Write program to obtain the Topological ordering of vertices in a given digraph.

USING DFS

```
#include <stdio.h>

int n, a[10][10], res[10], s[10], top = 0;

void dfs(int j, int n, int a[][10]);
void dfs_top(int n, int a[][10]);

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

   printf("Enter the adjacency matrix:\n");
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         scanf("%d", &a[i][j]);
      }
   }

   dfs_top(n, a);

   printf("Topological Sort (Reverse Order of Finishing Time): ");
   for (int i = n - 1; i >= 0; i--) {
      printf("%d ", res[i]);
   }

   return 0;
}
```

```
void dfs_top(int n, int a[][10]) {
    for (int i = 0; i < n; i++) {
        s[i] = 0;  // Initialize visited array
    }

    for (int i = 0; i < n; i++) {
        if (s[i] == 0) {
            dfs(i, n, a);
        }
    }
}

void dfs(int j, int n, int a[][10]) {
    s[j] = 1;  // Mark current node as visited

    for (int i = 0; i < n; i++) {
        if (a[j][i] == 1 && s[i] == 0) {
            dfs(i, n, a);  // Visit all unvisited neighbors
        }
    }

    res[top++] = j;  // Push node to result stack after visiting neighbors
}
```

USING SOURCE REMOVAL METHOD

```
#include <stdio.h>

int a[10][10], n, t[10], indegree[10];
int stack[10], top = -1;

void computeIndegree(int n, int a[][10]);
void tps_SourceRemoval(int n, int a[][10]);

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

   printf("Enter the adjacency matrix:\n");
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         scanf("%d", &a[i][j]);
      }
   }

   computeIndegree(n, a);
   tps_SourceRemoval(n, a);

   printf("Topological Sort (Source Removal): ");
   for (int i = 0; i < n; i++) {
      printf("%d ", t[i]);
   }

   return 0;
}
```

```
// Compute indegree of all vertices
void computeIndegree(int n, int a[][10]) {
    int i, j, sum;
    for (i = 0; i < n; i++) {
        sum = 0;
        for (j = 0; j < n; j++) {
            sum += a[j][i];  // Count how many edges come into node i
        }
        indegree[i] = sum;
    }
}

// Perform Topological Sort using Source Removal (Kahn's Algorithm)
void tps_SourceRemoval(int n, int a[][10]) {
    int i, j, v, k = 0;

    // Push all nodes with 0 indegree into the stack
    for (i = 0; i < n; i++) {
        if (indegree[i] == 0) {
            stack[++top] = i;
        }
    }

    // While stack is not empty
    while (top != -1) {
        v = stack[top--];   // Pop a node with 0 indegree
        t[k++] = v;         // Add it to the topological order

        // For all its neighbors
        for (i = 0; i < n; i++) {
```

```
            if (a[v][i] != 0) {        // If there's an edge from v to i
                indegree[i]--;         // Reduce indegree
                if (indegree[i] == 0) { // If indegree becomes 0, push it
                    stack[++top] = i;
                }
            }
        }
    }
}
```

OUTPUT: