Final Exam program (45 points)

For this program you will simulate a series of function calls and returns and intermediate processing (specifics of the class are defined at the end). The "commands" will be coming from the file *final.dat* that will contain a series of lines of one of the following forms

call *name instructionPtr*
return
stringCommand

Copy the data file with the command: **cp ~cen/data/final.dat .**

The words "*call*" and "*return*" will appear exactly as that at the first of a line. The lines containing call will also have a function name that is a string with no spaces and an integer representing the instruction pointer for the starting location of the function. A line with the word *return* will only have the word *return* on that line and represents the return command in C++. Lines that do not begin with *call* or *return* will have a single string with no spaces on it representing some form of processing that is being done in the function.

Your program should utilize the stack class that you write to push current function data on the stack when a call is encountered and pop an old function off the stack whenever a return is encountered. If you read a line that has a *stringCommand* on it, simply print the command.

Specifically:
1. If you read a ***call,*** push the current function data onto the stack, read the new name and new instruction pointer and set the current function data to be the newly read data. Then print that you are calling function *newName*.
2. If you read a ***return***, print returning from *currentFunctionName*, pop function data off the stack making it the new current function data, and print this new function's name.
3. If you read something other than call or return, simply print it.
4. Since a program is actually called from the command line, begin your program by printing the string "me@ranger1%~/OLA$ ./a.out " to represent your running the program.
5. The second thing the program should do is set up and push the following function data onto the stack so that the final return from main will print the prompt:
   name = "me@ranger1%~/OLA$ "
   instructionPtr = 0
6. Before reading and processing the file, but after specification #5 above, set the current function data to have a name of main and an instruction pointer = 1000.

Specifications for data structures (i.e., you need all the following :-):
1. structure called functionData that has two members: a string name and an int instructionPtr
2. structure called Node that has data of type functionData and a next variable which is a pointer to a Node.
3. a Stack class that has a private Node pointer called head and the following functions
   constructor
   void push(functionData)
   void pop()
   functionData top()
   bool empty()
   destructor

A sample run is as follows:

final.dat contains:

```
call func1 1204
processing
return
call func2 4427
doStuff
call func3 5321
reallyImportantStuff
return
call func4 1708
moreStuff
return
finishOff
return
wrapUp
return
```

program output

```
calling function func1
processing
returning from func1
main
calling function func2
doStuff
calling function func3
reallyImportantStuff
returning from func3
func2
calling function func4
moreStuff
returning from func4
func2
finishOff
returning from func2
main
wrapUp
returning from main
me@ranger1%~/OLA$
```

Turnin the program with command: **handin final stack.h stack.cpp main.cpp**