

# Task 5: Breadth-First Search (BFS) Implementation

For a given undirected graph, implement BFS to traverse the graph starting from a given node and print each node in the order it is visited.

```
#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);
```

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

// Function to perform Breadth-First Search (BFS)

```
void BFS(struct Graph* graph, int start) {
    bool* visited = (bool*)malloc(graph->numVertices *
sizeof(bool));
    for (int i = 0; i < graph->numVertices; ++i)
        visited[i] = false;

    // Create a queue for BFS
    int queue[MAX_VERTICES];
    int front = 0, rear = 0;

    // Mark the current node as visited and enqueue it
    visited[start] = true;
    queue[rear++] = start;

    while (front != rear) {
```

```

// Dequeue a vertex from the queue and print it
int currentVertex = queue[front++];
printf("%d ", currentVertex);

// Get all adjacent vertices of the dequeued vertex
currentVertex

// If an adjacent vertex has not been visited, then
mark it visited

// and enqueue it

struct Node* temp = graph-
>array[currentVertex].head;
while (temp != NULL) {
    int adjVertex = temp->dest;
    if (!visited[adjVertex]) {
        visited[adjVertex] = true;
        queue[rear++] = adjVertex;
    }
    temp = temp->next;
}
}

printf("\n");

```

```
}
```

```
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);
```

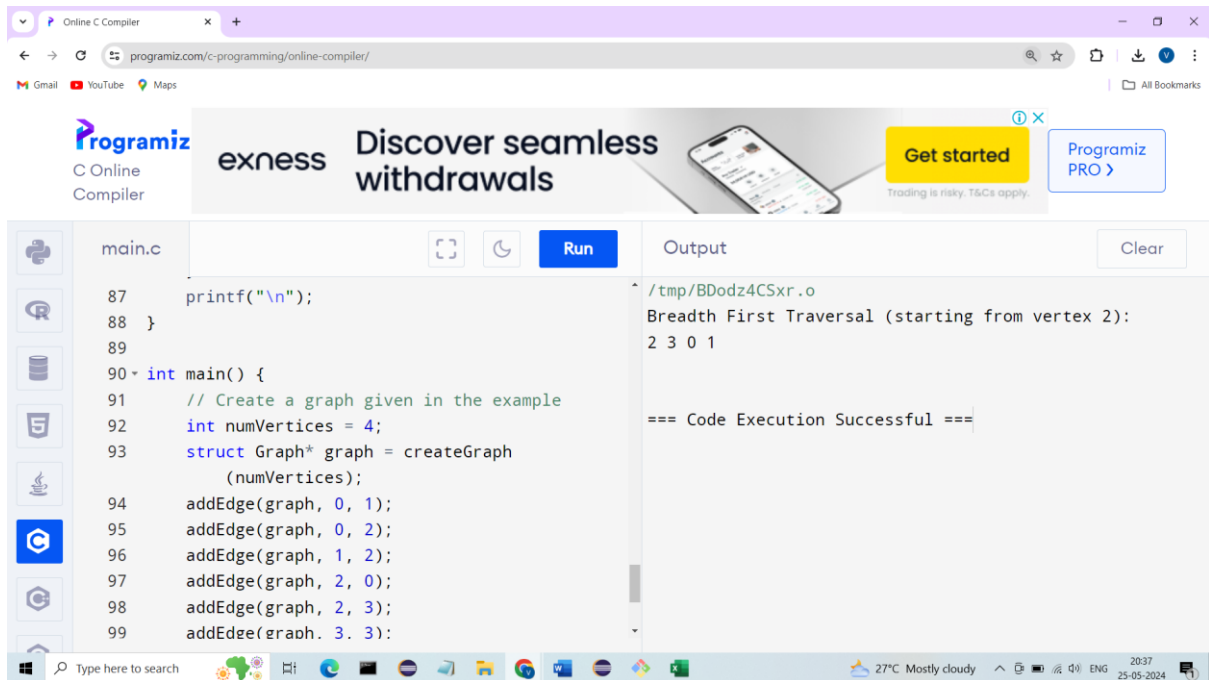
```
    printf("Breadth First Traversal (starting from vertex  
2):\n");
```

```
    BFS(graph, 2);
```

```
    return 0;
```

```
}
```

## Output: -



The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features a banner for "exness" with the text "Discover seamless withdrawals" and a "Get started" button. Below the banner, the "Programiz C Online Compiler" interface is visible. The code editor on the left contains a C program named `main.c` with the following code:

```
87 printf("\n");
88 }
89
90 int main() {
91     // Create a graph given in the example
92     int numVertices = 4;
93     struct Graph* graph = createGraph
        (numVertices);
94     addEdge(graph, 0, 1);
95     addEdge(graph, 0, 2);
96     addEdge(graph, 1, 2);
97     addEdge(graph, 2, 0);
98     addEdge(graph, 2, 3);
99     addEdge(graoh. 3. 3):
```

The "Output" panel on the right displays the execution results:

```
/tmp/BDodz4CSxr.o
Breadth First Traversal (starting from vertex 2):
2 3 0 1

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

The Windows taskbar at the bottom shows the system clock as 20:37 on 25-05-2024, with a weather forecast of 27°C Mostly cloudy.