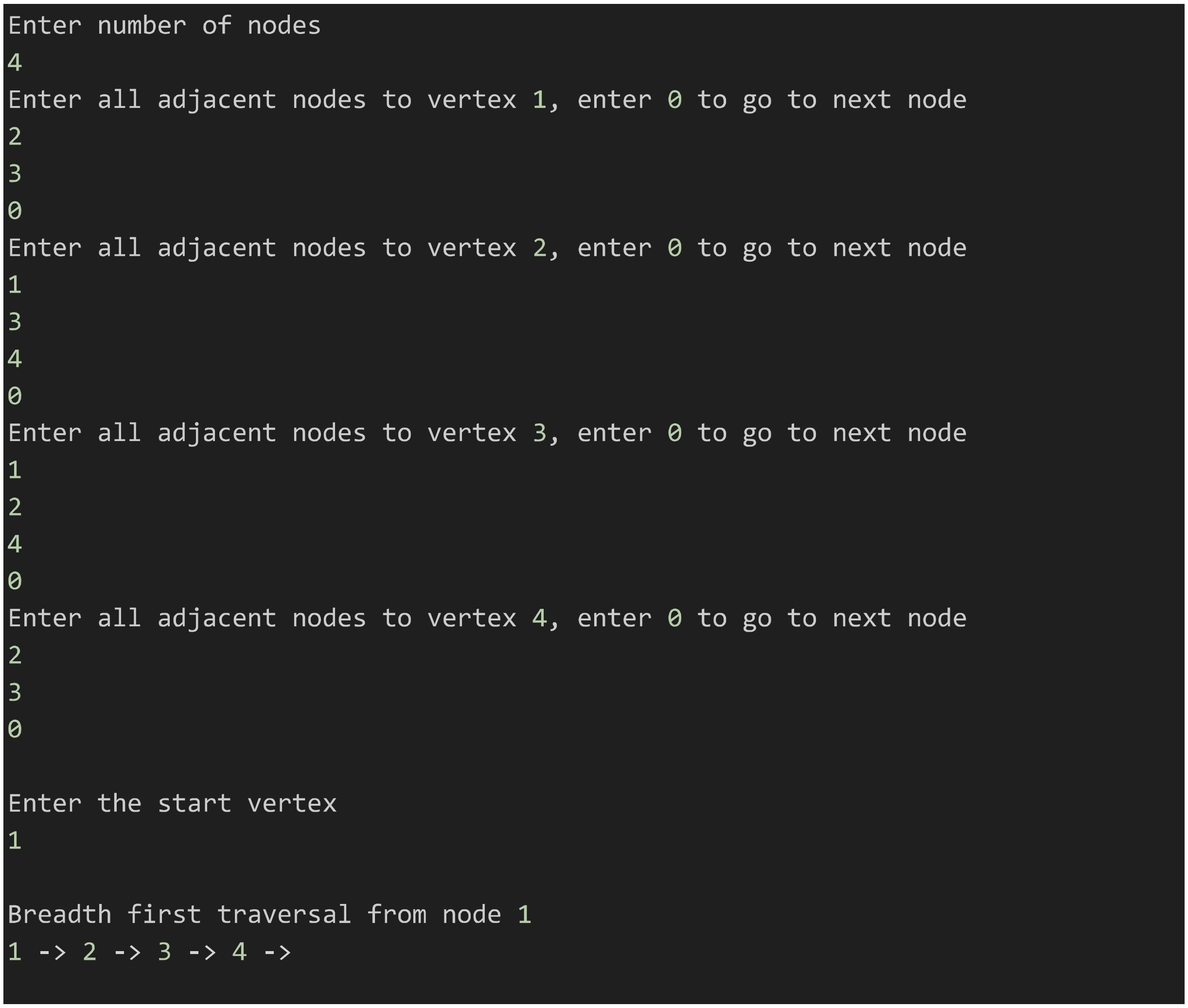
```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node *next;
};

struct Node *vertices[100], *temp, *head;

const int qSize = 100;
int q[100];
int front = -1, rear = -1;

void qPush(int);
int qPop();
bool qEmpty();


void breadthFirstTraversal(int u)
{
    int visited[100] = {0};
    visited[u] = 1;
    qPush(u);
    int curr_node;
    while (!qEmpty())
    {
        curr_node = qPop();
        printf("%d -> ", curr_node);
        temp = vertices[curr_node];
        while (temp) // check the neighbouring nodes
        {
            if (!visited[temp->data])
            {
                visited[temp->data] = 1;
                qPush(temp->data);
            }
            temp = temp->next;
        }
    }
}

int main()
{
    int n;
    printf("Enter number of nodes\n");
    scanf("%d", &n);
    int i, c;
    for (i = 1; i <= n; i++)
    {
        vertices[i] = (struct Node *)malloc(sizeof(struct Node));
        vertices[i]->data = i;
        vertices[i]->next = NULL;

        head = vertices[i];

        printf("Enter all adjacent nodes to vertex %d, enter 0 to go to next node\n", i);

        do

        {

            scanf("%d", &c);

            if (c != 0)

            {

                temp = (struct Node *)malloc(sizeof(struct Node));

                temp->data = c;

                temp->next = NULL;

                head->next = temp;

                head = temp;

            }


        } while (c != 0);

    }


    int startNode;

    printf("\nEnter the start vertex\n");

    scanf("%d", &startNode);

    printf("\nBreadth first traversal from node %d\n", startNode);

    breadthFirstTraversal(startNode);


    printf("\n");

    return 0;

}
```

```
void qPush(int data)

{

    if (rear == qSize - 1)

        return;

    if (front == -1)

        ++front;

    q[++rear] = data;

}


int qPop()

{

    if (qEmpty())

        return -1;

    int deleted = q[front];

    if (rear == front)

        front = rear = -1;

    else

        front++;

    return deleted;

}


bool qEmpty()

{

    return front == -1 && rear == -1;

}
```



```
Enter number of nodes
4
Enter all adjacent nodes to vertex 1, enter 0 to go to next node
2
3
0
Enter all adjacent nodes to vertex 2, enter 0 to go to next node
1
3
4
0
Enter all adjacent nodes to vertex 3, enter 0 to go to next node
1
2
4
0
Enter all adjacent nodes to vertex 4, enter 0 to go to next node
2
3
0

Enter the start vertex
1

Breadth first traversal from node 1
1 -> 2 -> 3 -> 4 ->
```