#include <stdio.h>

#include <stdlib.h>

#define MAX 100

// Graph structure

typedef struct Node {

int vertex;

struct Node\* next;

} Node;

Node\* adjList[MAX];

int visited[MAX];

int queue[MAX], front = 0, rear = 0;

// Queue operations

void enqueue(int v) {

queue[rear++] = v;

}

int dequeue() {

return queue[front++];

}

int isEmpty() {

return front == rear;

}

// Add edge to adjacency list

void addEdge(int src, int dest) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->vertex = dest;

newNode->next = adjList[src];

adjList[src] = newNode;

}

// BFS traversal

void bfs(int startVertex, int vertices) {

for (int i = 0; i < vertices; i++)

visited[i] = 0;

enqueue(startVertex);

visited[startVertex] = 1;

printf("BFS Traversal: ");

while (!isEmpty()) {

int current = dequeue();

printf("%d ", current);

Node\* temp = adjList[current];

while (temp) {

int adj = temp->vertex;

if (!visited[adj]) {

enqueue(adj);

visited[adj] = 1;

}

temp = temp->next;

}

}

printf("\n");

}

// Sample usage

int main() {

int vertices = 6;

// Initialize adjacency list

for (int i = 0; i < vertices; i++)

adjList[i] = NULL;

// Sample graph edges

addEdge(0, 1);

addEdge(0, 2);

addEdge(1, 3);

addEdge(1, 4);

addEdge(2, 4);

addEdge(3, 5);

addEdge(4, 5);

bfs(0, vertices);

return 0;

}