NAME: Solution ID#:

COMP 250 Winter 2013 - Midterm examination 1

February 6th 2013, 9:35-10:25; BA
This is an open book exam, but no electronic devices is allowed.

1 Java programming (25 points)

a) (9 points) What will the following Java program print when executed?

```
class question2 {
   static public void questionA(int x) {
      x = x * 3;
   }

   static public int questionB(int x) {
      x = x + 2;
      return x;
    }

    static public void questionC(int array[]) {
       array[0] = array[0] + 1;
    }

    public static void main(String args[]) {
       int x, y;
       int a[] = new int[10];
       x = 1;
       y = 1;
       a[0] = 1;
       questionA(x);
       y = questionB(y);
       questionC(a);

       System.out.println("x = " + x);
       System.out.println("y = " + y);
       System.out.println("a[0] = " + a[0]);
   }
}
```

Note: Not all copies had the same question.

Answer:

x = 1
y = 3
a[0] = 2

b) (16 points) We say that two students are evil twins if the sum of their student ID numbers is exactly 999999999. Write the Java code for the containsEvilTwins method below, which takes as argument a studentList and return true if the studentList contains at least one pair of evil twins, and false otherwise. Your program must run in time $O(n \log n)$, but you don't need to actually demonstrate that it does.

You can use any of the Java methods seen in class or in the assignments, without having to give the code for them.

```
class studentList {
    int studentID[];
    int numberOfStudents;
    String courseName;

    public static boolean containsEvilTwins( studentList L ) {
    /* WRITE YOUR CODE HERE */
    Array.sort( L.studentID);
    for (i=0; i<[redacted] L.numberOfStudents; i++) {
        if (binarySearch (L.studentID, 0, L.numberOfStudents,
                          999999999-L.studentID[i])){
            return true;
        }
    }
    return false;


    }
}
```

2 Execution of recursive algorithms (24 points)

Consider the following sequence of integers S, defined as:

$$\begin{aligned} S(0) &= 1 \\ S(1) &= 2 \\ S(2) &= 3 \\ S(n) &= S(n-3) * S(n-2), \text{ if } n \geq 3 \end{aligned}$$

a) (8 points) What is the value of $S(5)$?

n	0	1	2	3	4	5
S(n)	1	2	3	6	6	6

$S(5) = 6$

b) (8 points) Consider the following recursive algorithm that computes the value of $S(n)$:

Algorithm RecursiveS(n)
Input: A non-negative integer n
Output: The value of the $S(n)$

print n
if $(n \leq 2)$ **then return** $n + 1$
else
. a = RecursiveS$(n - 3)$
. b = RecursiveS$(n - 2)$
. **return** $a * b$

What will be *printed* when RecursiveS(5) is called?

Execution	Printed
RS(5)	5
RS(2)	2
RS(3)	3
RS(0)	0
RS(1)	1

c) (8 points) Let $P(n)$ be the number of elements that will be printed during the execution of RecursiveS(n). Give a recurrence for $P(n)$. You *don't* need to prove your answer. You *don't* need to obtain an explicit formula for your recurrence.

$$P(n) = \begin{cases} 1 + P(n-3) + P(n-2) & \text{if } n > 2 \\ 1 & \text{if } n \leq 2 \end{cases}$$

3 Big-O notation (20 points)

a) (12 points) Indicate, for each pair of functions (f, g) in the table below, whether $f(n)$ is $O(g(n))$, and whether $g(n)$ is $O(f(n))$. Write either "yes" or "no" in each box. No justifications are necessary. [Correct answer: 1.5 point, wrong answer = -0.5 point, no answer = 0 point].

f(n)	g(n)	f(n) ∈ O(g(n))	g(n) ∈ O(f(n))
$\sin(n) + 3$	$\cos(n) + 2$	Yes	Yes
$n^2 + n\log(n) + 1$	$n\log(n) + 5n$	No	Yes
3^n	$2^n + 10$	No	Yes
$n^{1.01}$	$n \cdot \log(n)$	No	Yes

b) (8 points). Give the formal proof for *one* of eight Yes/No answer you gave in (a). You are free to pick whichever one you find easiest. Your proof should only rely on the formal definition of the big-O notation.

Proof of: $\sin(n)+3$ is $O(\cos(n)+2)$

We must show that $\exists c, n_0$ such that $\sin(n)+3 \le c\cdot(\cos(n)+2)$ for all $n \ge n_0$.

