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

int s[100], j, res[100]; /*GLOBAL VARIABLES */

void AdjacencyMatrix(int a[][100], int n) { //To generate adjacency matrix for given
nodes

   int i, j;
   for (i = 0; i < n; i++) {
      for (j = 0; j <= n; j++) {
         a[i][j] = 0;
      }
   }
   for (i = 1; i < n; i++) {
      for (j = 0; j < i; j++) {
         a[i][j] = rand() % 2;
         a[j][i] = 0;
      }
   }
}

void dfs(int u, int n, int a[][100]) { /* DFS */

   int v;
   s[u] = 1;
   for (v = 0; v < n - 1; v++) {
      if (a[u][v] == 1 && s[v] == 0) {
         dfs(v, n, a);
      }
   }
   j += 1;
   res[j] = u;
}

void topological_order(int n, int a[][100]) { /* TO FIND TOPOLOGICAL ORDER*/

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

```
        }
    }
    return;
}
int main() {
    int a[100][100], n, i, j;

    printf("Enter number of vertices\n"); /* READ NUMBER OF VERTICES */
    scanf("%d", &n);

    AdjacencyMatrix(a, n); /*GENERATE ADJACENCY MATRIX */

    printf("\t\tAdjacency Matrix of the graph\n"); /* PRINT ADJACENCY MATRIX */
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("\t%d", a[i][j]);
        }
        printf("\n");
    }
    printf("\nTopological order:\n");

    topological_order(n, a);

    for (i = n; i >= 1; i--) {
        printf("-->%d", res[i]);
    }
    return 0;
}
```