Part 1: Multiple Choice

Enter your answers on the Scantron sheet.

We will not mark answers that have been entered on this sheet.

Each multiple choice question is worth 3.5 marks.

In all the questions below, $\log(x)$ means $\log_2(x)$. This fact might be useful to you: $\log(x^y) = y \log(x)$.

Note. When you are asked about the time complexity of an algorithm, the best (tightest) characterization of the order of the time complexity is implied. For example, it is correct to say that the time complexity $f_{LS}(n)$ of binary search is $O(n)$. However, even though $f_{LS}(n)$ is also $O(n^2)$, you should not say that the time complexity of linear search is $O(n^2)$.

1. Three programs $P_1$, $P_2$, and $P_3$ have time complexities $f_1(n)$, $f_2(n)$, and $f_3(n)$, respectively, such that
   - $f_1(n)$ is $O(f_2(n))$,
   - $f_2(n)$ is $O(f_1(n))$,
   - $f_1(n)$ is $O(f_3(n))$, and
   - $f_3(n)$ is not $O(f_1(n))$.

Which of the following statements is true?
(A) Program $P_1$ is faster than $P_2$ and $P_3$ for very large size inputs.
(B) Program $P_2$ is faster than $P_1$ and $P_3$ for very large size inputs.
(C) Program $P_3$ is faster than $P_1$ and $P_2$ for very large inputs.
(D) Program $P_3$ is slower than $P_1$ and $P_2$ for very large inputs.
(E) Program $P_1$ must have the exact same running time as $P_2$.

2. Consider the following pairs of functions: $f(n)$, $g(n)$. For which pair, the functions are such that $f(n)$ is $O(g(n))$ and $g(n)$ is not $O(f(n))$?
   (A) $f(n) = n^2$, $g(n) = 2n^2 + 7$
   (B) $f(n) = \log n$, $g(n) = \log(n^2)$
   (C) $f(n) = 17$, $g(n) = n + 1$
   (D) $f(n) = n$, $g(n) = 3000n + 1$
   (E) $f(n) = \log n$, $g(n) = 1/n$

3. Consider a hash table of size $M$ that initially stores $n$ keys. Assume that the hash function maps keys uniformly across the entire table and that separate chaining is used to resolve collisions. What is the worst case time complexity of the find($k$) algorithm invoked on this hash table when its parameter is a key $k$ that is not in the table? Assume that $n$ is much bigger than $M$.
   (A) $O(1)$ time
   (B) $O(n)$ time
   (C) $O(M)$ time
   (D) $O(n/M)$ time
   (E) $O(\log n)$ time
4. Consider the following algorithms, where $A$ is an array storing $n$ positive integer values.

```
Algorithm foo(A, n)
Input: Array A of size n
for k ← 0 to n - 1 do {
    if A[k] > 0 then foofoo(A, n)
}

Algorithm foofoo(A, n)
Input: Array A of size n
i ← 0
while i < n do {
    for j ← 0 to n - 1 do {
        i ← i + 1
    }
}
```

What is the worst case time complexity of algorithm $foo$?
(A) $O(n)$
(B) $O(nk)$
(C) $O(n^2)$
(D) $O(n^2i)$
(E) $O(n^3)$

5. Let $T$ be a proper binary tree with root $r$. Consider the following algorithm.

```
Algorithm traverse(r)
Input: Root r of a proper binary tree.
if r is a leaf then return 0
else {
    t ← traverse(left child of r)
    s ← traverse(right child of r)
    return s + t + 1
}
```

What does the algorithm do?
(A) It computes the height of the tree.
(B) It computes the number of nodes in the tree.
(C) It computes the number of nodes in the tree plus 1.
(D) It computes the number of leaves in the tree.
(E) It computes the number of internal nodes in the tree.
6. The following algorithm performs a traversal of a binary tree and returns some value based on the keys stored in the nodes.

```
Algorithm traverse2(r)
Input: Root r of a binary tree.
if r is a leaf then return key stored at r
else {
    a ← traverse2(left child of r)
    if a > key stored in r then a ← key stored in r
    b ← traverse2(right child of r)
    return a − b
}
```

Let the algorithm `traverse2` be executed on the following tree. What is the value returned by the algorithm?

