1>

```
#include <stdio.h>

#define INF 999

int cost[100][100], parent[100];


int find(int i) {

    while (parent[i]) i = parent[i];

    return i;

}


void unite(int i, int j) {

    parent[j] = i;

}


int main() {

    int n, i, j, u, v, min, a = -1, b = -1, sum = 0, edges = 0;

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

    printf("Enter cost matrix (0=self, 999=no edge):\n");

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


    printf("Edges in MST:\n");

    while (edges < n - 1) {

        min = INF;

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

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

                if (i != j && cost[i][j] < min && find(i) != find(j)) {

                    min = cost[i][j]; a = i; b = j;

                }

            }

        }

        unite(find(a), find(b));

        printf("%d -> %d = %d\n", a, b, min);

        sum += min; edges++;

        cost[a][b] = cost[b][a] = INF;

    }

    printf("Total cost of MST: %d\n", sum);

    return 0;

}
```

```
Enter number of vertices: 4

Enter cost matrix (0=self, 999=no edge):

0 1 3 999

1 0 2 4

3 2 0 5

999 4 5 0
```

2>

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

```
#define INF 999

int main() {
    int n, i, j, u, v, min, cost[10][10], dist[10], parent[10], visited[10] = {0}, sum = 0;
    printf("Enter no. of vertices: "); scanf("%d", &n);
    printf("Enter cost matrix (0=self, 999=no edge):\n");
    for (i = 1; i <= n; i++) for (j = 1; j <= n; j++) scanf("%d", &cost[i][j]);
    printf("Enter source vertex: "); scanf("%d", &u);

    for (i = 1; i <= n; i++) dist[i] = cost[u][i], parent[i] = u;
    visited[u] = 1;

    for (i = 1; i < n; i++) {
        min = INF;
        for (j = 1; j <= n; j++)
            if (!visited[j] && dist[j] < min) min = dist[j], v = j;

        visited[v] = 1; sum += dist[v];
        printf("%d -> %d = %d\n", parent[v], v, dist[v]);

        for (j = 1; j <= n; j++)
            if (!visited[j] && cost[v][j] < dist[j])
                dist[j] = cost[v][j], parent[j] = v;
    }

    printf("Total cost = %d\n", sum);
    return 0;
}
```

4>

```
#include <stdio.h>
#define INF 999

void dijkstra(int cost[10][10], int n, int src, int dist[10]) {
    int vis[10] = {0}, i, j, min, u;
    for (i = 1; i <= n; i++) dist[i] = cost[src][i];
    vis[src] = 1;

    for (i = 1; i < n; i++) {
        min = INF;
        for (j = 1; j <= n; j++)
            if (!vis[j] && dist[j] < min) min = dist[j], u = j;
        vis[u] = 1;
        for (j = 1; j <= n; j++)
            if (!vis[j] && dist[u] + cost[u][j] < dist[j])
                dist[j] = dist[u] + cost[u][j];
    }
}

int main() {
    int n, i, j, src, cost[10][10], dist[10];
    printf("Enter number of nodes: "); scanf("%d", &n);
    printf("Enter cost matrix (999 = no edge):\n");
    for (i = 1; i <= n; i++) for (j = 1; j <= n; j++) scanf("%d", &cost[i][j]);
    printf("Enter source node: "); scanf("%d", &src);
    dijkstra(cost, n, src, dist);
    for (i = 1; i <= n; i++)
        printf("Shortest Distance from %d to %d = %d\n", src, i, dist[i]);
    return 0;
}
```

```
Enter number of nodes: 4
Enter cost matrix (999 = no edge):
0 3 999 7
3 0 1 999
999 1 0 2
7 999 2 0
Enter source node: 1
```

5>

```
#include <stdio.h>

int main() {
    int n, i, j, a[10][10], indeg[10] = {0}, temp[10], k = 0;
    printf("Enter number of vertices: "); scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++) {
            scanf("%d", &a[i][j]);
            if (a[i][j]) indeg[j]++;
        }

    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++)
            if (indeg[j] == 0) {
                temp[k++] = j; indeg[j] = -1;
                for (int x = 1; x <= n; x++) if (a[j][x]) indeg[x]--;
                break;
            }
    }

    if (k < n) printf("Topological ordering is not possible\n");
    else {
        printf("Topological ordering: ");
        for (i = 0; i < k; i++) printf("%d ", temp[i]);
    }
    return 0;
}
```

```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 0 0
0 0 1 0
0 0 0 1
0 0 0 0
```

