CSI 3105 ASSIGNMENT 1 Fall, 2014

Due Date: Wednesday October 1, 2014, by 4:00 pm. You can hand in your assignment directly to me in class that day, or else you can bring it to my office (STE 5106) before 4:00 pm (put it under my door if I am not there). Assignments from 1 minute to 24 hours late will lose 6 marks out of 60 marks (10%). Assignments received after 4:00 pm on Oct. 2 will receive a grade of 0.

Please remember that these assignments are my way of providing practice questions for you for the tests in the course.

On this and all future assignments: You must be sure to include your name and student ID on the assignment, as well as the course code and the assignment number. You will lose one mark if any of these are missing. In your assignment, please be sure to explain all of your solutions CLEARLY, or you will lose marks. To be complete, in any worst case analysis you do you must state the input size, the basic operations being counted, and a general input of your fixed input size which gives the worst case (you should state these things in your solution even if some of them are given in the questions). Whenever possible, simplify your answers by using any appropriate formulas found in Appendix A. Also, you will be marked on the efficiency of any algorithms you design.

This assignment will be marked out of 60.

---

1. [11 marks total]
Consider the following variation of the Fibonacci numbers, called the Boyd numbers, which are defined recursively as follows:

$B_0 = 0,\ B_1 = 1,\ B_2 = 2,\ B_3 = 3,\ B_4 = 4$ and for $n \geq 5,\ B_n = B_{n-1} + B_{n-2} + B_{n-5} + 2.$

So the Boyd Sequence is 0, 1, 2, 3, 4, 9, 16, 29, …..

A recursive algorithm which finds the nth term in the Boyd Sequence is the following:

Problem: Determine the nth term in the Boyd Sequence.
Inputs: a nonnegative integer n.
Outputs: boyd, the nth term in the Boyd Sequence.

```
int boyd (int n)
{
   if (n<= 4)
      return n;
   else
      return boyd(n-1) + boyd(n-2) + boyd (n-5) + 2;
 }
```

a) [1 mark] Draw the recursive call tree for this algorithm for n = 9.
b) [1 mark] Let T(n) represent the number of recursive calls required by the algorithm to calculate B_n. Create a table that shows the values of T(n) for n = 0, 1, ..., 9.

c) [9 marks] Let T(n) represent the number of recursive calls required by the algorithm to calculate B_n. Prove the following, using induction: For n>=5, $T(n) > 3^{n/5}$. Note that you must CLEARLY state, for each line in your proof, the reasoning behind it (i.e. how you got to that line from the line above it).

2. [5 marks] Write an iterative algorithm (using the same sort of C++-like pseudocode used in the text) for finding the nth term in the Boyd Sequence which is much more efficient than the algorithm in question 1. Do a worst-case analysis of your algorithm in which you count the number of additions and subtractions (but do not count additions or subtractions that are increments of a loop counter).

3. [8 marks (4 each)]
For each of the following inputs, calculate the exact number of comparisons of x with keys (counting 2 inside the loop, as we did in class) for the iterative version of binary search (Alg. 1.5). For each iteration of the algorithm, show the values of high, low and mid (so it is easy for us to check your answer). For both examples, compare the number you get to the number of comparisons in the worst case for this value of n, using the exact analysis we did in class. Is the number of comparisons the same or less than the worst case number? (If you get a number that is larger, you had better check it!).

a) n = 8, x = 4 (i.e. x is not in S, and is smaller than all the numbers in S),
S = [8, 14, 23, 29, 30, 32, 36, 45]

b) n = 16, x = 22 (i.e. x is not in S, and x is between the middle two numbers), and
S = [4, 5, 6, 8,10, 16, 18, 19, 25, 27, 30, 31, 40, 45, 47, 51]

4. [12 marks total, 6 marks each]
Do a worst-case analysis for the following algorithm segments, counting the number of multiplications which occur. (I have marked the lines with the multiplications you are to count with {****}). For all of these algorithms, use n as your fixed input size (even though n doesn't really represent the "size" of the input). Be sure to include an explanation with your answers.

a)
```
z = 2;
p = 5;
for (i = 1; i <= n, i++)   {
   for (j =1; j <= n; j++) {
       for (k =1; k <= j; k++) {
          p = p * 20 * z;                {****}
       }
   }
}
```

b)
```
answer = 25;
for (i = 0; i <= n-1; i++) {
        for (k = 1; k <= 3^i; k++) {
            answer = answer * 8;         {****}
        }
 }
```

5. [11 marks total]
Suppose you need to search a given unsorted list of n keys for a key x, and you know you will need to perform such a search on the same list for different values of x a total of R times. You may assume that n is a power of 2, i.e. $n = 2^k$ for some $k \geq 0$.

Consider two algorithms:

Alg. 1: Use sequential search R times.
Alg. 2: Use exchange sort once (to sort the list) and then use binary search (iterative version from class) R times.

a) [5 marks]
Do a worst case analysis of each of these 2 algorithms. Use n as your input size, and count the number of comparisons of x to keys in the list and comparisons of keys to each other. When stating the complexity of sequential search, exchange sort and binary search, you must use the value for W(n) which we developed IN CLASS (note that this is sometimes different than the W(n) from the text).

b) [5 marks]
For what general values of R will Alg. 2 be faster than Alg. 1? (i.e. how big does R have to be, as a function of n, for Alg. 2 to be the more efficient algorithm?).

c) [1 marks]
Use your solution from b) to determine how large R must be for a list with 2048 keys.

6. [13 marks total]
Suppose that you are given a number k and an array N of k numbers, indexed from 1 to k. You may assume the numbers in N are integer, but you may not assume they are non-negative, and you may not assume they are distinct. Design an algorithm that decides if N contains the numbers 1, 2, 3, ..., k, arranged in any order. For example, suppose that k=7, and N = [4, 1, 5, 2, 3, 7, 6], then the answer would be yes, but if N= [1, 2, 3, 4, 10, 6, 7] the answer would be no. Remember that you will be marked on the clarity and time efficiency of your algorithm.

a) [9 marks]
Describe the main ideas of your algorithm in words, then describe your algorithm in pseudocode (try to use pseudocode which is similar to that in the textbook).

b) [4 marks]
Do a worst case analysis of your algorithm, counting both the comparison of the numbers in N with anything, AND counting the number of times you put a number into an array. Use k as your input size. Be sure to explain your analysis.