**Write a C program to create a graph and display the ordering of vertices.**

**Algorithm:**

CODE:-

// C Program to implement 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

// Pointer to an array containing adjacency lists

struct List\* adj;

};

// Structure to represent a list (adjacency list)

struct List {

int data;

struct List\* next;

};

// Create a new node for the stack

struct Stack\* createStackNode(int data)

{

struct Stack\* newNode

= (struct Stack\*)malloc(sizeof(struct Stack));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

// Create a new node for the adjacency list

struct List\* createListNode(int data)

{

struct List\* newNode

= (struct List\*)malloc(sizeof(struct List));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

// Function to initialize a graph with V vertices

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;

}

// Function to add an edge to the 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;

}

// A recursive function used by topologicalSort

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;

}

// Push the current vertex to stack which stores the

// result

struct Stack\* newNode = createStackNode(v);

newNode->next = \*stack;

\*stack = newNode;

}

// The function to do Topological Sort. It uses recursive

// topologicalSortUtil

void topologicalSort(struct Graph\* graph)

{

struct Stack\* stack = NULL;

// Mark all the vertices as not visited

bool\* visited = (bool\*)malloc(graph->V \* sizeof(bool));

for (int i = 0; i < graph->V; ++i) {

visited[i] = false;

}

// Call the recursive helper function to store

// Topological Sort starting from all vertices one by

// one

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);

}

// Driver program to test above functions

int main()

{

// Create a graph given in the above diagram

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