√(A) −3
(B) −2
(C) 0
(D) 2
(E) 5

7. A set of n keys: \{k_1, \ldots, k_n\} is to be stored in an initially empty binary search tree. Which of the following statements is always true?

(A) The resulting binary search tree has the same height, regardless of the order in which the keys are inserted in the tree.
√(B) If k_i is the largest key, then in every binary search tree storing the above set of keys, the right child of the node storing k_i is a leaf.
(C) A preorder traversal of the tree visits the keys in increasing order of value.
(D) After inserting the keys, the key stored at the root of the tree is the same regardless of the order in which the keys are inserted.
(E) None of the above statements is always true.

8. Let T be a proper binary tree with 7 nodes: a, b, c, d, e, f, g.

A preorder traversal of T visits the nodes in this order: b, g, d, a, c, f, e.
An inorder traversal of T visits the nodes in this order: d, g, c, a, f, b, e.

Which node is at distance 3 from the root of T? (Hint. In tree T, node a is the right child of node g.)

(A) a
√(B) c
(C) d
(E) e
(F) g
Part 2: Written Answers

Write your answers directly on these sheets. You might find these facts useful: In a proper binary tree with \( n \) nodes the number of leaves is \( (n + 1)/2 \) and the number of internal nodes is \( (n - 1)/2 \).

9. [14 marks] Write in pseudocode an algorithm that receives as input the root of a tree and it returns \textbf{true} if the tree is a proper binary tree (i.e. each internal node has 2 children) and \textbf{false} otherwise. Assume that \( r.\text{children} \) is the number of children of a node \( r \).

For example, for the tree below with root \( A \) the algorithm must return the value \textbf{true}, and for the tree with root \( B \) it must return the value \textbf{false}.

\[
\text{Algorithm } \text{isProper}(r) \\
\text{Input: Root } r \text{ of a tree.} \\
\text{Output: True is the tree is proper binary; false otherwise} \\
\text{if } r \text{ is a leaf then return true} \\
\text{else if } r.\text{children} \neq 2 \text{ then return false} \\
\text{else if } \text{isProper(left child of } r) = \text{false} \text{ then return false} \\
\text{else return } \text{isProper(right child of } r) \\
\]

10. [3 marks] Compute the time complexity of the above algorithm in the worst case as a function of the number \( n \) of nodes and give its order.

[4 marks] Explain what the worst case for the algorithm is and how you computed the time complexity.

The worst case for the algorithm is when the tree is proper binary, as then the algorithm does not terminate early and thus every node of the tree needs to be visited. Ignoring recursive calls, the algorithm performs a constant number \( c \) of operations per leaf and a constant number \( c' \) of operations per internal node of the tree. Since the recursive calls make the algorithm traverse the entire tree visiting every node once, then the total number of operations performed by the algorithms is

\[
c \times \text{number of leaves} + c' \times \text{number of internal nodes} = c \times \frac{n + 1}{2} + c' \times \frac{n - 1}{2} \text{ is } O(n)
\]
11. [14 marks] Given an array $A$ storing $m$ integer values and an array $B$ storing $n$ integer values, write in pseudocode an algorithm $\text{subarray}(A, B, m, n)$ that returns the value $\text{true}$ if $A$ is a sub-array of $B$ and it returns $\text{false}$ otherwise. $A$ is a sub-array of $B$ if there is a value $0 \leq j \leq n - m$ such that $A[0] = B[j], A[1] = B[j + 1], \ldots, A[m] = B[j + m - 1]$, i.e., if the values of $A$ appear in consecutive positions of $B$. For example, for the arrays $A$ and $B$ shown below the algorithm must return the value $\text{true}$, but for the arrays $A'$ and $B$, the algorithm must return the value $\text{false}$.

\[
A = \begin{bmatrix} 8 & 4 & 7 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 4 & 7 & 5 & 8 & 4 & 7 & 1 & 5 & 5 \end{bmatrix} \quad A' = \begin{bmatrix} 1 & 5 & 8 \end{bmatrix}
\]

**Algorithm** $\text{subarray}(A, B, m, n)$

**Input:** Array $A$ of size $m$ and array $B$ of size $n$

**Output:** True if $A$ is a subarray of $B$ and false otherwise

for $i_B \leftarrow 0$ to $n - m$ do {

\[i_A \leftarrow 0\]

while ($i_A < m$) and $A[i_A] = B[i_B + i_A]$ do

\[i_A \leftarrow i_A + 1\]

if $i_A = m$ then return true

} return false

12. [3 marks] Compute the worst case time complexity of the above algorithm as a function of $m$ and $n$ and give its order.

