CS 2420 Section 2 and Section 3
Algorithms and Data Structures

Spring Semester, 2016

Assignment 1: Algorithm Analysis

Due Date: 11:59 p.m., Friday, Jan. 29, 2016

(**Note:** This assignment has **seven** questions, including both written questions and programming questions. For the programming questions, please submit only your source files (.cpp files) via Canvas. For written questions, please submit your solution in a word document via Canvas. Please compress all your documents in one .zip file for Canvas submission.)

1. For each of the following pairs of functions, indicate whether it is one of the three cases: $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. For each pair, you only need to give your answer and the proof is not required. **(20 points)**

   (a) $f(n) = 9n^2 - 5n + 6$ and $g(n) = n^3 + n^2 - 129$.

   (b) $f(n) = \log n + 2n$ and $g(n) = 3\log^2 n + \log n$.

   (c) $f(n) = n^4 + n^3$ and $g(n) = 6n^4 - 3n^2 - 10$.

   (d) $f(n) = n \log n$ and $g(n) = n + n^2$.

   (e) $f(n) = 2^n$ and $g(n) = 3n^{100} + 7n^3 - 6$.

2. For each of the following program fragments, please write the time complexity (e.g., $T(n)$) in terms of the number of primitive operations performed and give the running time using big-O notation of n. **(20 points)**

   (a)
   ```
   sum = 0;
   for (i = 0; i < n; i++)
       sum++;
   ```

   (b)
   ```
   sum = 0;
   for (i = 0; i < n; i++)
       for (k = 0; k < n * n; k++)
           sum++;
   ```

   (c)
   ```
   sum = 0;
   for (i = 0; i < n; i++)
       for (k = 0; k < i; k++)
           sum++;
   ```

(d)
```
sum = 0;
for (i = 0; i < n; i++)
    for (k = 0; k < i * i; k++)
        sum++;
```

(e)
```
sum = 0;
for (i = 0; i < n; i++)
    for (k = i; k < n; k++)
        sum++;
```

3. Use recurrence to write the time complexity (e.g., $T(n)$) in terms of the number of primitive operations performed and give the time complexity of the following recursive function using big-O notation of n. **(5 points)**

```
void fun(int n)
{
    if (n ≤ 1)
        return;
    fun(n − 1);
    for (i = 0; i < n; i++)
        cout << " * ";
}
```

4. Consider sorting n numbers stored in an array $A[0, \ldots, n-1]$ in the ascending order by first finding the smallest element of A and exchanging it with the element in $A[0]$. Then, find the second smallest element of A, and exchange it with $A[1]$. Continue this manner for the first $n-1$ elements of A. This algorithm is known as **selection sort**. **(15 points)**

   (a) Use the convention mentioned on slide 19 of Ch1.AlgorithmAnalysis.pdf to write the pseudocode for this algorithm.

   (b) Give the (worst-case) running time of this algorithm using the big-O notation.

   (c) Give the **best-case** running time of this algorithm using the big-O notation.

5. Write a C++ program to implement the **selection** sort algorithm introduced in the last question. Please use the "starter" cpp file called "hw1_Q5.cpp" to add your C++ implementation for each function. The program first reads the number of elements in the input file "hw1_Q5_input.txt" and then reads the numbers and stores them in the dynamic array A. After the numbers in the array are sorted, the numbers will be output to an output file called "hw1_Q5_output.txt" (they will be output on the console/screen too). A sample input file ("hw1_Q5_input.txt") and its output file ("sample_hw1_Q5_output.txt") are provided for your reference. **(15 points)**

   In addition, please follow the instructions below, which are also applicable to Question 6 (and programming exercises in future assignments).

- The input and output files are only used for grading. When you test/debug your program, you may use other ways for the input and output. When grading your code, the grader will use an input file with different numbers.
- Please submit only your cpp file to Canvas. Before submitting your program, you must change the file name "hw1_Q5.cpp" by adding an _ followed by your A number or your name. For example, if your A number is A1234567, you should change the file name to "hw1_Q5_A1234567.cpp". This will make the grading easier.
- Please try to write your code in a way that is easy to read. Give some appropriate comments wherever you feel necessary.

6. Implement the binary search algorithm discussed in CS1410. **(15 points)**

   Please use the "starter" cpp file "hw1_Q6.cpp" to add your C++ implementation for some necessary functions and two required functions. The program first reads the sorted numbers in the data file and stores them in a dynamic array A. Then, the program reads the numbers in the search file and stores them in a dynamic array B. For each number x of B, we search it in A by using binary search (i.e., if x is in A, then return its index in A; otherwise, return -1). The search results will be output in an output file "hw1_Q6_output.txt" (they will be output on the console/screen too). A sample input data file ("hw1_Q6_data.txt"), a search file ("hw1_Q6_search.txt"), and its output file ("sample_hw1_Q6_output.txt") are provided for your reference.

7. This exercise is to convince you that exponential time algorithms should be avoided.

   (10 points)

   Suppose we have an algorithm A whose running time is $O(2^n)$. For simplicity, we assume the algorithm A needs 2^n instructions to finish for any input size of n (e.g., if $n = 5$, A will finish after $2^5 = 32$ instructions).

   According to Wikipedia, the fastest supercomputer in the world in 2015 is Tianhe-2 (which is located in Guangzhou, China) can perform about 3.4×10^{16} instructions per second.

   Suppose we run the algorithm A on Tianhe-2. Answer the following questions.

   (a) For the input size $n = 100$ (which is a relative small input size), how much time does Tianhe-2 need to finish the algorithm? Give the time in terms of **centuries**.

   (b) For the input size $n = 1000$, how much time does Tianhe-2 need to finish the algorithm? Give the time in terms of **centuries**.

Total Points: 100