WEEK 12 - Performing Topological Sorting

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

// Structure to represent a stack
struct Stack {
	int data;
	struct Stack* next;
};

struct Graph {
	int V; // No. of vertices
	struct List* adj;
};
struct List {
	int data;
	struct List* next;
};
struct Stack* createStackNode(int data)
{
	struct Stack* newNode
		= (struct Stack*)malloc(sizeof(struct Stack));
	newNode->data = data;
	newNode->next = NULL;
	return newNode;
}

struct List* createListNode(int data)
{
```

```
	struct List* newNode
		= (struct List*)malloc(sizeof(struct List));
	newNode->data = data;
	newNode->next = NULL;
	return newNode;
}

struct Graph* createGraph(int V)
{
	struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));

	graph->V = V;
	graph->adj
		= (struct List*)malloc(V * sizeof(struct List));
	for (int i = 0; i < V; ++i) {
		graph->adj[i].next = NULL;
	}
	return graph;
}

void addEdge(struct Graph* graph, int v, int w)
{
	struct List* newNode = createListNode(w);
	newNode->next = graph->adj[v].next;
	graph->adj[v].next = newNode;
}

void topologicalSortUtil(struct Graph* graph, int v,
                                                bool visited[],
                                                struct Stack** stack)
{
```

```
	visited[v] = true;

	struct List* current = graph->adj[v].next;
	while (current != NULL) {
		int adjacentVertex = current->data;
		if (!visited[adjacentVertex]) {
			topologicalSortUtil(graph, adjacentVertex,
								visited, stack);
		}
		current = current->next;
	}


	struct Stack* newNode = createStackNode(v);
	newNode->next = *stack;
	*stack = newNode;
}

void topologicalSort(struct Graph* graph)
{
	struct Stack* stack = NULL;

	bool* visited = (bool*)malloc(graph->V * sizeof(bool));
	for (int i = 0; i < graph->V; ++i) {
		visited[i] = false;
	}


	for (int i = 0; i < graph->V; ++i) {
		if (!visited[i]) { topologicalSortUtil(graph, i, visited, &stack); }
```

```
        }

        // Print contents of stack
        while (stack != NULL) {
                printf("%d ", stack->data);
                struct Stack* temp = stack;
                stack = stack->next;
                free(temp);
        }

        // Free allocated memory
        free(visited);
        free(graph->adj);
        free(graph);
}

int main()
{

        struct Graph* g = createGraph(6);
        addEdge(g, 5, 2);
        addEdge(g, 5, 0);
        addEdge(g, 4, 0);
        addEdge(g, 4, 1);
        addEdge(g, 2, 3);
        addEdge(g, 3, 1);

        printf("Topological Sorting Order: ");
        topologicalSort(g);

        return 0;
```

```
}
```

OUTPUT:-

```
Topological Sorting Order:5 4 2 3 1 0
```