LAB -4

Question : Write program to obtain the

1. Topological sorting using source removal method
2. Topological sorting using DFS

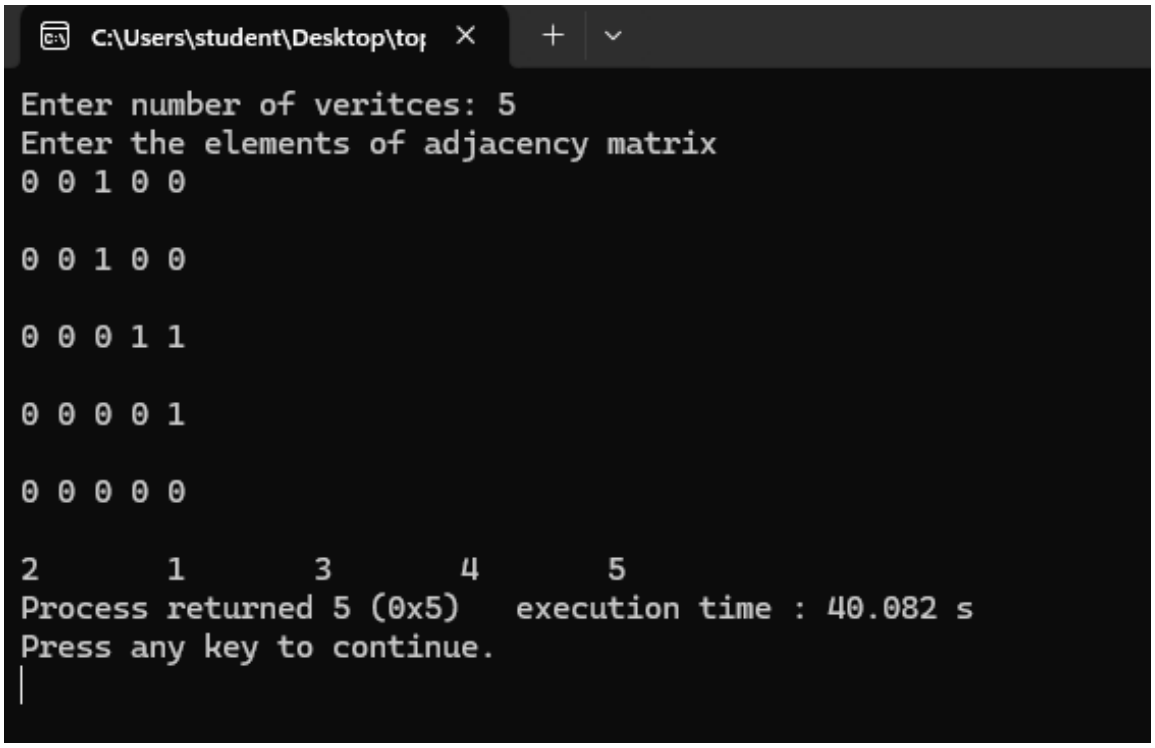
1. SOURCE CODE:

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

void topologicalSort(int **a, int n) {
  int indegree[n], s[n], top = -1, T[n], k = 0;

  for (int i = 0; i < n; i++) {
    indegree[i] = 0;
    for (int j = 0; j < n; j++) {
      indegree[i] += a[j][i];
    }
  }

  for (int i = 0; i < n; i++) {
    if (indegree[i] == 0) {
      s[++top] = i;
    }
  }

  while (top != -1) {
    int u = s[top--];
    T[k++] = u;

    for (int v = 0; v < n; v++) {
      if (a[u][v] == 1) {
        indegree[v]--;
        if (indegree[v] == 0) {
          s[++top] = v;
        }
      }
    }
```

```
    }

    if (k != n) {
        printf("Graph has a cycle. Topological sorting not possible.\n");
        return;
    }

    printf("Topological Order: ");
    for (int i = 0; i < k; i++) {
        printf("%d ", T[i]);
    }
    printf("\n");
}

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

    int **a = (int **)malloc(n * sizeof(int *));
    for (int i = 0; i < n; i++) {
        a[i] = (int *)malloc(n * sizeof(int));
    }

    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]);
        }
    }

    topologicalSort(a, n);

    for (int i = 0; i < n; i++) {
        free(a[i]);
    }
    free(a);
```

```
    return 0;
}
```

RESULT:



2. **SOURCE CODE:**

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

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

void topologicalOrder(int n, int **a) {
    int s[n];
    int res[n];
```

```
    int j = 0;

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

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

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

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

    int **a = (int **)malloc(n * sizeof(int *));
    for (int i = 0; i < n; i++) {
        a[i] = (int *)malloc(n * sizeof(int));
    }

    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]);
        }
    }

    topologicalOrder(n, a);
```

```
    for (int i = 0; i < n; i++) {
        free(a[i]);
    }
    free(a);

    return 0;
}
```

RESULT: