, u);
return 0;
}
void dijistra (int G[Max][Max], int n, int startnode)
{
int cost [Max][Max], distance[Max], pred [Max];
int visited [Max], count, mindistance, nextnode, i, j;
for (i= 0; i<n; i++)
  for (j=0; j<n; j++)
  if (G[i][j] = = 0)
     cost [i][j] = infinity;
else
cost [i][j] = G[i][j];
for (i=0; i<n; i++)
{
distance[i] = cost [startnode][i];
```

```
Pred [i] = start node;
visited [i]=0;
}
distance [startnode] = 0;
visited [startnode]=1;
count=1;
while (count<n-1)
{
mindistance = infinity;
for (i=0; i<n;i++)
if (distance [i]<mindistance && !visited [i])
{
mindistance = distance [i];
next node = i;
}
visited [next node]= 1;
for (i=0; i<n; i++)
if (!visited [i])
if (mindistance + cost [next node] [i]<distance [i])
{
distance [i] = mindistance + cost [next node][i];
pred [i]= nextnode;
}
count++;
}
for (i=0; i<n; i++)
if (i! = start node)
{
Printf ("\n Distance of node %d = %d", i, distance [i]);
printf ("\n Path= %d", i);
   j= i;
do
{ j = pred [j];
   Printf ("%d", j);
}while (j = start node); }}
```

Output.

Enter no. of vertices : 6

Enter the adjacency matrix.

0	25	35	99	100	99
99	0	27	14	99	99
99	99	0	29	99	99
99	99	99	0	99	21
99	99	50	99	0	99
99	99	99	99	48	0

Enter the starting node : 0

Distance of node 1 = 25

Path = 1 ← 0

Distance of node 2 = 35

Path = 2 ← 0

Distance of nod 3 = 39

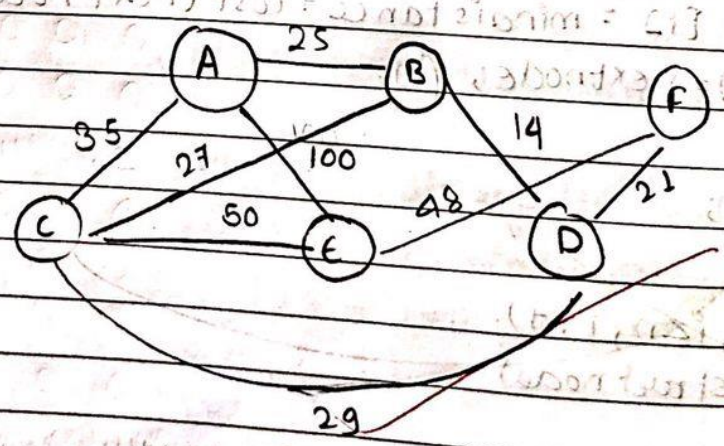
Path = 3 ← 1 ← 0

Distance of node 4 = 100

Path = 4 ← 0

Distance of node 5 = 60

Path = 5 ← 3 ← 1 ← 0.



write a c-code to implement N-queens.

```
#include <stdio.h>
# include <math.h>
int m [20];
int count =0;
int place (int k, int i) {
 for (int j = 1; j <= k - 1; j++) {
if (m[j] = = i || abs (m[j] - i) = = abs(j-k)) {
 return 0;
 }
}
return 1;
}
void nqueens (int k, int n) {
 for (int i = 1; i <= n; i++) {
 if (place (k, i)) {
 m [k] = i;
 if (k = = n) {
 count ++;
 Printf ("solution %d: \n", count);
 for (int j = 1; j <= n; j++) {
 for (int l = 1; l <= n; l++) {
 Printf ("Q");
 }
 else
 Printf ("0");
 }
 }
 Printf ("\n");
 }
 Printf ("\n");
 } else {
 nqueens (k+1, n);
 }
} } }
```

```
int main() {
int n;
Printf ("Enter the no. of queens: ");
scanf ("%d", &n);
if (n<=0) {
Printf ("Invalid input \n");
return 1;
}
nqueens (1, n);
if (count ==0) {
Printf ("No solutions found for %d queens. \n", n);
} else {
Printf ("Total solutions: %d \n", count);
}
return 0;
}
```

output

```
Enter the number of queens: 4
solution 1:
0 Q 0 0
0 0 0 Q
Q 0 0 0
0 0 Q 0
solution 2
0 0 Q 0
Q 0 0 0
0 0 0 Q
0 Q 0 0
Total solutions : 2.
```

10/10/23