```c

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

#include <stdio.h>

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

/** 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_dfs_visit(Graph *graph, int vi, void (*callback)(int)) {
    int asz, i, avi;

    graph->nstate[vi] = GRAY;
    callback(vi); asz = graph->data[vi][0];
    for (i = 1; i <= asz; i++) {
        avi = graph->data[vi][i];
        if (graph->nstate[avi] == WHITE)
            graph_dfs_visit(graph, avi, callback);
    }

    graph->nstate[vi] = BLACK;
}

void graph_dfs(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_dfs_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);
```

```
80
81      printf("Enter graph data:\n");
82      graph_input(&graph);
83
84      printf("\nDFS result:\n");
85      graph_dfs(&graph, print_utility);
86      printf("\n");
87
88      return 0;
89  }
90
```