**Practical-12**

**AIM:** To implement Breadth First Search in C language

**SOFTWARE REQUIRED:** Vs Code

**PSEUDO CODE:**

function BFS(graph, startVertex):

// Create a queue for BFS

queue = empty queue data structure

visited = array of boolean values, initialized to false

// Mark the start vertex as visited and enqueue it

visited[startVertex] = true

enqueue(queue, startVertex)

// Loop until the queue is empty

while queue is not empty:

// Dequeue a vertex from the queue

currentVertex = dequeue(queue)

print currentVertex

// Explore all adjacent vertices of the current vertex

for each neighbor in getNeighbors(currentVertex):

if not visited[neighbor]:

visited[neighbor] = true

enqueue(queue, neighbor)

function getNeighbors(vertex):

// Helper function to return the neighbors of a vertex

neighbors = empty list

for each adjacentVertex in graph[vertex]:

if adjacentVertex is connected and not visited[adjacentVertex]:

add adjacentVertex to neighbors

return neighbors

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_VERTICES 100

struct Queue {

    int items[MAX\_VERTICES];

    int front;

    int rear;

};

void initializeQueue(struct Queue \*q) {

    q->front = -1;

    q->rear = -1;

}

int isEmpty(struct Queue \*q) {

    return q->front == -1;

}

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

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

        printf("Queue is full\n");

    } else {

        if (q->front == -1) {

            q->front = 0;

        }

        q->rear++;

        q->items[q->rear] = value;

    }

}

int dequeue(struct Queue \*q) {

    int item;

    if (isEmpty(q)) {

        printf("Queue is empty\n");

        item = -1;

    } else {

        item = q->items[q->front];

        q->front++;

        if (q->front > q->rear) {

            q->front = q->rear = -1;

        }

    }

    return item;

}

void BFS(int adjacencyMatrix[MAX\_VERTICES][MAX\_VERTICES], int vertices, int startVertex) {

    int visited[MAX\_VERTICES] = {0};

    struct Queue q;

    initializeQueue(&q);

    visited[startVertex] = 1;

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

    printf("%d ", startVertex);

    enqueue(&q, startVertex);

    while (!isEmpty(&q)) {

        int currentVertex = dequeue(&q);

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

            if (adjacencyMatrix[currentVertex][i] && !visited[i]) {

                visited[i] = 1;

                printf("%d ", i);

                enqueue(&q, i);

            }

        }

    }

}

int main() {

    int vertices, startVertex;

    printf("Ananta Walli, A2305221322");

    printf("\nEnter the number of vertices: ");

    scanf("%d", &vertices);

    int adjacencyMatrix[MAX\_VERTICES][MAX\_VERTICES];

    printf("Enter the adjacency matrix:\n");

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

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

            scanf("%d", &adjacencyMatrix[i][j]);

        }

    }

    printf("Enter the starting vertex: ");

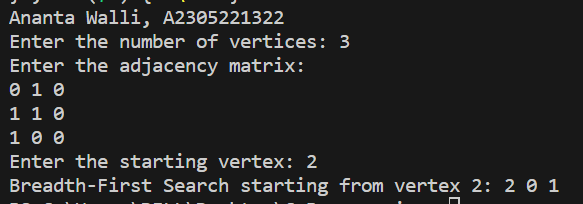
    scanf("%d", &startVertex);

    BFS(adjacencyMatrix, vertices, startVertex);

    return 0;

}

**OUTPUT:**



**RESULT:** The above code implements the Breadth First Search in C programming.