Programming Assignment 2 (PA2)
Functions, Header Files, File I/O, Arrays, Random Numbers, Recursion, Pointers, Dynamic Memory

Out: October 3rd, 2015, Saturday-- DUE: October 29th, 2015, Thursday, 11:59pm
EC327 Introduction to Software Engineering – Fall 2015

Total: 100 points

- *You may use any development environment you wish, as long as it is ANSI C++ compatible. Please make sure your code compiles and runs properly under the Linux/Unix environment on the PHO 305/307 (or eng-grid) machines before submitting.*
- *PAs may be submitted up to a week late at the cost of a **30% fixed penalty** (e.g., submitting a day late and a week late is equivalent). It is in your best interest to complete as many PA questions as possible before the deadline. If you have missing questions in your original submission, you may complete and submit the missing solutions during the following week. Any submissions after the deadline will be subject to the 30% penalty. No credit will be given to solutions submitted after the 1-week late submission period following the deadline.*
- *Follow the assignment submission guidelines in this document or you will lose points.*

Submission Format (Please Read)

- Use the **exact** file names specified in each problem for your solutions.
- Complete submissions should have **10 files.** Put all of your files in a single folder named: *<your username>_PA2* (e.g., dougd_PA2), zip it, and submit it as a single file (e.g., dougd_PA2.zip).
- Submit to Blackboard.
- Please do **NOT** submit *.exe and *.o or any other files that are not required by the problem.
- **Comment your code (good practice!)**

Coding Style

As you become an experienced programmer, you will start to realize the importance of good programming style. There are many coding "guidelines" out there and it is important that you come to adopt your own. Naming conventions will help you recognize variableNames, functions, CONSTANTS, Classes etc. Similarly, there are many ways to elegantly format your code so that it is easy to read. All of these issues do not affect compilation and hence are easy to overlook at this stage in your development as a programmer. However, you ability (or inability) to create clean, readable code could be the difference in your career when you leave college.

Here are some links you should explore for more on good coding conventions:

http://www.c-xx.com/ccc/ccc.php

http://geosoft.no/development/cppstyle.html

http://google-styleguide.googlecode.com/svn/trunk/cppguide.html

Q1. Functions, Header Files, File I/O (30 pts)

READ THE WHOLE QUESTION BEFORE YOU START. IT IS LONG AND HAS MANY PARTS.

You need to write a C++ program that will interactively accept commands from the user. Each command is a single character followed by one, two, or three parameters and specifies a function that needs to be performed. Your program must do the following:

- Ask the user for the next command code – "`Please enter command code:`"
- Read the command code
- If the command is illegal, print an appropriate error message – "`Invalid command code`"
- else if the command is quit, terminate the program
- else ask the user for the parameters – "`Please enter command parameters:`"
- Read the arguments
- Execute command

Your program must continue this loop until the "quit" command is given.

Commands that you need to include:

F or f	Compute the **F**actorial value	Given an integer number
B or b	Compute the fI**B**onacci number	Given an integer index
R or r	Compute square **R**oots	The sqrt <math> function
D or d	Compute o**D**d numbers	List odd numbers between "first" and "last" number provided
L or l	Compute natural **L**og numbers	The log <math> function
N or n	Compute **N**yanCat value	(See function description)
I or i	Read from **I**nput file	
O or o	Write to **O**utput file	
Q or q	**Q**uit program	

You need to define a global program constant named "ENTRIES" which specifies the maximum calculations performed by your program. Initialize this to 10 at the beginning of your program; NOT INSIDE ANY FUNCTION. **(Think about why it cannot be in a function)**

```
extern const int ENTRIES = 10
```

This sets up a variable that is available to be used across multiple files. Look up how to do this properly.

The parameters for F/f and B/b are single integers. There are no parameters for q/Q. The parameters for O/o and I/i are names of the files you want to write to or read from. The parameters for D/d are the first and last values as integers.

