1. (1pt) **Name:** ______________________________

   Both the electronic and paper portion of this assignment are due at the beginning of class. Assignments after 8am will be considered late. Ensuring that the electronic portion of the assignment is submitted on-time and in the correct format is critical. Make sure to leave yourself plenty of time to submit the project.

2. (4pt) Write the MD5 hash of your submitted code here. ______________________________

3. (4pt) How many iterations does the following loop make? ____________

```cpp
int count = 1;
while (count < 30) {
   count = count * 2;
}
```

4. (4pt) How many iterations does the following loop make? ____________

```cpp
int count = 15;
while (count < 30) {
   count = count * 3;
}
```

5. (4pt) How many iterations does the following loop make? ____________

```cpp
int count = 1;
while (count < n) {
   count = count * 2;
}
```

6. (4pt) How many iterations does the following loop make? ____________

```cpp
int count = 15;
while (count < n) {
   count = count * 3;
}
```

7. (24pt) For each of the following code snippets, determine how many stars are displayed for $n = 5, 10, 20$. Use the Big O notation to estimate the time complexity.

   (a)

```cpp
for (int i = 0; i < n; i++) {
   cout << '*';
}
```

(b)

```
for (int i = 0; i < n; i++) {
   for (int j = 0; j < n; j++) {
      cout << '*';
   }
}
```

(c)

```
for (int k = 0; k < n; k++) {
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         cout << '*';
      }
   }
}
```

(d)

```
for (int k = 0; k < 10; k++) {
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         cout << '*';
      }
   }
}
```

8. (8pt) Design an $O(n)$ time algorithm for computing the sum of integers from n_1 to n_2 where $0 \leq n_1 \leq n_2$.

9. (7pt) Is it possible to improve the performance of the algorithm from the previous question so that it is $O(1)$ (a constant time algorithm)? If it is possible, provide the algorithm; if not, explain why it is not possible to create such an algorithm.

Electronic Submission

10. (80pt) You will write a program that will aid in evaluating the performance of the sorting routines we have discussed in class. Specifically, your program should perform tests on the `bubbleSort`, `insertionSort`, `mergeSort`, and `selectionSort` functions defined in `sort.cpp` that is posted to the course documents page[1]. Modify the functions so that they return a **`long long`** value that represents the amount of operations required to complete the sort.

    Typing **make** at the root of your project submission will create an executable file called **analyze**. Your program will make use of commandline arguments to determine which files to load, what sorting routine to execute, and the number of files that will be loaded. You will make use of the files

    asc1.txt, . . . , asc10.txt,

    desc1.txt, . . . , desc10.txt, and

    random1.txt, . . . , random10.txt

    that are posted to the course documents page. The **asc**, **desc**, and **random** files contain dictionary words in ascending, descending, and random order respectively. The program should be invoked in the following manner:

    ```
    analyze SORT FILENAMEPREFIX NUM
    ```

    The program arguments are defined as follows:

    SORT
    The name of one of the sorting routines mentioned above. Valid options are `bubbleSort`, `insertionSort`, `mergeSort`, and `selectionSort`.

[1] `https://raw.githubusercontent.com/wiki/markroyer/latex-homework-template/homework-examples.tgz`

FILENAMEPREFIX
The name of the file to be loaded without the number or `.txt` extension. For this assignment, we will use `asc`, `desc`, and `random`.

NUM
The number of files to be read.

The output of the program is a single file named **SORT-FILENAMEPREFIX.data** that contains two columns of data separated by a single tab (`"\t"`). The first column is the number of words in the file, and the second column is the amount of operations it took for the function to sort the data. The output file may have lines containing descriptive information as long as those lines begin with a `#` symbol.

Example

Suppose that the program was run with the options as shown below:

```
analyze selectionSort asc 10
```

This would generate the output file **`selectionSort-asc.data`**. The contents of the file would have ten lines of output (one for each file) and look similar to the following:

```
# Result of running selectionSort on asc1-10
# n     ops
10      100
100     10000
1000    1000000
etc ...
```

For each sorting routine, use the generated output file from your program to graph the results versus the expected time complexity for the chosen sorting algorithm. Put these results and a brief discussion of why or why not your results match the theoretical ones in a single pdf file named **results.pdf**.

Note: As with previous assignments, no other output should be generated by the program. You will lose points if the specification is not strictly adhered to.

Submission: Submit a single tgz file to `http://yoursite.com/u/` using the following naming convention.

```
lastname-hwNN-SNUM.tgz
```

You should replace *lastname* with your last name, *NN* with the assignment number (eg. 05), and *SNUM* with a four digit number specifying the submission version. Extracting the contents of the tarball should create a single folder containing your project following the naming convention above without the extension. The folder should contain the following files:

README A file that describes the submission content and how to build and run your program.

Makefile The script for building your project.

Doxyfile Doxygen input file for creating documentation.

sort.cpp Source file containing the main function, the sorting functions, and any other additional functions that may be needed.

results.pdf

Important: Make sure that your submission does not include generated files such as object files and doxygen output documentation. You may lose points if this is not strictly followed.

Questions: For clarification, please post additional questions to the newsgroup.

Extra Credit* (10pt) In the textbook the author claims that $O(\log n) = O(\log_2 n) = O(log_a n)$. Does the same hold for Θ? That is, can we say that for all positive real numbers a and b where $a \neq 1, b \neq 1$ does $\Theta(\log_a n) = \Theta(\log_b n)$? If so, prove that the equality holds, otherwise show why it does not.

Extra Credit* (30pt) Modify the **analyze** program to support **externalSort**. Perform the same analysis of the sorting routine using the ascending, descending, and random input files from the homework and summarize the results. Append these results to the end of your **results.pdf** file.