6>

```
#include<stdio.h>
int w[10], p[10], n;
int max(int a, int b) {
    return a > b ? a : b;
} int knap(int i, int m) {
    if (i > n)  // base case corrected
        return 0;
    if (w[i] > m)
        return knap(i + 1, m);
    return max(knap(i + 1, m), knap(i + 1, m - w[i]) + p[i]);
}
void main() {
    int m, i, max_profit;
    printf("\nEnter the number of objects: ");
    scanf("%d", &n);
   printf("\nEnter the knapsack capacity: ");
    scanf("%d", &m);
    printf("\nEnter profit followed by weight:\n");
    for (i = 1; i <= n; i++)
        scanf("%d %d", &p[i], &w[i]);
    max_profit = knap(1, m);
    printf("\nMax profit = %d\n", max_profit);
}
```

7>

```
#include <stdio.h>
#define MAX 50
float w[MAX], p[MAX], x[MAX];
int n, m;

void greedyKnapsack() {
    float ratio[MAX], temp;
    int i, j, cw = 0;
    double maxprofit = 0.0;

    for (i = 0; i < n; i++) ratio[i] = p[i] / w[i];

    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (ratio[i] < ratio[j]) {
                temp = ratio[i]; ratio[i] = ratio[j]; ratio[j] = temp;
                temp = w[i]; w[i] = w[j]; w[j] = temp;
                temp = p[i]; p[i] = p[j]; p[j] = temp;
            }
        }
    }

    for (i = 0; i < n; i++) {
        if (cw + w[i] <= m) {
            x[i] = 1.0; cw += w[i]; maxprofit += p[i];
        } else {
            x[i] = (m - cw) / w[i];
            maxprofit += x[i] * p[i];
            break;
        }
    }
    printf("Max profit: %.2f\n", maxprofit);
}

int main() {
    printf("Enter number of items: "); scanf("%d", &n);
    printf("Enter weights: "); for (int i = 0; i < n; i++) scanf("%f", &w[i]);
    printf("Enter profits: "); for (int i = 0; i < n; i++) scanf("%f", &p[i]);
    printf("Enter capacity: "); scanf("%d", &m);
    greedyKnapsack();
    return 0;
}
```

```
Enter number of items: 3
Enter weights: 10 20 30
Enter profits: 60 100 120
Enter capacity: 50
```

8>

```
#include <stdio.h>
#define MAX 10
int s[MAX], x[MAX], d;

void sumofsub(int p, int k, int r) {
    x[k] = 1;

    if (p + s[k] == d) {

        printf("{ "); for (int i = 1; i <= k; i++) if (x[i]) printf("%d ", s[i]); printf("}\n");

    } else if (p + s[k] + s[k+1] <= d)

        sumofsub(p + s[k], k + 1, r - s[k]);

    if ((p + r - s[k] >= d) && (p + s[k+1] <= d)) {

        x[k] = 0;

        sumofsub(p, k + 1, r - s[k]);

    }

}


int main() {

    int n, sum = 0;

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

    printf("Enter set (sorted): "); for (int i = 1; i <= n; i++) scanf("%d", &s[i]), sum += s[i];

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

    if (sum < d || s[1] > d) printf("No subset possible\n");

    else sumofsub(0, 1, sum);

    return 0;

}
```

```
Enter number of elements: 4

Enter set (sorted): 2 4 6 8

Enter target sum: 10
```

12>

```
#include <stdio.h>

#include <stdlib.h>

#define MAX 50


int can_place(int c[], int r) {

    for (int 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[], int n) {

    for (int i = 0; i < n; i++, printf("\n"))
```

```
        for (int j = 0; j < n; j++)
            printf("%c", c[i] == j ? 'Q' : '-');
    printf("\n");
}

void n_queens(int n) {
    int c[MAX], r = 0; c[0] = -1;
    while (r >= 0) {
        c[r]++;
        while (c[r] < n && !can_place(c, r)) c[r]++;
        if (c[r] < n) {
            if (r == n - 1) display(c, n);
            else c[++r] = -1;
        } else r--;
    }
}
int main() {
    int n;
    printf("Enter number of queens: ");
    scanf("%d", &n);
    n_queens(n);
    return 0;
}
```