22AIE203 – Data Structures and
Algorithm - 2

LAB EXP 1a
BFS

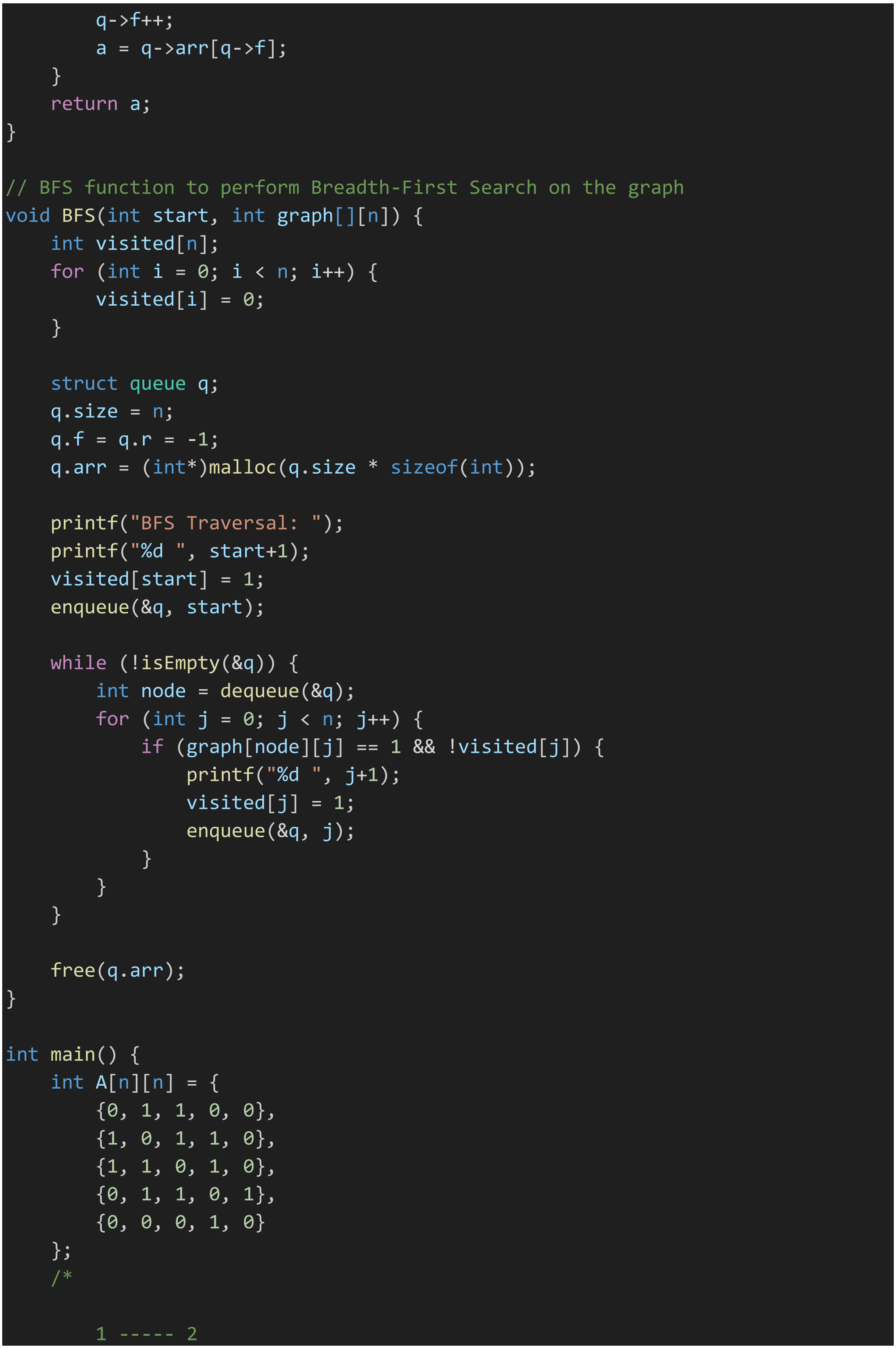
NAME :GURUPRASATH M R

ROLLNO:CH.EN.U4.AIE22015

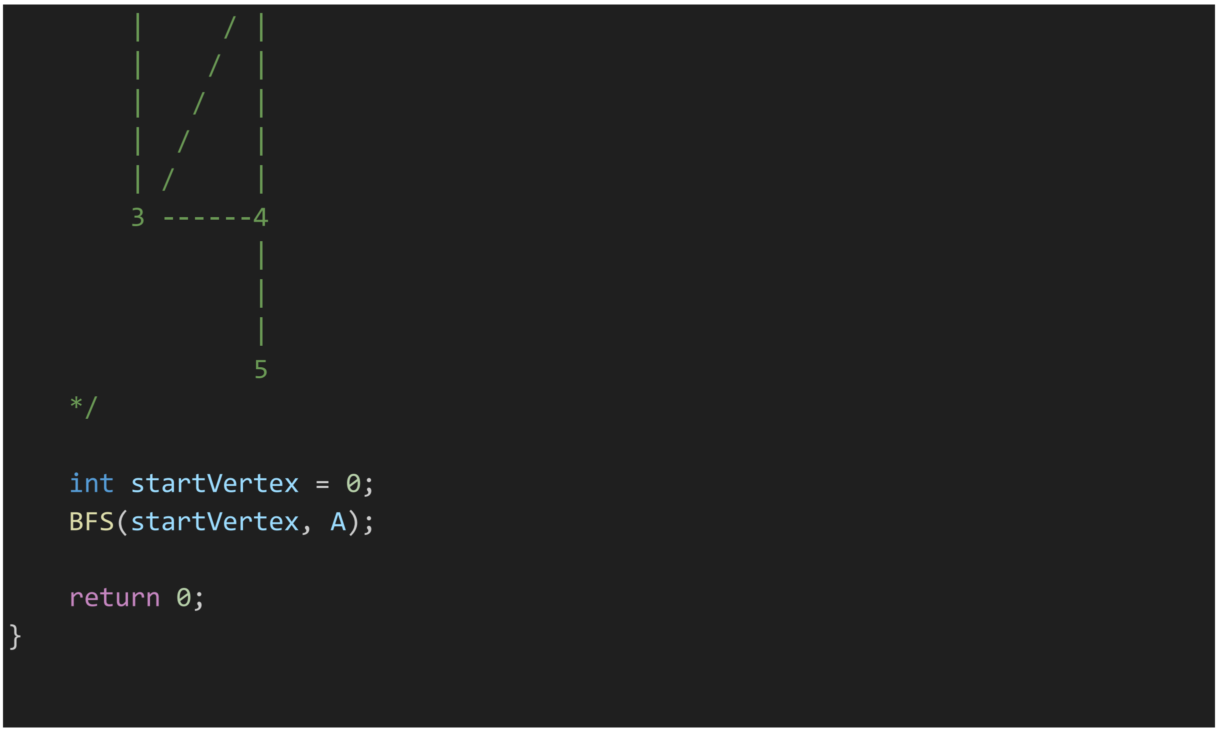
CLASS:2 YEAR 3 SEM

CODE

```
#include <stdio.h>
#include <stdlib.h>

#define n 5

struct queue {
    int size;
    int f;
    int r;
    int* arr;
};



// Function to check if the queue is empty
int isEmpty(struct queue* q) {
    if (q->r == q->f) {
        return 1;
    }
    return 0;
}

// Function to check if the queue is full
int isFull(struct queue* q) {
    if (q->r == q->size - 1) {
        return 1;
    }
    return 0;
}

// Function to enqueue an element into the queue
void enqueue(struct queue* q, int val) {
    if (isFull(q)) {
        printf("This Queue is full\n");
    } else {
        q->r++;
        q->arr[q->r] = val;
    }
}

// Function to dequeue an element from the queue
int dequeue(struct queue* q) {
    int a = -1;
    if (isEmpty(q)) {
        printf("This Queue is empty\n");
    } else {
```



```
        q->f++;
        a = q->arr[q->f];
    }
    return a;
}

// BFS function to perform Breadth-First Search on the graph
void BFS(int start, int graph[][n]) {
    int visited[n];
    for (int i = 0; i < n; i++) {
        visited[i] = 0;
    }

    struct queue q;
    q.size = n;
