EXP 21 BREADTH FIRST SEARCH

Program

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VERTICES 100
typedef struct {
    int items[MAX_VERTICES];
    int front, rear;
} Queue;
void initQueue(Queue *q) {
    q->front = -1;
    q->rear = -1;
}
int isEmpty(Queue *q) {
    return q->front == -1;
}
void enqueue(Queue *q, int value) {
    if (q->rear == MAX_VERTICES - 1)
        return;
    if (q->front == -1)
        q->front = 0;
    q->items[++q->rear] = value;
}
int dequeue(Queue *q) {
    if (isEmpty(q))
        return -1;
    int item = q->items[q->front];
    if (q->front == q->rear)
        q->front = q->rear = -1;
```

```
    else
        q->front++;
    return item;
}
typedef struct Node {
    int vertex;
    struct Node* next;
} Node;
typedef struct Graph {
    int numVertices;
    Node* adjLists[MAX_VERTICES];
    int visited[MAX_VERTICES];
} Graph;
Node* createNode(int v) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    newNode->vertex = v;
    newNode->next = NULL;
    return newNode;
}
Graph* createGraph(int vertices) {
    Graph* graph = (Graph*) malloc(sizeof(Graph));
    graph->numVertices = vertices;
    for (int i = 0; i < vertices; i++) {
        graph->adjLists[i] = NULL;
        graph->visited[i] = 0;
    }
    return graph;
}
void addEdge(Graph* graph, int src, int dest) {
```

```
    Node* newNode = createNode(dest);
    newNode->next = graph->adjLists[src];
    graph->adjLists[src] = newNode;
    newNode = createNode(src);
    newNode->next = graph->adjLists[dest];
    graph->adjLists[dest] = newNode;
}
void bfs(Graph* graph, int startVertex) {
    Queue q;
    initQueue(&q);

    graph->visited[startVertex] = 1;
    enqueue(&q, startVertex);
    printf("BFS traversal starting from vertex %d:\n", startVertex);
    while (!isEmpty(&q)) {
        int currentVertex = dequeue(&q);
        printf("%d ", currentVertex);
        Node* temp = graph->adjLists[currentVertex];
        while (temp) {
            int adjVertex = temp->vertex;
            if (graph->visited[adjVertex] == 0) {
                graph->visited[adjVertex] = 1;
                enqueue(&q, adjVertex);
            }
            temp = temp->next;
        }
    }
    printf("\n");
}
```

```
int main() {
int vertices = 6;
Graph* graph = createGraph(vertices);
addEdge(graph, 0, 1);
addEdge(graph, 0, 2);
addEdge(graph, 1, 3);
addEdge(graph, 1, 4);
addEdge(graph, 2, 4);
addEdge(graph, 3, 5);
addEdge(graph, 4, 5);
bfs(graph, 0);
return 0;
}
```

Output