Basic Command Processing
Cpt S 223 Homework Assignment
Washington State University

Submission Instructions: (see syllabus)

Assignment Instructions:

Read all the instructions *carefully* before you write any code.

Part 1: Set up a main function to read commands from stdin:

In this assignment, you will read simple commands from standard input. Create your code in the file main.cpp. This assignment only requires a single .cpp file. Future assignments will have multiple .cpp and .h files. You will display your name and ID number on a single line to start with and then enter the command processing loop. There will be one command per line, so your main function (pseudo-code) will be something like:

```
cout << "Student Name, ID#" << endl;
while (true)
{
  string cmd = ReadLine();
  ProcessCommand(cmd);
}

cout << "Done" << endl;
```

Most command strings will be much like C function calls in terms of syntax. In future assignments, you will have a specific set of commands to handle. In this assignment, you are just processing generic commands and parsing out function name and arguments. Commands will be of the form:

```
commandName(paramList)
```

For each command, display 2 lines of output. The first is the string "Name:" followed by a space and the function name. The next is the string "Parameters:" followed by a space and the parameters (meaning the string content between the parentheses). Make sure you match casing! Examples follow in the table on the next page.

Important: There is one special-case command used to break the processing loop. If the command is **quit()**, then break the processing loop and fall through to the code that displays "Done" as the last line.

Input Command	Output
function1(a,b,c)	Name: function1 Parameters: a,b,c
this_is_the_function_name(params)	Name: this_is_the_function_name Parameters: params
CPTS223(Casing_Matters)	Name: CPTS223 Parameters: Casing_Maters
print(Spacing matters too)	Name: print Parameters: Spacing matters too
12345(Supports,numbers,@nd,symbols)	Name: 12345 Parameters: Supports,numbers,@nd,symbols
NoParamFunc()	Name: NoParamFunc Parameters:

Part 3: Link up all the pieces:

Put all the .cpp and .h files from your project, and ONLY these files, into a .zip file and submit it online when you are done. Do not put the files in a .rar or some other format. Use .zip or you will not get credit for your submission.

Testing using I/O redirection: This will be covered in class and you must attend lecture to get all the details. The short story is, if you've compiled a.out, want to test input file in1.txt, and produce output in myout1.txt, the command is:

```
./a.out < in1.txt > myout1.txt
```

Final notes:

Only display information on the screen with cout when the command says to. An automated grading application will be used to grade these assignments and the output must exactly match the assignment requirements for it to be properly recognized and graded. Take a good look at the sample input and output files included with the assignment to make sure your application produces the same results for each input.

Also, for commands that you cannot get working, you must still process the command so that the number of lines in your output matches up with the expected number of lines. Simply process the command and display something like "not implemented" if you did not implement the command fully. This will ensure that the grading app can grade your output line-by-line with no problems.