COMP 250 – Midterm
October 17th 2014, 18:10 – 19:55

- This exam has 7 questions.
- This is an open book and open notes exam. No electronic equipment is allowed.

Question 1 (15 points). Java programming

What will the following Java program print when executed?

class question1 {
    static public void questionA(int x) {
        x = x + 2;
    }
    static public int questionB(int x) {
        x = x + 3;
        return x;
    }
    static public void questionC(int array[]) {
        array[0] = array[0] + 4;
    }
    static public int questionD(int n) {
        if (n<1) return 1;
        return questionD(n-1)+questionD(n-2);
    }
    public static void main(String args[]) {
        int x, y, z;
        int a[] = new int[10];
        x = 1;
        y = 1;
        a[0] = 1;
        questionA(x);
        y = questionB(y);
        questionC(a);
        z = questionD(6);
        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("a[0] = " + a[0]);
        System.out.println("z = " + z);
    }
}

Answer:

x =

y =

a[0] =

z =
Question 2 (20 points). Stacks and recursion

Professor Stackbottom proposes the following recursive algorithm that is using a stack as argument.

```
Algorithm mistery(Stack S)
Input: Stack S
Output: Modifies the stack S and returns a number

value = S.pop()
if (S is empty) then return value
else {
    result = mistery(S)
    S.push(value)
    return result
}
```

The objective of this question is to discover the purpose of this algorithm. We start by executing the following commands.

```
S = new Stack();
S.push('1');
S.push('2');
S.push('3');
```

a) (4 points) Draw the content of the stack at this point.

Stack S:

```
```

b) (8 points) If we now execute

```
int x = mistery(S);
```

What is the value of x, and what is the content of the stack after the execution of the algorithm?

```
x = Stack S:
```

```
```

c) (4 points) In one sentence, explain what is this algorithm doing when given a stack S as input.

d) (4 points) Using the big-Oh notation, give the running time of the mistery algorithm if it is executed on a stack of n elements. No justification is needed.
Question 3 (15 points). Proofs by induction

Prove by induction on \( n \) that for every integer \( n \geq 0 \) and any real number \( a > 0 \), we have

\[
a^0 + a^1 + a^2 + \ldots + a^n = \frac{a^{n+1} - 1}{a - 1}.
\]

Base case:

Induction hypothesis:

Inductive step:
Question 4 (15 points). Recursive algorithms

Complete the pseudocode of the RecursiveSum algorithm below to obtain a recursive algorithm such that given a positive integer \( n \), it prints all the ways of expressing \( n \) as sums of positive integers. For example, given \( n=4 \), the output should look like this:

1+1+1+1=4
1+1+2=4
1+2+1=4
1+3=4
2+1+1=4
2+2=4
3+1=4
4=4

Note: This will be easier to do if we add, in addition to \( n \) itself, two additional arguments to the RecursiveSum algorithm:

• an array \( A \) large enough to store up to \( n \) elements, which will be used to accumulate partial sums through recursive calls.

• an integer \( \text{soFar} \) that keeps track of how many elements of \( A \) have been filled already.

Then, the result shown above would be obtained by calling RecursiveSum(\( A[\ ] \), 0, 4).

Algorithm RecursiveSum(A[ ], soFar, n)

Inputs: \( A[\] \) is an array of integers, where elements \( A[0,..., soFar-1] \) are already filled
\( n \) is an integer

Output: The algorithm prints out every possible ways to complete the partial sum already stored in \( A[0,\ldots,soFar-1] \) so that the numbers add up to \( n \).

\[
\]

if ( sumSoFar = n ) then print \( A[0] "+" A[1] "+" \ldots "+" A[soFar-1] "=\" n \\
else { /* WRITE YOUR PSEUDOCODE HERE */
\}
Question 5 (10 points). Big-Oh notation

Prove, using only the definition of the big-Oh notation, that \( \log (n^2 + 1) + n + 1 \) is \( O(n) \).
Question 6 (10 points). Solving recurrences

Using the substitution method, obtain an explicit formula for the following recurrence:

\[ T(n) = T(n-1) + 2n + 1 \quad \text{if } n > 0 \]
\[ 0 \quad \text{if } n = 0 \]
**Question 7 (15 points). Running time of algorithms**

Give the worst-case running time of the following algorithms, using the simplest Θ() notation (big-Theta notation) possible. No justification needed.

<table>
<thead>
<tr>
<th>Algorithm 1 (int n)</th>
<th>Θ() Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>i ← 2 * 2^n</td>
<td></td>
</tr>
<tr>
<td>while (i &gt; 1) do {</td>
<td></td>
</tr>
<tr>
<td>i ← i / 2</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 2 (int n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>for i = 1 to n do {</td>
</tr>
<tr>
<td>for j = 1 to 999 do {</td>
</tr>
<tr>
<td>print “Bazinga!”</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 3 (A[ ], int n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>for i = 0 to n-1 do { A[i]=i }</td>
</tr>
<tr>
<td>merge(A, 0, n/2, n-1)</td>
</tr>
<tr>
<td>pivot = partition(A, 0, n-1)</td>
</tr>
</tbody>
</table>

**Note:** merge and partition refer to the algorithms seen in class.
This page is left intentionally empty. You can use it for drafting your solutions.