I&C SCI 46  Diagnostic Exam 2, Fall 2022

DO NOT OPEN EXAM UNTIL INSTRUCTED TO DO SO

SILENCE MOBILE PHONE AND OTHER DEVICES

This is a diagnostic exam intended to help you evaluate your readiness for the real exam.

Write the following information clearly. You may write this information only before the instructor calls to begin the exam. You may not write this information after the instructor calls to stop writing.

Name: ____________________________________________________________

UCI Email Address: ____________________________________________@uci.edu

UCI Student ID #: __________________________________________________

Read and understand the following rules; failure to abide by these rules, or directions given by course staff during the exam, may result in disciplinary action, including but not limited to a failing grade in the class.

• This exam is solely for students enrolled in this lecture. Anyone not enrolled in this lecture may not take an exam.

• Keep your UCI ID readily accessible during the test. Proctors may request to see it.

• This exam is closed book, closed notes, and is individual effort. Once course staff begin passing out exams, you may not communicate with anyone other than proctors for any reason, nor may you have electronics, including calculators, watches, and phones, available to you during the test for any reason.

YOU DO NOT NEED A CALCULATOR!

• If you leave your seat during the test for any reason, your instructor may collect it and deem you to have turned it in. Do not ask proctors for an exemption to this, they are not authorized to grant such.

• If you are still seated at X:35 AM at the real quiz, you may not leave your seat until explicitly dismissed by the instructor. Leaving after X:35 AM and before being dismissed may result in a penalty.

• You must take the exam in your assigned seat unless the professor (not a TA) tells you otherwise. You may not open the exam until explicitly told to do so by the professor. The instructor will call to cease writing at X:45 AM, at which point you must immediately cease writing and close the exam. You may not write any further at that point, including finishing one’s current sentence.

• If you believe a question is ambiguous, write at least two reasonable interpretations and indicate clearly which one you will be using. Then answer your question with that assumption. Unless your interpretation makes the problem much more trivial than intended, we will grade your response as if one of us had made that clarification.

• The purpose of the real exam is to evaluate how well you understand the material presented in the course. It is an academic integrity violation to do anything that subverts the goals of this assessment including, but not limited to, not doing your own work or submitting that of anyone else.

• Write your answers in the space provided for each question.

• Write your UCI email at the top of each answer page. You may not do this until the exam has begun. There is one point for doing this.
1. Suppose we have a Cuckoo Hash Table with each table having room for \( m = 11 \) entries each. Our hash functions are \( h_0(x) = x \% 11 \) and \( h_1(x) = (x/11) \% 11 \), where the / is integer division (floor of division; discard remainder). For example, \( h_0(1289) = 2 \) and \( h_1(1289) = 7 \). We insert the keys 46, 51, 84, 200, 134, 138, 52, 184, 181, 179, 25, 28, 36, 72, 85, in that order, into the Hash Table, in the manner described in lecture.

For your convenience, here are the hash values:

<table>
<thead>
<tr>
<th>( x )</th>
<th>46</th>
<th>51</th>
<th>84</th>
<th>200</th>
<th>134</th>
<th>138</th>
<th>52</th>
<th>184</th>
<th>181</th>
<th>179</th>
<th>25</th>
<th>28</th>
<th>36</th>
<th>72</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

(a) (1 point) After the first five keys are inserted, are they in the upper array (indexed by \( h_0 \)) or the lower array (indexed by \( h_1 \)) after all of first five keys have been inserted? Circle your choice on each clearly.

(b) (1 point) Which value will be the last one inserted successfully into the table prior to the rehash/resize? Recall that we are rehashing / resizing only after an element cannot be inserted; there is no \textit{a priori} maximum eviction length as there was in project three.

