ICS/CSE 46 Midterm Exam

Name: SOLUTIONS
Student ID: SOLUTIONS

Read all of the following information before starting the exam:

- Show all work, clearly and in order, if you want to get full credit.
- Circle or otherwise indicate your final answers.
- Please keep your written answers brief; be clear and to the point. I may take points off for rambling and for incorrect or irrelevant statements.
- This test has 8 problems on 9 total pages, including this one, the score sheet page, and a scratch page. It is your responsibility to make sure that you have all of the pages!
- Each problem is worth 10 points for a total of 80 points on this test.
- If you are going to write a portion of an answer on the back of a page, clearly indicate you are doing so on the front of that same page.
- Good luck!
Score sheet

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>
1. Please clearly mark the following statements as True or False. Do not write a single letter; spell the words out.

(a) Inserting the numbers 4, 2, 1, 3, 6, 5, 7 into a binary search tree in that order gives a perfectly balanced binary search tree.  

   TRUE

(b) Because appending to an array list is \(\Theta(n)\) worst case time, starting with an empty array list and appending \(n\) items takes \(\Theta(n^2)\) time.

   FALSE

(c) An \(\Theta(n)\) time algorithm will always be faster than a \(\Theta(n \log n)\) time algorithm.

   FALSE

(d) \(\sum_{i=0}^{n} 2^i = O(2^n)\)

   TRUE

(e) A perfectly balanced tree is also AVL balanced.

   TRUE

(f) An AVL balanced tree is also perfectly balanced.

   FALSE

2. Answer the following questions using \(O\)-notation.

(a) What is the worst case runtime of insertion sort on a list with \(n\) elements?

   \(O(n^2)\)

(b) What is the worst case runtime of insertion sort on a list with \(n\) elements and \(k\) inversions?

   \(O(n + k)\)

(c) What is the worst case runtime of merge sort on a list with \(n\) elements?

   \(O(n \log n)\)

(d) What is the worst case runtime of merge sort on a list with \(n\) elements and \(k\) inversions?

   \(O(n \log n)\)
3. For each pair of functions below circle the true statements.

(a) \( f(n) = 500n^2 \) and \( g(n) = n^3 - 3n + 2 \)
\[ f(n) = O(g(n)) \]

(b) \( f(n) = \sqrt{n} \) and \( g(n) = \log n \)
\[ f(n) = \Omega(g(n)) \]

(c) \( f(n) = n \) and \( g(n) = 2^{n/2} \)
\[ f(n) = O(g(n)) \]

4. (a) If a hash table with initial capacity of 10 is dynamically resized to double its current size whenever the load factor threshold passes 1.5 and 55 items are inserted, how large will the capacity of the hash table be after the insertions?

It will reach a capacity of 40.

(b) While implementing his hash table, Bob forgot to implement resizing. What will happen if a large number of elements are inserted if Bob used linear probing?

Because linear probing only allows one element per slot in the underlying array, the hash table will be unable to handle inserting an element after all of the slots are filled.

What will happen if a large number of elements are inserted if Bob used separate chaining?

In separate chaining colliding elements are chained together in a linked list. So as more and more elements are inserted the linked lists will grow longer and the hash table will slow down.
5. (a) What is the pre-order traversal of the above tree?

RMCOUPETECSINEC

(b) What is the in-order traversal of the above tree?

COMPUTERSCIENCE

(c) What is the post-order traversal of the above tree?

OCPTEUMSICCCENER
6. (a) Draw the result of inserting 5 into the above AVL tree.

(b) Draw the result of deleting 23 from the above binary search tree.

Moving 25 to the root was also acceptable.
7. Radix sort is being run on the following list $L$ using a base of 10. Fill in the blank lists with the state of $L$ after each pass of bucket sort.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>437</td>
<td>124</td>
<td>901</td>
<td>182</td>
<td>746</td>
</tr>
</tbody>
</table>

(a) After the first pass of bucket sort:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>182</td>
<td>124</td>
<td>746</td>
<td>437</td>
</tr>
</tbody>
</table>

(b) After the second pass of bucket sort:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>124</td>
<td>437</td>
<td>746</td>
<td>182</td>
</tr>
</tbody>
</table>

(c) After the final pass of bucket sort:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>182</td>
<td>437</td>
<td>746</td>
<td>901</td>
</tr>
</tbody>
</table>
8. Give pseudocode for or precisely describe in words an algorithm to decide if a binary rooted tree is a binary search tree. Full credit will be awarded for an \(O(n)\) time solution.

First solution:
Perform an inorder traversal of the tree. Verify that the traversal is in sorted order. If the traversal is not in sorted order the tree can not be a binary search tree and if it is in sorted order than the tree is a binary search tree.

Second solution:
Return \(is\_bst(root, -\infty, \infty)\).

```python
def is_bst(n, min, max):
    if n == nullptr:
        return true
    if n->data < min or n->data > max:
        return false
    return is_bst(n->left, min, n->data) and is_bst(n->right, n->data, max)
```