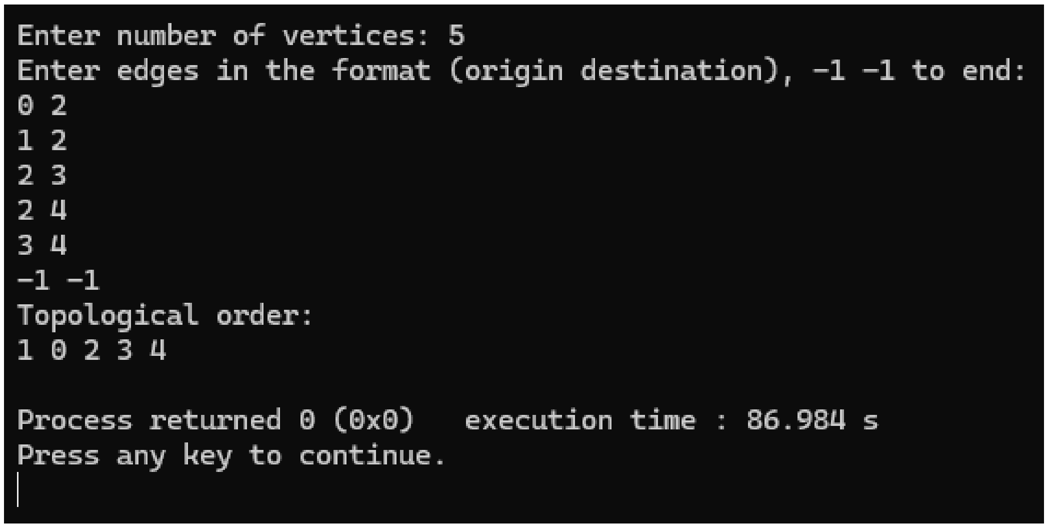
Write a program to obtain the Topological ordering of vertices in a given digraph.

Code:

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

#define MAX 100

int adj[MAX][MAX]; // Adjacency matrix
int visited[MAX]; // Visited flag for each vertex
int stack[MAX]; // Stack to store the topological order
int top = -1;
int n; // Number of vertices

void create_graph() {
    int max_edges, origin, destination;

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

    max_edges = n * (n - 1);

    printf("Enter edges in the format (origin destination), -1 -1 to end:\n");
    for (int i = 0; i < max_edges; i++) {
        scanf("%d %d", &origin, &destination);

        if (origin == -1 && destination == -1)
            break;

        if (origin < 0 || origin >= n || destination < 0 || destination >= n) {
            printf("Invalid edge!\n");
            i--;
        } else {
            adj[origin][destination] = 1;
        }
    }
}

void dfs(int v) {
    visited[v] = 1;

    for (int i = 0; i < n; i++) {
        if (adj[v][i] && !visited[i]) {
            dfs(i);
        }
    }

    stack[++top] = v; // Push vertex to stack after visiting all its neighbors
}

void topological_sort() {
    for (int i = 0; i < n; i++)
        visited[i] = 0;

    for (int i = 0; i < n; i++) {
        if (!visited[i])
            dfs(i);
    }

    printf("Topological order:\n");
    while (top != -1)
        printf("%d ", stack[top--]);
    printf("\n");
}

int main() {
    create_graph();
    topological_sort();
    return 0;
}
```

Output:



```
Enter number of vertices: 5
Enter edges in the format (origin destination), -1 -1 to end:
0 2
1 2
2 3
2 4
3 4
-1 -1
Topological order:
1 0 2 3 4

Process returned 0 (0x0)   execution time : 86.984 s
Press any key to continue.
```