The parameters for R/r, L/l, N/n are the double values "first," "last," and "delta." These values will be inputted by the user as three doubles. When a function is called, the corresponding function must be called for all arguments values from first to last, in steps of delta. For instance, for command 'R':

```
Square root of (first)
Square root of (first + delta)
Square root of (first + delta*2)
```

```
Square root of (first + delta*3)
…
Etc
```

If delta is ≤ 0 or first > last, then no computation takes place. **Print "No computation needed."** If these special cases are not triggered, then compute the values until you have completed the full range of values or you have computed more than **ENTRIES**, whichever comes first. Your program must check for legal command codes, legal delta values, and make sure first ≤ last. However, you DO NOT need to check that a given argument range is legal for the function being called. For instance, you do not need to check that sqrt is being called with positive argument. You are allowed to assume that the user will only enter one keystroke when asked for the command. You should always include the first value in your computation, then proceed in delta increments until you hit the last value or exceed the ENTRIES value. **If a delta increment exceeds the "last" value, just use the last value for your final computation.**

All the functions must be declared in a *.h file and defined in a *.cpp file. They will be defined as follows:

```
pa2Functions.h
pa2Functions.cpp
```

pa2Functions.h will hold all the function prototypes while **pa2Functions.cpp** will hold the implementations. In addition to these two files, you will have a **Q1.cpp** where you will run the main function and call the functions in your library.

pa2Functions.h must include the following prototypes, **EXACTLY** as given here.

```
void initialize();
```

This function must print the program output header. The output header should contain the name and number of the course, the semester and year, the assignment name, and the value of ENTRIES. The values of ENTRIES must be obtained from Q1.cpp. This function is called from main().

(Header Sample)

```
EC327: Introduction to Software Engineering
Fall 2015
Programming Assignment 2
Value of Entries is: 10
```

```
bool checkCode(char);
```

This function should return true if the command code in parameter "code" is a legal code character; otherwise, the function should return false. You have to use this function to check your user input. This function is called from main().

```
void writeDataToFile(const char *);
```

This function will write the output of the program to a file specified by user input. This only works for input provided AFTER this command has been provided. The output being written to a file should also be displayed to the console as well. Writing to a file ends when the quit command is provided.

`void readDataFromFile(const char *);`

The function will read commands from a file specified by user input. This function is called from main(). The commands in this file should be formatted each on two lines consistent with how the user would enter them from the command line.

e.g.

```
f
10
R
10 20 .2
```

`double findNyanCatValue(double);`

If this function takes in "myNum" as an argument this function with return $(2 * myNum) + (myNum * (6)^{myNum})$

`int factorial(int);`

This function returns the factorial value of the given integer argument.

`int fibonacci(int);`

This function returns the Fibonacci number at the index given by the provided argument.

`double findSqrtValue(double);`

This function returns the square root of the argument.

`double naturalLog(double);`

This function returns the natural log of the provided argument.

`int findNextOddValue(int);`

This function returns an odd number **closest and higher** to the provided number.

Final requirements

- All functions (other than main) must be prototyped (declared) in **pa2Functioins.h**
- All functions (other than main) must be implemented (defined) in **pa2Functions.cpp**
- Main function must be implemented in **Q1.cpp**
- Comment your code!!!
- All input and output (other than file I/O) must be done with `cin` and `cout`, respectively

SAMPLE OUTPUT WILL BE PROVIDED TO BLACKBOARD. Please follow this formatting!!

Q2. Loops, functions, and basic recursion (20 pts)

Write a recursive function (`void PrintRhombus(int n);`) that takes an integer as an input, and prints the number rhombus shown below. Write a test program that prompts the user to enter an integer between 1 and 9, and then calls the function. Include a check statement in your main program or in the PrintRhombus function to print an error message if the user enters a number outside [1,9].

Turn in one file called Q2.cpp that contains all of your code for this problem.

Text in < > in the examples demonstrates the user inputs entered via the keyboard.

Sample run:

Enter a number [1-9]: <5> <enter>

```
        1
      1 2 1
    1 2 3 2 1
  1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
  1 2 3 4 3 2 1
    1 2 3 2 1
      1 2 1
        1
```

**Spaces on the right and left sides of the rhombus and the exact number of spaces between two columns are not significant as long as the numbers are aligned as shown above.*

HINT: Think about how to generate the list of numbers first. Then figure out how you would format this into the diamond. You could generate the list of numbers first and then output them formatted correctly outside of the recursive call. You DO however need to use recursion to calculate the number sequence.

