```c

/**
 * Implementation of Breadth First Search
 * Author: Mithusayel Murmu
 */

#include <stdio.h>

#define GRAPH_SZ 50
typedef enum { FALSE, TRUE } BOOL;

/** Rudimentary Queue implementation */
typedef struct _Queue Queue;
struct _Queue {
    int size, front, rear;
    int data[GRAPH_SZ];
};

void queue_init(Queue *que) {
    que->size = que->front = 0;
    que->rear = -1;
}

BOOL queue_enque(Queue *que, int val) {
    if (que->size == GRAPH_SZ)
        return FALSE;

    que->data[++que->rear] = val;
    que->size++;
    return TRUE;
}

int queue_deque(Queue *que) {
    if (que->size == 0)
        return 0;

    que->size--;
    int ret = que->data[que->front++];
    if (que->size == 0) queue_init(que);
    return ret;
}

/** Rudimentary Graph implementation */
typedef struct _Graph Graph;
typedef enum { WHITE, GRAY, BLACK } NodeState;

struct _Graph {
    int size;
    int data[GRAPH_SZ][GRAPH_SZ+1];
    NodeState nstate[GRAPH_SZ];
};

void graph_init(Graph *graph) {
    int i;
    for (i = 0; i < GRAPH_SZ; ++i) {
        graph->data[i][0] = 0;
        graph->nstate[i] = WHITE;  // Undiscovered
    }
}

#define _scand(n) scanf("%d", &(n))
void graph_input(Graph *graph) {
    int vs, asz, vi, i, j;

    _scand(vs);  // Number of vertices
    graph->size = vs;
    for (i = 0; i < vs; ++i) {
        _scand(asz);  // Adjacency list size
        graph->data[i][0] = asz;
        for (j = 1; j <= asz; ++j) {
            _scand(vi);
            graph->data[i][j] = vi;
        }
    }
}

void graph_bfs_visit(Graph *graph, int _vi, void (*callback)(int)) {
    int vi, asz, i, avi; Queue queue;
    queue_init(&queue);

    queue_enque(&queue, _vi); graph->nstate[_vi] = GRAY;
    while (queue.size) {
        vi = queue_deque(&queue);
        // Enqueue adjacent vertices
        if (graph->nstate[vi] == GRAY) {
            // Get adjacency list size
            asz = graph->data[vi][0];
            for (i = 1; i <= asz; i++) {
                avi = graph->data[vi][i];
                if (graph->nstate[avi] == WHITE) {
                    queue_enque(&queue, avi);
                    graph->nstate[avi] = GRAY;
                }
            }
        }

        // Done visiting this node
        graph->nstate[vi] = BLACK; callback(vi);
    }
}

void graph_bfs(Graph *graph, void (*callback)(int)) {
    if (graph == NULL || graph->size == 0)
        return;

    int i;
    // Iterate through vertices in the forest
    for (i = 0; i < graph->size; ++i)
        if (graph->nstate[i] == WHITE)
            graph_bfs_visit(graph, i, callback);

    // Reset vertex states
    for (i = 0; i < graph->size; ++i)
        graph->nstate[i] = WHITE;
}

static void print_utility(int n) { printf("%d ", n); }

/** Driver function */
int main(int argc, char const *argv[]) {
    Graph graph; graph_init(&graph);

    printf("Enter graph data:\n");
    graph_input(&graph);

    printf("\nBFS result:\n");
    graph_bfs(&graph, print_utility);
    printf("\n");

    return 0;
}

```