Lab 15: Depth-first search and breath-first search

In this lab, we will implement depth-first search and breath-first search for finding a given element. When a graph structure (i.e. a set of nodes and edges) is given, your program constructs a graph by using adjacency list. Subsequently, your program traverses the graph by using both depth-first search and breath-first search. For depth-first search, you should implement two different approaches, the recursive approach and the iterative approach using stack. For breath-first search, you can use queue as we discussed in class. When you mark any node as visited, your program should print the node. At the end of the traversal, your program should print whether the given number is in the graph or not.
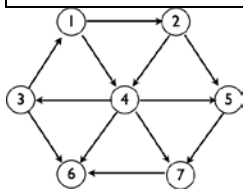
1. Input
Read a set of vertices from the first line and a set of edges from the second line of the given input file. Each line is described below. You may assume that your node is represented by any integer.

• Vertices are given in the first line. Each vertex is separated by a space.
• Edges are given in the second line. Each edge is represented by a pair of vertices and its weight. For example, "1-3-4" represents an edge from the vertex 1 to 3 with the weight value of 4.
• The number you are supposed to search is in the third line.

An exemplary input file is given below with the corresponding graph for your reference.
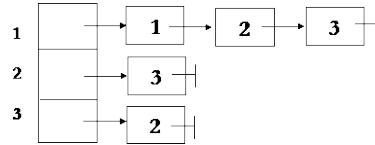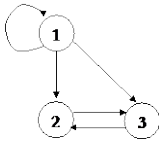
Input.txt

```
1 2 3 4 5 6 7
1-2  1-4  2-5  2-4  3-1  3-6  4-3  4-6  4-7
4-5  5-7  7-6
```


2. Data structure

You can use an adjacency list to store your graph information as we discussed in class. An example is shown below.

3. Program description
- name : p14.c
- input : an input file name is given as a command line argument. See the example in "1. input".
- output : the shortest path (for the given graph in input) in the standard output

Submit to the course website (https://portal.hanyang.ac.kr) your source code and a written report. Your report should include the description of your own implementation.