2. (3 points) Describe how to perform a removal from a hash table that uses linear probing to resolve collisions where we do not use a special marker to represent deleted elements. That is, we must rearrange the contents so it appears that the removed entry was never inserted in the first place.
Circle clearly your answers for 1(a) here. **Note that they are listed in numerical order, not the order in which they were inserted. Answers that depend on an insertion order other than that which we listed will not earn credit.**

<table>
<thead>
<tr>
<th>Number</th>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>upper</td>
<td>lower</td>
</tr>
<tr>
<td>51</td>
<td>upper</td>
<td>lower</td>
</tr>
<tr>
<td>84</td>
<td>upper</td>
<td>lower</td>
</tr>
<tr>
<td>134</td>
<td>upper</td>
<td>lower</td>
</tr>
<tr>
<td>200</td>
<td>upper</td>
<td>lower</td>
</tr>
</tbody>
</table>

Write your answer for 1(b) in this box:

```
1(b)
```

Write your answer to question 2 here. Note that your answer might not use the entire space provided.
3. (1 points) Consider the following AVL Tree. We saw in lecture that an insert operation will cause at most one re-balance operation. We also saw that a delete operation could take more than that. Which key can I delete in the following tree to cause two separate re-balancing operations? Do not select a node that has two children for your answer. You will get a zero for this question if you do so.

4. (2 points) In project 4, you did not have to write a delete on your AVL tree. You’re going to write it here.

You may assume every node in the tree stores a Key, Value pair, along with left, right, and parent pointers. You also have the following helper functions already written and tested (you may call them and assume they work):

- Node * find(Key k) returns a pointer to a Node that has the given key. It returns nullptr if the key is not in the tree.
- Node * rebalance(Node * z), which performs the local rebalancing, given that z is the lowest unbalanced node. This is the same definition of z we used in lecture.
- Node * localDelete(Node * n), which deletes the given node, freeing all relevant memory. It returns a pointer to the “replacement” (same child of same parent) if a non-leaf with one child is deleted, the parent if a leaf is deleted, or the actually-deleted node if the parameter had two children (after moving the key/values appropriately). This is the “normal BST” localDelete, and does not do any height re-balancing.
- int height(Node *n), which returns the height of the given node. That is, the number of hops to fall off the farthest leaf underneath it. If the parameter is nullptr, it returns -1.

While we will not require strict C++, you should write your code in such a way as to make it very clear to the grader that you could implement it correctly from your ideas.

void MyAVLTree<Key, Value>::delete(Key k)
What is your answer for question 3? Write the key in this box:

Write your code for question 4 here. Note that your answer might not use the entire space provided.

```cpp
void MyAVLTree<Key, Value>::delete(Key k)
```
5. (2 points) Consider the following Binary Search Tree, being balanced via the rules of AVL-Trees. We are examining it in the middle of the “insert” operation. A Key has been inserted into the tree, but we have not (yet) performed any rotations.

(a) Which key did we just insert?
(b) In setting up the update operation, what is the key stored at $z$, as described in lecture?
(c) In setting up the update operation, what is the key stored at $y$, as described in lecture?
(d) In setting up the update operation, what is the key stored at $x$, as described in lecture?

6. (1.5 points) Suppose we have a hash table of size $m = 13$, implemented with quadratic probing (defined here to mean that the $i$th choice is $(h(k) + i^2)\%m$, with the initial choice being $i = 0$) and with a hash function of $h(k) = k\%m$. The table stores unsigned values. Insert the following values into the hash table in the order they are given. Do not resize the hash table, regardless of load factor. The values to insert are: 38, 39, 26, 25, 16, 14, 22, 9 (in the order listed).

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. (2 points) I have a binary tree (not a binary search tree) with the following properties:

- Each non-null node of $T$ contains a single character
- An in-order traversal of the tree reads “YXBJSPWTHGMR”
- A post-order traversal of the tree reads “YXSJTWPBGMRH”

*This is a reminder that this is not a binary search tree.*

Give a pre-order traversal of the tree. It may help for you to draw the tree, although that is not required (and should not go on the answer sheet).
What is your answer for question 5?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5(a)</td>
<td>5(b)</td>
<td>5(c)</td>
<td>5(d)</td>
</tr>
</tbody>
</table>

Fill in your answer for question 6 here.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 7. Write only the relevant characters in this box. You may use the rest of the space below the dotted line but outside the box as scratch paper if you wish; nothing in that region will be graded, however.