International University of Sarajevo  
Faculty of Engineering and Natural Science  
Algorithms and Data Structures  
CS 302 Final Exam

Lecturer Dr. Suleiman Abu Kharmeh  
Time Limit: 120 Minutes  
Date: 21/January/15  
Student Name: ________________________

1. (10 points) What is the runtime, assuming that the worst case executions of selected algorithms (in steps) are:

(a) (2 points) $400 + 21n^2 + 8n^3$  
(b) (2 points) $100000n^3 + 4293n^2 + 9400n$  
(c) (2 points) $50 + 30n^3 + \log(n)$  
(d) (2 points) $\log(\log(n)) + 99.99n^3$  
(e) (2 points) $33.3n^3 + n\log(n)$

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>105</td>
<td></td>
</tr>
</tbody>
</table>

2. (20 points) Assuming a Java array of integers (called a) was ordered in decreasing order.

(a) (5 points) List two searching algorithms and explain briefly (20 words maximum) how they work.
(b) (5 points) What is the runtime for each algorithm?

(c) (5 points) Which of your two algorithms is faster for searching the $a$ array described above and why? Assuming an array size of $1024^2$, how many iterations (worst case) would each algorithm need to perform before finding an item? (hint: runtime).

(d) (5 points) Write the Java code for one of the two algorithms.
3. (16 points) Define the following (20 words maximum per definition)
   (a) (2 points) Tree

   (b) (2 points) Binary tree

   (c) (2 points) Binary search tree

   (d) (2 points) Full binary search tree

   (e) (2 points) Complete binary search tree

   (f) (2 points) Perfect binary search tree

   (g) (2 points) Balanced binary search tree

   (h) (2 points) B-Trees
4. (10 points) Insert the following keys into an initially empty 5-way B-Tree

    a g f b k d h m j e s i r x c l n t u p

5. (12 points) Answer the following questions about heaps:
   
   (a) (3 points) What is a binary heap?

   (b) (3 points) In which ADT implementation would it typically be used?

   (c) (3 points) A binary heap can be implemented as an array such that:
   For any element in array position \( i \):
   
   - The left child is in position \( 2i \).
   - The right child is in the cell after the left child \((2i + 1)\).
   - The parent is in position \( \lfloor i/2 \rfloor \)

   Using the above description, draw the tree that is represented by the following array:

   \[
   \begin{array}{ccccccccccccc}
   J & I & H & G & F & E & D & C & B & A & \ldots & \ldots & \ldots \\
   0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14
   \end{array}
   \]
(d) (3 points) Write the Java method that would find the maximum value in the heap described by the above array.

6. (5 points) **(bonus)** Explain briefly what would you change about this course?

7. (32 points) Write a complete "Balanced Binary Search Tree" class in Java. As a guidance, all the figures demonstrating the rotation methods as well as a summary of all the rotation operations are included below. Your class should include:

(a) (4 points) All the necessary methods and private variables related to a binary search tree. E.g. references to children, constructors, ...

(b) (4 points) GetMax method which returns the node with the maximum element.

(c) (4 points) GetMin method which returns the node with the minimum element

(d) (4 points) Single left rotation method.

(e) (4 points) Single right rotation method.

(f) (4 points) Double rotation methods (RightLeft, LeftRight).

(g) (4 points) A GetBalance method which calculates the balance of a node using a recursive GetHeight method which you also need to implement.

(h) (4 points) A generic Add method which adds a node to the tree and immediately balances the tree. It should use a number of the methods declared so far (i.e. GetBalance, and some rotation methods).
Single Left Rotation

Single left rotation about LC

Single right rotation about P

Double right rotation implemented by two single rotations.
<table>
<thead>
<tr>
<th>Type of imbalance</th>
<th>Balance factor of parent</th>
<th>Balance factor of child</th>
<th>Direction of 1st rotation</th>
<th>Direction of 2nd Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-left</td>
<td>-2</td>
<td>-1</td>
<td>Right</td>
<td>NA</td>
</tr>
<tr>
<td>Right-right</td>
<td>+2</td>
<td>+1</td>
<td>Left</td>
<td>NA</td>
</tr>
<tr>
<td>Left-right</td>
<td>-2</td>
<td>+1</td>
<td>Left (child)</td>
<td>Right (node)</td>
</tr>
<tr>
<td>Right-left</td>
<td>+2</td>
<td>-1</td>
<td>Right (child)</td>
<td>Left (node)</td>
</tr>
</tbody>
</table>