Task 6: Depth-First Search (DFS) Recursive

Write a recursive DFS function for a given undirected graph. The function should visit every node and print it out.

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

#define MAX_VERTICES 100

// Structure representing a node in the adjacency list
struct Node {
   int dest;
   struct Node* next;
};

// Structure representing the adjacency list for each vertex
```

```
struct AdjList {
    struct Node* head;
};

// Structure representing the graph
struct Graph {
    int numVertices;
    struct AdjList* array;
};

// Function to create a new node
struct Node* createNode(int dest) {
    struct Node* newNode = (struct
Node*)malloc(sizeof(struct Node));
    newNode->dest = dest;
    newNode->next = NULL;
    return newNode;
}

// Function to create a graph with a given number of
vertices
```

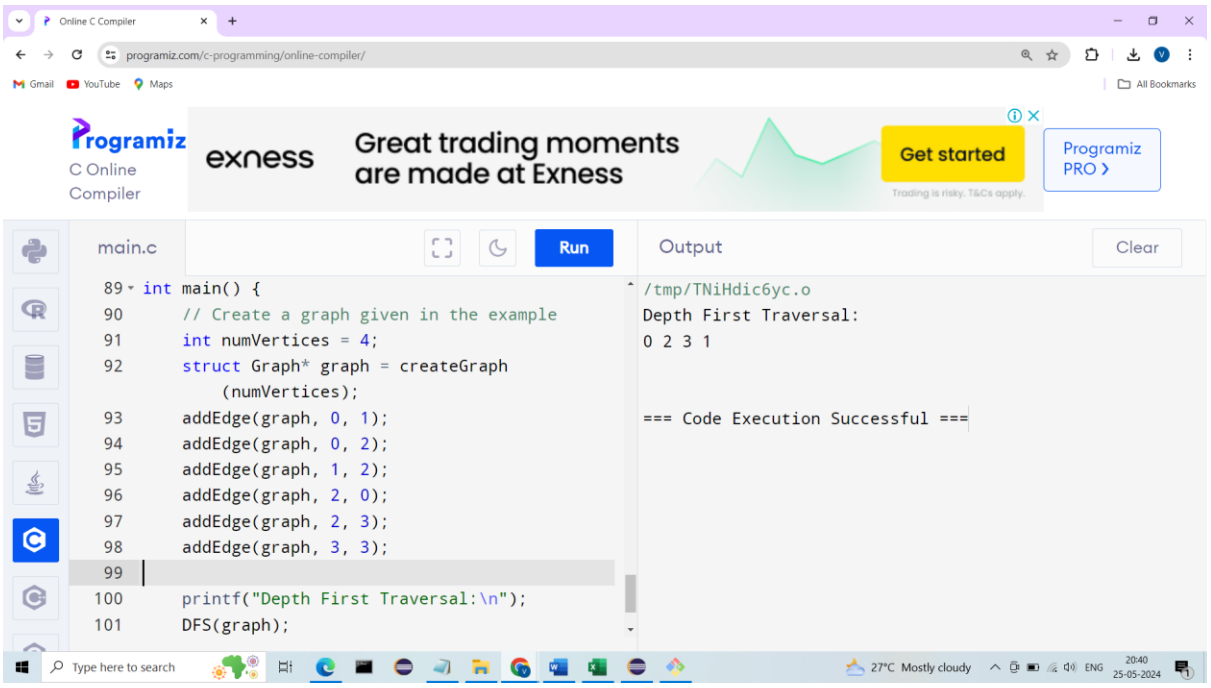
```
struct Graph* createGraph(int numVertices) {
    struct Graph* graph = (struct
Graph*)malloc(sizeof(struct Graph));
    graph->numVertices = numVertices;
    graph->array = (struct AdjList*)malloc(numVertices *
sizeof(struct AdjList));
    for (int i = 0; i < numVertices; ++i)
        graph->array[i].head = NULL;
    return graph;
}

// Function to add an edge to the graph
void addEdge(struct Graph* graph, int src, int dest) {
    // Add an edge from src to dest
    struct Node* newNode = createNode(dest);
    newNode->next = graph->array[src].head;
    graph->array[src].head = newNode;

    // Add an edge from dest to src since the graph is
undirected
    newNode = createNode(src);
```

```c
    newNode->next = graph->array[dest].head;
    graph->array[dest].head = newNode;
}

// Recursive function to perform Depth-First Search (DFS)
void DFSUtil(struct Graph* graph, int vertex, bool visited[]) {
    // Mark the current node as visited and print it
    visited[vertex] = true;
    printf("%d ", vertex);

    // Recur for all the vertices adjacent to this vertex
    struct Node* temp = graph->array[vertex].head;
    while (temp != NULL) {
        int adjVertex = temp->dest;
        if (!visited[adjVertex]) {
            DFSUtil(graph, adjVertex, visited);
        }
        temp = temp->next;
    }
```

```
}

// Function to perform Depth-First Search (DFS)
void DFS(struct Graph* graph) {
    bool* visited = (bool*)malloc(graph->numVertices *
sizeof(bool));
    for (int i = 0; i < graph->numVertices; ++i)
        visited[i] = false;

    // Call the recursive helper function to print DFS
traversal
    // starting from all vertices one by one
    for (int i = 0; i < graph->numVertices; ++i) {
        if (!visited[i]) {
            DFSUtil(graph, i, visited);
        }
    }
    printf("\n");
    free(visited);
}
```

```c
int main() {
    // Create a graph given in the example
    int numVertices = 4;
    struct Graph* graph = createGraph(numVertices);
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 2);
    addEdge(graph, 1, 2);
    addEdge(graph, 2, 0);
    addEdge(graph, 2, 3);
    addEdge(graph, 3, 3);

    printf("Depth First Traversal:\n");
    DFS(graph);

    return 0;
}
```

Output: -



```c
int main() {
    // Create a graph given in the example
    int numVertices = 4;
    struct Graph* graph = createGraph
        (numVertices);
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 2);
    addEdge(graph, 1, 2);
    addEdge(graph, 2, 0);
    addEdge(graph, 2, 3);
    addEdge(graph, 3, 3);

    printf("Depth First Traversal:\n");
    DFS(graph);
```

```
/tmp/TNiHdic6yc.o
Depth First Traversal:
0 2 3 1


=== Code Execution Successful ===
```