**Name : Tanishq Thuse**

**Branch : CS(AI)**

**Year : SY**

**Div : B**

**Roll no. : 60**

**PRN : 12310237**

**Title : AI Assignment-2 : Implement Uninformed Search Algorithms**

**Q1) Implementation of BFS.**

**Code :**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#define MAX\_VERTICES 100

struct Queue {

int items[MAX\_VERTICES];

int front;

int rear;

};

int graph[MAX\_VERTICES][MAX\_VERTICES];

int visited[MAX\_VERTICES];

int vertices, edges;

struct Queue\* createQueue() {

struct Queue\* q = (struct Queue\*)malloc(sizeof(struct Queue));

q->front = -1;

q->rear = -1;

return q;

}

void enqueue(struct Queue\* q, int value) {

if (q->rear == MAX\_VERTICES - 1)

printf("Queue is full");

else {

if (q->front == -1)

q->front = 0;

q->rear++;

q->items[q->rear] = value;

}

}

int dequeue(struct Queue\* q) {

int item;

if (q->front == -1)

printf("Queue is empty");

else {

item = q->items[q->front];

q->front++;

if (q->front > q->rear) {

q->front = q->rear = -1;

}

return item;

}

return -1;

}

// BFS algorithm

void bfs(int startVertex) {

struct Queue\* q = createQueue();

visited[startVertex] = 1;

enqueue(q, startVertex);

printf("Breadth-First Traversal starting from vertex %d: ", startVertex);

while (q->front != -1) {

int currentVertex = dequeue(q);

printf("%d ", currentVertex);

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

if (graph[currentVertex][i] == 1 && !visited[i]) {

visited[i] = 1;

enqueue(q, i);

}

}

}

}

int main() {

printf("Enter the number of vertices: ");

scanf("%d", &vertices);

printf("Enter the number of edges: ");

scanf("%d", &edges);

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

visited[i] = 0;

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

graph[i][j] = 0;

}

printf("Enter the edges (format: vertex1 vertex2):\n");

for (int i = 0; i < edges; i++) {

int vertex1, vertex2;

scanf("%d %d", &vertex1, &vertex2);

graph[vertex1][vertex2] = 1;

graph[vertex2][vertex1] = 1;

}

int startVertex;

printf("Enter the starting vertex for BFS: ");

scanf("%d", &startVertex);

bfs(startVertex);

printf("\n");

return 0;

/\* Visualization

0

/ \

1 2

/ \

3 4

\*/

/\*\*Example run :

\* Enter the number of vertices: 5

Enter the number of edges: 4

Enter the edges (format: vertex1 vertex2):

0 1

0 2

1 3

1 4

Enter the starting vertex for BFS: 0

\*/

}

**Output** :

