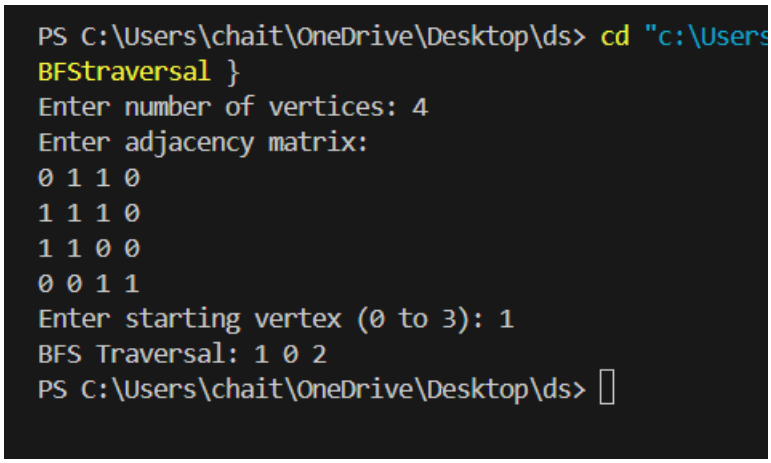
9A. a) Write a program to traverse a graph using BFS method.

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

#define MAX 100

int queue[MAX], front = -1, rear = -1;

// Function to add an element to the queue
void enqueue(int vertex) {
    if (rear == MAX - 1) {
        printf("Queue is full\n");
        return;
    }
    if (front == -1) front = 0;
    queue[++rear] = vertex;
}

// Function to remove an element from the queue
int dequeue() {
    if (front == -1 || front > rear) {
        return -1; // Queue is empty
    }
    return queue[front++];
}

// Function to check if queue is empty
```

```
int isEmpty() {
    return front == -1 || front > rear;
}

// BFS function
void BFS(int graph[MAX][MAX], int n, int start) {
    int visited[MAX] = {0};
    enqueue(start);
    visited[start] = 1;

    printf("BFS Traversal: ");

    while (!isEmpty()) {
        int vertex = dequeue();
        printf("%d ", vertex);

        for (int i = 0; i < n; i++) {
            if (graph[vertex][i] == 1 && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
    printf("\n");
}

int main() {
    int n, start;
```

```
    int graph[MAX][MAX];

    printf("Enter number of vertices: ");
    scanf("%d", &n);

    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);

    printf("Enter starting vertex (0 to %d): ", n-1);
    scanf("%d", &start);

    BFS(graph, n, start);

    return 0;
}
```

OUTPUT:


PS C:\Users\chait\OneDrive\Desktop\ds> cd "c:\Users
BFStraversal }
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 1 1 0
1 1 0 0
0 0 1 1
Enter starting vertex (0 to 3): 1
BFS Traversal: 1 0 2
PS C:\Users\chait\OneDrive\Desktop\ds>