    q.f = q.r = -1;
    q.arr = (int*)malloc(q.size * sizeof(int));

    printf("BFS Traversal: ");
    printf("%d ", start+1);
    visited[start] = 1;
    enqueue(&q, start);

    while (!isEmpty(&q)) {
        int node = dequeue(&q);
        for (int j = 0; j < n; j++) {
            if (graph[node][j] == 1 && !visited[j]) {
                printf("%d ", j+1);
                visited[j] = 1;
                enqueue(&q, j);
            }
        }
    }

    free(q.arr);
}

int main() {
    int A[n][n] = {
        {0, 1, 1, 0, 0},
        {1, 0, 1, 1, 0},
        {1, 1, 0, 1, 0},
        {0, 1, 1, 0, 1},
        {0, 0, 0, 1, 0}
    };
    /*

        1 ----- 2
```

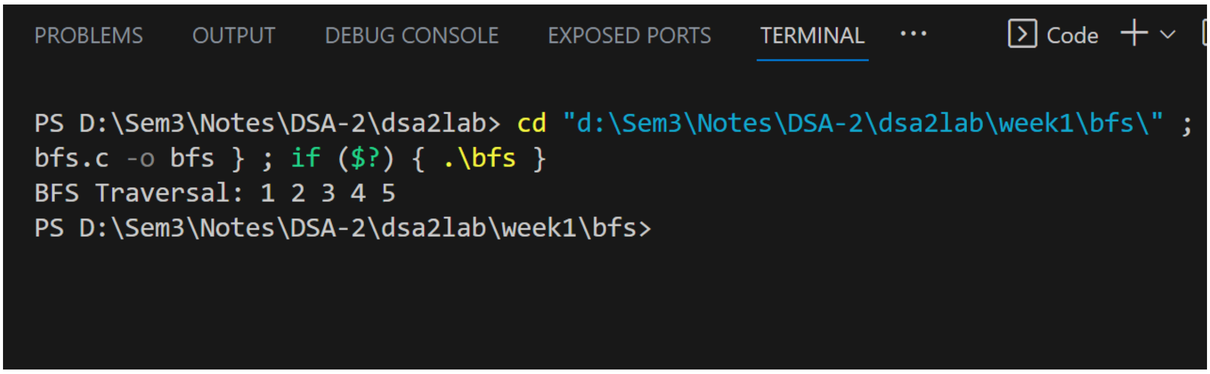


```
        |     / |
        |    /  |
        |   /   |
        |  /    |
        | /     |
        3 ------4
                |
                |
                |
                5
    */

    int startVertex = 0;
    BFS(startVertex, A);

    return 0;
}
```

Input

```
1 ----- 2
|     / |
|    /  |
|   /   |
|  /    |
| /     |
3 ------4
        |
        |
        |
        5
```

Output



```
PS D:\Sem3\Notes\DSA-2\dsa2lab> cd "d:\Sem3\Notes\DSA-2\dsa2lab\week1\bfs\" ;
bfs.c -o bfs } ; if ($?) { .\bfs }
BFS Traversal: 1 2 3 4 5
PS D:\Sem3\Notes\DSA-2\dsa2lab\week1\bfs>
```