Project 1

100 points
DUE DATE: March 2nd by 9:00pm.

Homework 2 Deliverables via Handin:

- Stack.h
- Terminal.cpp

Remember you will be submitting only these 2 files via Handin.
Project submissions will always be through Handin.

This is not a team work, do not copy somebody else's work.

Problem 1

Your job is to implement a stack. You will be given a header file, Stack.h with all of the method declarations. Fill in the definitions for all of the methods described below:

- **Stack(int size)** - The constructor for the stack has a size argument for the maximum number of elements in the stack.
- **void push(T value)** - The push method adds a new element to the top of the stack.
- **T pop()** - The pop method removed the top element of the stack and returns the element removed
- **T top() -** The top method returns the top element of the stack, but does not remove it.
- **bool isEmpty()** - returns true if the stack is empty, false otherwise.
- **bool isFull()** - returns true if the number of elements in the stack is equal to the maximum number of elements allowed.
- **void clear()** - empties the stack. After clearing, isEmpty should return true.

Problem 2

Your job is to use your stack implementation to emulate file navigation in the linux terminal. A brief description of cd can be found here:
https://en.wikipedia.org/wiki/Cd_(command)
You will be provided Terminal.h which contains all method declarations. The Terminal class has the following methods:

- **Terminal(string homeDir)** - The constructor for the Terminal class takes a home directory as an argument. This will be the home directory that will be used when navigating to ~. Just like in the terminal, ".." will move up a directory and "../.." will move up two directories, etc.
- **void cd(string path)** - This will move the working directory to the path passed in. Some examples of how this method works are as follows:

Sample Run:
Terminal t("/user/cse331");
t.cd("Documents"); // working directory is now /user/cse331/Documents"
t.cd(".."); //working directory is now /user/cse331
t.cd("/etc"); // working directory is now /etc
t.cd("."); //working directory is still /etc
t.cd(""); // working directory is set to home if empty string is passed in, (/user/cse331)
t.cd("~"); //also changes working directory to home directory

Notes:
- If the path string begins with "/", the stack needs to be cleared and the working directory is equal to the path passed in.
- If the path string does not start with "/", this denotes a subdirectory of the current directory and the path provided is appended to the working directory.

- **string pwd()** - this method simply returnsthe working directory as a string. For example, if the pathDir Stack looks like {"user", "cse331", "Documents" }, pwd() will return "/user/cse331/Documents".
- **void pushd(string path)** - pushd will push a new directory on to the navigation stack. For a description of this, consult https://en.wikipedia.org/wiki/Pushd_and_popd
  An example of how pushd works is below:
  Terminal t("/user/cse331");
  t.pwd(); // returns "/user/cse331"
  t.pushd("/var/www");
  t.pwd(); // returns "/var/www"

t.popd();
t.pwd(); // returns “/user/cse331”

- **void popd()** - popd does the opposite of pushd. It will pop off the last directory on the navigation stack. An example of popd is above.
- **vector<string> string splitPath(string path)** - this is a private helper method that takes a path string like “/user/cse331/Documents” and returns a vector of the directories in order, so “/user/cse331/Documents” => {“user”, “cse331”, “Documents”}.

Notes:

- The activeStack private member variable should be used to reference the navigation stack at the top of the storedStacks
- When pushd is called, you are pushing a new stack on to storedStacks, not activeStack.

Using the make file and Testing your work:

You are provided with a make file .
When you are finished with your program, upload your work to Arctic. Use the make file to run your main.cpp on Arctic, fix any issues that needs attention.
Once you are ready simply run make test command to test your code.
Please make sure to have your tests sub folder in your project folder in order to run the make test command successfully.