Q3. File I/O, Arrays, Random Numbers, and Functions (30 pts)

1. Write a function named `DataToFile()` that generates N random integers from 0 to M (inclusive) and writes them to a text file named `numbers.txt` (this will be one of the arguments), one number per line. It should return 0 if successful and 1 otherwise. You should use the following function prototype:

```
int DataToFile(const char *filename, int N, int M);
```

2. Write a second function called `DataFromFile()` that reads the data from the specified text file (think what that name should be), and also finds the size. It should return 0 if successful and 1 otherwise. You can use the following function prototype:

```
int DataFromFile(const char *filename, int myArray[], int &size);
```

3. Write functions `getMean()` and `getStdDev()` to compute the mean and standard deviation **using arrays**. Use the non-biased version for the standard deviation σ (i.e. division by N):

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - mean)^2}$$

Each function should accept the array as an argument and return the mean or standard deviation as a float. You can use the following function prototypes:

```
float getMean(int myArray[], int size);
float getStdDev(int myArray[], int size);
```

4. Write a `main()` function that assumes the `numbers.txt` file is given in the same folder as the code files, calculate the size of an array, mean and standard deviation, the largest value, the smallest value, and prints the array out in forward and reverse order. For example, your final output screen should look like this (for some random `numbers.txt` file):

```
Array size is: 4
Mean is: 44.5
StdDev is: 17.56
Array values forward are: 42, 63, 56, 17
Array values reverse are: 17, 56, 63, 42
The largest array value is: 63
The smallest array value is: 17
```

We will test your code on our `numbers.txt` file. Assume the file can be **any** size between (and including) 0 and 1000.

Submission requirements: You should submit **five separate files:** `Q3.cpp, q3func.h, q3func.cpp, statistics.h, statistics.cpp.`

- `Main()` should be saved in `Q3.cpp`.
- `DataToFile()` and `DataFromFile()` should be stored in a single `q3func.cpp` file with the header file named `q3func.h`.
- `getMean()` and `getStDev()` should be stored in a single `statistics.cpp` file with the header file named `statistics.h`.

Important:

1. You may use <cmath> library inside your statistics.cpp ONLY for function "sqrt", but you may not use <cmath> for ANY other function inside statistics.cpp.

2. You cannot have "cout" inside either statistics.cpp or q3func.cpp.

3. You cannot have "#include<fstream>" inside q3.cpp or statistics.cpp.

4. Please do not call "DataToFile" function inside q3.cpp. We will be writing code that calls your DataToFile function for testing.

Q4. Array of pointers and dynamic memory allocation (20 points)

Write a function called **ReverseMultiply** that doubles the size of an integer array and fills the second half with a copy of the first half in reverse order and adds a final additional location for the product of all the numbers.

Example: Given array: [1 2 3 4 5]
The resulting array obtained by ReverseMultiply function: [1 2 3 4 5 5 4 3 2 1 14400]

The function prototype is as follows: **int * ReverseMultiply (int *list, int size);**

The function should return a pointer to an integer array that has a size equal to 2*size + 1. The values in the new array at indices from 0 to size-1 should have the same values as in the original input array in the same order. The rest of the values in the new array should be the reverse order of the original array. The final value should be the product.

Your main function (in Q4.cpp along with the ReverseMultiply function) should request the size of the original array and then allow the user to input the values. It should then print out the original array and address along with the new array and address.
For example:

```
Enter the number of entries: 3 <newline>
Entry 0 is:  3 <newline>
Entry 1 is:  7 <newline>
Entry 2 is:  20 <newline>
*****
Original array is: 3 7 20 and the address of the zero element is: <something in hex>
Final array is: 3 7 20 20 7 3 176400 and the address of zero element is: <something in
hex>
```

You should submit `Q4.cpp`

Submission format (same as the one on the first page)

- Use the exact file names specified in each problem for your solutions.
- Complete submissions should have **ten files.** Put all of your files in a single folder named: *<your username>_PA2* (e.g., dougd_PA2), zip it, and submit it as a single file (e.g., dougd_PA2.zip).
- Submit to Blackboard as outlined in in the announcements section of Blackboard for this assignment.
- Please do **NOT** submit *.exe and *.o or any other files that are not required by the problem.
- **Comment your code (good practice!)**

Complete submission should have **ten files**: `Q1.cpp`, `pa2Functions.cpp`, `pa2Functions.h`. `Q2.cpp`, `Q3.cpp`, `q3func.h`, `q3func.cpp`, `statistics.h`, `statistics.cpp`, `Q4.cpp`