We have

$\sin(n)+3 \le 4 \le 4(\cos(n)+2)$

↑ because $-1 \le \sin(n) \le 1$; ↑ because $-1 \le \cos(n) \le 1$

So for $c=4$, $n_0=1$, $\sin(n)+3 \le c\cdot(\cos(n)+2)$ for all $n \ge n_0$

$\Rightarrow \sin(n)+3$ is $O(\cos(n)+2)$

Proof of $n\log(n)+5n$ is $O(n^2+n\log n+1)$

We must show that $\exists c, n_0$ such that $n\log(n)+5n \le n^2+n\log n+1$ for all $n \ge n_0$

We have

$n\log(n)+5n \le n\log(n)+n\cdot n$ if $n \ge 5$

$\le n\log(n)+n^2+1$

$= 1\cdot(n\log(n)+n^2+1)$

So for $c=1$, $n_0=5$, $n\log(n)+5n \le 1\cdot(n\log(n)+n^2+1)$ for all $n \ge n_0$

$\Rightarrow n\log(n)+5n$ is $O(n^2+n\log n+1)$

4 A faster recursive Power algorithm (21 points)

In class, we studied a recursive algorithm to compute the n-th power of some number a. The algorithm was based on the idea that $a^n = a \cdot a^{n-1}$ and had a running time that was $O(n)$.

Question: Write the pseudocode of a different version of the recursive Power algorithm that runs in time $O(\log_2(n))$. You don't need to analyze or prove the running time of your algorithm.

Hint: Your algorithm should be based on the following observation: When n is even, $a^n = (a^{n/2})^2$. When n is odd, $a^n = a \cdot (a^{(n-1)/2})^2$.

Algorithm Power(a, n)
Input: A real number a and a non-negative integer n.
Output: Returns the value of a^n.

```
if (n=0) return 1
if (n is even) then
    x ← Power(a, n/2)
    return x*x;
else
    x ← Power(a, (n-1)/2)
    return a*x*x
```

5 Running time analysis (10 points)

The following algorithm takes two arrays of integers as input and counts how many elements they have in common.

Algorithm intersectionSize(A, B, n)
Input: A is an array of n distinct integers; B is an array of n distinct integers
Output: The number of elements that are present in both arrays

```
count ← 0                                                    1
mergeSort(B, 0, n − 1)                                       (3n log n + 4n + 5) + 1
for i ← 0 to n − 1                                           1 (init)
    if (binarySearch(B, 0, n − 1, A[i]) ≠ −1) then           n times: 2 + (4 log(n) + 2) + 2^3
        count ← count + 1                                    2 (increment i)
return count                                                 2
                                                             1
```

Assume that the worst case running time of mergeSort on an array of size n is $3n\log(n) + 4n + 5$ and the worst case running time of binarySearch on an array of size n is $4\log(n) + 2$. Let $T(n)$ be the worst case total number of primitive operations performed by the intersectionSize algorithm when executed on arrays of size n. Give an exact formula for $T(n)$.

Total: $9 + 3n\log(n) + 4n + n(13 + 4\log(n))$

$T(n) = 4n\cdot\log(n) + 17n + 9$

For checking purposes:

n	1	2	4	8
$T(n)$	1	6	23	78

6 Bonus question (5 points)

Let $T(n)$ be defined as follows:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n/2) + n + 1 & \text{if } n > 1 \end{cases}$$

Give an explicit formula for $T(n)$.

$T(n) = 3T(\frac{n}{2}) + n + 1$ (1)

$= 3\left(3T(\frac{n}{4}) + \frac{n}{2} + 1\right) + n + 1 = 9T(\frac{n}{4}) + (1+\frac{3}{2})n + (1+3)$ (2)

$= 9\left(3 \cdot T(\frac{n}{8}) + \frac{n}{4} + 1\right) + n(1+\frac{3}{2}) + (1+3) = 27T(\frac{n}{8}) + (1+\frac{3}{2}+\frac{9}{4})n + (1+3+9)$ (3)

$\vdots$

$= 3^k T(\frac{n}{2^k}) + n \cdot \sum_{i=0}^{k-1} (\frac{3}{2})^i + \sum_{i=0}^{k-1} 3^i$ (k)

$= 3^k T(\frac{n}{2^k}) + n \cdot \left(\frac{(\frac{3}{2})^k - 1}{(\frac{3}{2} - 1)}\right) + \frac{3^k - 1}{3-1}$

$= 3^k T(\frac{n}{2^k}) + 2n\left(\frac{3^k}{2^k} - 1\right) + \frac{(3^k - 1)}{2}$

When $\frac{n}{2^k} = 1$, recursion stops. $\frac{n}{2^k} = 1 \Leftrightarrow n = 2^k \Leftrightarrow k = \log_2 n$

$T(n) = 3^{\log_2 n} \cdot T(1) + 2n\left(\frac{3^{\log_2 n}}{n} - 1\right) + \frac{(3^{\log_2 n} - 1)}{2}$

$= \left(\frac{7}{2}\right) 3^{\log_2 n} - 2n - \frac{1}{2}$

$\boxed{T(n) = \left(\frac{7}{2}\right) n^{\log_2 3} - 2n - \frac{1}{2}}$

Check: $T(4) = \left(\frac{7}{2}\right) 3^{\log_2 4} - 2 \cdot 4 - \frac{1}{2} = \left(\frac{7}{2}\right) \cdot 3^2 - 8 - \frac{1}{2} = 23$ ✓

$T(8) = \left(\frac{7}{2}\right) \cdot 3^{\log_2 8} - 2 \cdot 8 - \frac{1}{2} = 78$ ✓