Graph Traversal

1. Introduction

In this lab we are going to be traversing graphs. One way you will traverse the graph is with breadth first search (BFS) and the other is depth first search (DFS). The BFS is similar to the level order traversal you did on trees and will use a queue. The depth first search is like a pre-order traversal and will use recursion. You can do DFS with a stack for fun too if you like. In fact, DFS and BFS are exactly the same, but with different data structures, stacks (or recursion) and queues.

2. Implementation

The graph is in a class with the expected public functions. It is stored as a two dimensional array of bools (adjacency matrix). The number of nodes in the graph is a global called `NODE_COUNT`. Your task is to write two functions.

```
void Graph::BFS();
void Graph::DFS();
```

When you visit a node you should print it to the screen and you should mark it as visited. To mark it as visited you can use the public function in the graph class. Also, you will unmark all vertices before beginning, it is also implemented for you in a public function.

3. BFS - pseudocode

```
unmark all,
choose x
mark and process (print) x add x to the Q
while Q not empty
      remove vertex u from Q
      for all unmarked neighbors w of u
            mark and process (print) w
            insert w into Q
```

4. DFS - pseudocode

```
unmark all vertices

choose x
mark and process (print) x

for each unmarked neighbor of x, w
      recurse on w
```

5. Some queue functions

Ex:

```
#include <queue> //need this in with the other includes

queue<int> q1;
q1.push(14);     //push on something
int x;
x = q1.front();  //copy top
q1.pop();        //delete the top
```

Function	Description
`void pop();`	Deletes front of the queue.
`bool empty() const;`	Return true if the queue is empty.
`TYPE& front();`	Returns the front of the queue.
`void push(const TYPE& val)`	Puts val of type TYPE at the back of the queue.
`size_type size() const;`	Returns the size of the queue

Getting the files:

First make a directory called lab3 and change into that directory:

"**mkdir lab10 && cd lab10**"

At the command prompt, enter (don't forget the space and period at the end):

cp /comp/15/public_html/labs/lab10/graphs/* .

1. Open Eclipse from the desktop menu (Applications->Programming->Eclipse)

2. Use the following steps to set up your project (you'll get used to the steps quickly, steps A-C only need to be done once at the beginning of the course—if you already did this for homework 0, you don't need to do A-C again):

A. Default location for your workspace is fine
B. Click on "Workbench", then click "Window->Open Perspective->Other", and choose C++
(note: if C++ is not listed, close Eclipse and re-open by typing Eclipse)
C. Window->Preferences
General->Editors->Text Editors
Displayed tab width (8) (recommended)
Show print margin: 80 col
Show line numbers
C/C++ -> Code Style
Select K&R [built-in] and then "Edit", then rename to "K&R 8-tab"
Change Tab size to 8 (recommended)
Click "Apply" then "Ok"
General->Workspace
Check "Save automatically before build"

D. File->New C++ Project
Fill in project name: lab10_project

Use default location should be checked.
Select Empty Project->Linux GCC
Click Next
Click "Advanced Settings"
1. On the left, under C/C++ General->Paths and Symbols (left hand side)
(you may have to click on the triangle next to C/C++ General)
Select "[Debug]" at the top for Configuration
Source Location Tab
Click "Link Folder"
Check "Link to folder in the file system"
Browse to find your folder.
(should be h/your_username/lab10)
Click OK
Click Apply

2. C++ Build->Settings (may have to click on the triangle)
For Configuration (at the top), select [All configurations]
GCC C++ Compiler: Command should be "clang++" (no quotes)
GCC C Compiler: Command: clang
GCC C++ Linker: Command: clang++
Click "Apply"
Click OK

Click Finish

E. Click on the white triangle next to your project name.
The folder you linked to should be listed. Click on its
triangle (“lab10”)
The files in that folder should be listed.

F. Test the build by clicking on the hammer in the icon bar.
You should see some compiling messages in the Console window
at the bottom. (Things like, "Building file...; clang++, etc.)

G. The program will compile, but will have some warnings. You need to write the functions described above.

Compiling and Running:

1. At the command prompt, enter “**`make`**” and press enter or return to compile.
2. At the command prompt, enter “**`./graphTraversal`**” and press enter or return to run the new program.

Providing:

At the command prompt, enter “**`make provide`**” and press enter or return.