ADA LAB

Backtracking

1. N-Queens Problem

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int board[MAX], n;
int isSafe(int row, int col) {
   for (int i = 1; i < row; i++) {
      if (board[i] == col || abs(board[i] - col) == abs(i - row))
         return 0;
   }
   return 1;
}

void printSolution() {
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
         if (board[i] == j)
            printf("Q ");
         else
            printf(". ");
      }
      printf("\n");
   }
   printf("\n");
}
```

```
void solve(int row) {
   for (int col = 1; col <= n; col++) {
      if (isSafe(row, col)) {
         board[row] = col;
         if (row == n)
            printSolution();
         else
            solve(row + 1);
      }
   }
}

int main() {
   printf("Enter number of queens: ");
   scanf("%d", &n);
   solve(1);
   return 0;
}
```

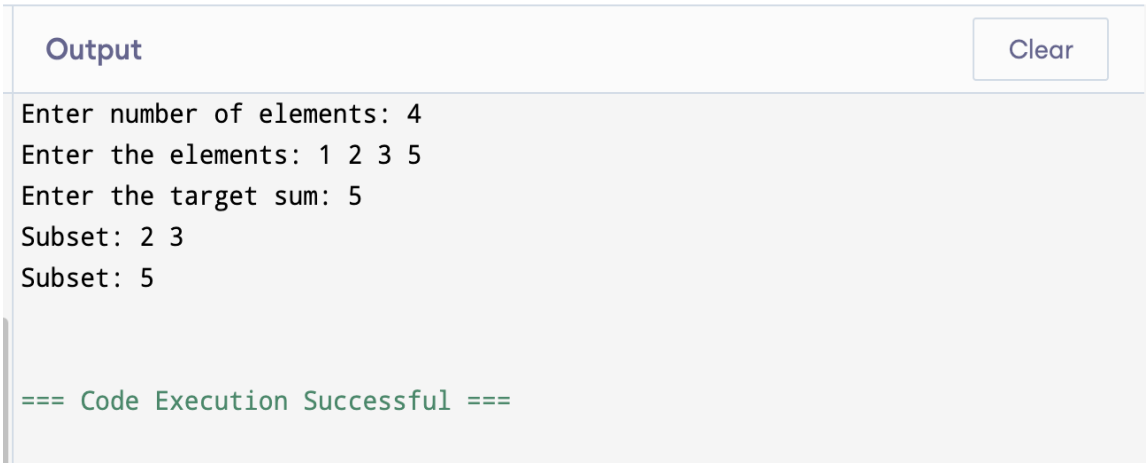
Output:

2. Sum of Subsets Problem

```
#include <stdio.h>

int w[10], x[10], n, sum, total = 0;

void sumOfSubsets(int i, int weight, int targetSum) {
   if (weight == targetSum) {
      printf("Subset: ");
      for (int j = 0; j < i; j++) {
         if (x[j])
            printf("%d ", w[j]);
      }
      printf("\n");
      return;
   }

   if (i >= n || weight > targetSum) return;

   x[i] = 1;
   sumOfSubsets(i + 1, weight + w[i], targetSum);
   x[i] = 0;
   sumOfSubsets(i + 1, weight, targetSum);
}

int main() {
   int target;
   printf("Enter number of elements: ");
   scanf("%d", &n);
```

```
    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &w[i]);
        total += w[i];
    }
    printf("Enter the target sum: ");
    scanf("%d", &target);
    sumOfSubsets(0, 0, target);
    return 0;
}
```

Output:



3.Floyd's Algorithm

```
#include <stdio.h>
#define INF 9999
#define MAX 10

int main() {
   int V, i, j, k;
   int graph[MAX][MAX], dist[MAX][MAX];

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

   printf("Enter adjacency matrix (use %d for INF):\n", INF);
   for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
         scanf("%d", &graph[i][j]);

   for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
         dist[i][j] = graph[i][j];

   for (k = 0; k < V; k++)
      for (i = 0; i < V; i++)
         for (j = 0; j < V; j++)
            if (dist[i][k] + dist[k][j] < dist[i][j])
               dist[i][j] = dist[i][k] + dist[k][j];

   printf("Shortest distances:\n");
```

```
    for (i = 0; i < V; i++) {
        for (j = 0; j < V; j++) {
            if (dist[i][j] == INF)
                printf("INF ");
            else
                printf("%3d ", dist[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

Output Clear

```
Enter number of vertices: 4
Enter adjacency matrix (use 9999 for INF):
0 9999 3 9999
2 0 9999 9999
9999 7 0 1
6 9999 9999 0
Shortest distances:
  0  10   3   4
  2   0   5   6
  7   7   0   1
  6  16   9   0


=== Code Execution Successful ===
```

4.Warshall's Algorithm

```
#include <stdio.h>

#define MAX 10

int main() {
   int V, i, j, k;
   int graph[MAX][MAX], reach[MAX][MAX];

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

   printf("Enter adjacency matrix (0 or 1):\n");
   for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
         scanf("%d", &graph[i][j]);

   for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
         reach[i][j] = graph[i][j];

   for (k = 0; k < V; k++)
      for (i = 0; i < V; i++)
         for (j = 0; j < V; j++)
            reach[i][j] = reach[i][j] || (reach[i][k] && reach[k][j]);

   printf("Transitive closure:\n");
   for (i = 0; i < V; i++) {
```

```
    for (j = 0; j < V; j++)
        printf("%d ", reach[i][j]);
    printf("\n");
  }

  return 0;
}
```

Output: