11. topological source removal

```c
#include <stdio.h>

int adjMatrix[10][10], nodes, sortedOrder[10], inDegree[10];
int stack[10], stackTop = -1;

void calculateInDegree(int nodes, int adjMatrix[][10]) {
    for (int i = 0; i < nodes; i++) {
        inDegree[i] = 0;
        for (int j = 0; j < nodes; j++) {
            inDegree[i] += adjMatrix[j][i];
        }
    }
}

void topologicalSortBySourceRemoval(int nodes, int adjMatrix[][10]) {
    int currentNode, sortedIndex = 0;

    for (int i = 0; i < nodes; i++) {
        if (inDegree[i] == 0) {
            stack[++stackTop] = i;
        }
    }

    while (stackTop != -1) {
        currentNode = stack[stackTop--];
        sortedOrder[sortedIndex++] = currentNode;

        for (int i = 0; i < nodes; i++) {
            if (adjMatrix[currentNode][i] != 0) {
                inDegree[i]--;
                if (inDegree[i] == 0) {
                    stack[++stackTop] = i;
                }
            }
        }
    }

    if (sortedIndex != nodes) {
        printf("Graph contains a cycle. Topological sort is not possible.\n");
    } else {
        printf("Topological Sort (Source Removal): ");
```

```
        for (int i = 0; i < nodes; i++) {
            printf("%d ", sortedOrder[i]);
        }
        printf("\n");
    }
}

int main() {
    printf("Enter the number of nodes: ");
    scanf("%d", &nodes);

    printf("Enter the adjacency matrix (use 0 for no edge and non-zero values for edges):\n");
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            scanf("%d", &adjMatrix[i][j]);
        }
    }

    calculateInDegree(nodes, adjMatrix);
    topologicalSortBySourceRemoval(nodes, adjMatrix);

    return 0;
}
```

```
Enter the number of nodes: 6
Enter the adjacency matrix:
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 0 1
0 0 0 0 0 0
Topological Sort (DFS): 5 4 3 2 1 0
```