[3 marks] Explain what the worst case of the algorithm is and how you computed the time complexity.

The worst case for the algorithm is when all values stored in $B$ are the same, say all of them are 1, and all values in $A$ are also 1 except the last one of them. This will ensure that the condition of the if statement in the algorithm is never true, so the algorithm does not terminate early. Also, note that since the first $m - 1$ values in $A$ are the same as those in $B$, then every time that the while loop is executed, it will iterate $m$ times (the $m$-th time the second condition will be false).

Each iteration of the while loop performs a constant number $c$ of operations and the loop is repeated $m$ times, so the total number of operations performed by the loop is $mc$. Outside the while loop, but inside the for loop an additional constant number $c'$ of operations is performed. So, every iteration of the for loop performs $c' + cm$ operations. Since in the worst case the for loop performs $n - m$ iterations, and outside the for a constant number $c''$ of operations is performed, the total number of operations performed by the algorithm is $c'' + (n - m)(c' + cm)$ is $O(mn)$. 
13. Consider the following algorithm. \( A \) is an array of size \( n \).

**Algorithm rec\((n)\)**

**In:** Integer value \( n \).

\[
\text{if } n = 0 \text{ then return } 1 \\
\text{else } \\
i \leftarrow \text{rec}(n - 1) \\
A[n] \leftarrow i \\
\text{return } i
\]

[6 marks] Write a recurrence equation for the time complexity of this algorithm.

If \( n = 0 \) the algorithm performs a constant number \( c' \) of operations; if \( n > 0 \) the algorithm performs a constant number \( c \) of operations, if we ignore the recursive calls. Hence, the following recurrence equation characterizes the time complexity of the algorithm:

\[
f(0) = c' \\
f(n) = c + f(n - 1) \text{ if } n > 0
\]

[6 marks] Solve the above recurrence equation by repeated substitution and give the order of the time complexity.

\[
f(n) = c + f(n - 1) \\
f(n - 1) = c + f(n - 2) \\
f(n - 2) = c + f(n - 3) \\
\vdots \\
f(1) = c + f(0) = c + c'
\]

Hence, \( f(n) = c + c + c + \cdots + c + c' = nc + c' \) is \( O(n) \).
14. Consider a hash table of size 7 with hash function \( h(k) = k \mod 7 \). Draw the hash table after inserting in it, in the given order, the following values into the table: 14, 28, 2, 26, and 70:

[3 marks] (a) when linear probing is used to resolve collisions

[7 marks] (b) when double hashing with secondary hash function \( h'(k) = 5 - (k \mod 5) \) is used to resolve collisions.

<table>
<thead>
<tr>
<th>Linear probing</th>
<th>Double hashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 14</td>
<td>0 14</td>
</tr>
<tr>
<td>1 28</td>
<td>1</td>
</tr>
<tr>
<td>2 2</td>
<td>2 28</td>
</tr>
<tr>
<td>3 70</td>
<td>3 70</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5 26</td>
<td>5 2</td>
</tr>
<tr>
<td>6</td>
<td>6 26</td>
</tr>
</tbody>
</table>

15. [3 marks] Consider the following binary search tree. Insert the keys 15 and 10 (in that order) into the tree and show the resulting tree. You must use the algorithms described in class for inserting data into a binary search tree.
16. [6 marks] Consider the following binary search tree. Remove the key 13 from the tree and draw the resulting tree. Then remove the key 5 from this new tree and show the final tree (so in the final tree both keys, 13 and 5, have been removed). You must use the algorithms described in class for removing data from a binary search tree.
# CS2210A Midterm Examination

November 4, 2014

9 pages, 16 questions

110 minutes

<table>
<thead>
<tr>
<th>PART I</th>
<th>PART II</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>

**Last Name:**

**First Name:**

**Student Number:**

---

**Instructions**

- Write your name and student number on the space provided.
- Please check that your exam is complete. It should have 9 pages and 16 questions.
- The examination has a total of 100 marks.
- When you are done, call one of the TA’s and he will pick your exam up.