INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this question paper until you are told to do so.

2. Quiz 1 is run on two large rooms in NUS, at the same time:
   Students with full name starting with ‘A’ until ‘L’ will go to LT19.
   Students with full name starting with ‘M’ until ‘Z’ will go to LT15.

3. This question paper contains THREE (3) sections with sub-questions.
   Each section has different length and different number of sub-questions.
   It comprises TWELVE (12) printed pages, including this page.

4. Write all your answers in this question paper, **but only in the space provided**.
   You can use either pen or pencil. Just make sure that you write legibly!
   Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.

5. This is an **Open Book Examination**. You can check the lecture notes, tutorial files, problem set files, Steven’s ‘Competitive Programming’ book, or any other books that you think will be useful. But remember that the more time that you spend flipping through your files implies that you have less time to actually answer the questions.

6. When this Quiz 1 starts, **please immediately write your Matriculation Number here:**
   ________________ (do not forget the last letter) and your Tutorial Group: _____
   (If you forget your tutorial group number, that’s ok, just put your tutorial time slot.)
   But **do not** write your name in order to facilitate unbiased grading.

7. All the best :).
   After Quiz 1, this question paper will be collected, graded manually in 2 weeks over recess week, and likely returned to you via your Tutorial TA on Week07.
This page is intentionally left blank. You can use it as ‘rough paper’.
1 CS1020 stuff and basic BST/AVL/Heap Application (12 Marks)

Grading scheme: 0 (no answer or totally wrong), 1 (the final answer has mistake(s)), 3 (the final answer is correct).

Q1. Consider the following functions: (3 marks)

\[ \begin{align*}
\sqrt{n} & \quad n^2 & \quad n \log n \\
2^{n} & \quad n^2 + \log n & \quad n^{1/3} + \log n & \quad n \log n^2
\end{align*} \]

Group these functions so that \( f(n) \) and \( g(n) \) are in the same group if and only if \( f(n) = O(g(n)) \) and \( g(n) = O(f(n)) \). List down the groups and the members in each group in the space provided below.

Q2. List down all the mistakes that you can find in the following binary search algorithm (written in pseudocode): (3 marks)

function search(A,x,l,r) // find x in array A[l..r] where A start from index 0
if (l >= r)
    return not found
else {
    m = ceil((l+r+1)/2)
    if (x < A[m])
        return search(A,x,l,m-1)
    else if (x > A[m])
        return search(A,x,m+1,r)
    else
        return A[m]
}
Q3. Give an example of the smallest AVL tree (in terms of number of vertices) of height $= 4$ (to avoid ambiguity, we define an AVL tree with just one root vertex as having height $= 0$). Just give the structure of the AVL tree, you don’t have to provide the values for each vertex. (3 marks)

Q4. Given the integers $\{1, 10, 20, 30, 40, 50, 70, 80, 90, 100\}$, show a modified max heap containing these integers where in addition to the heap property, we have for each vertex $i$ that is not a leaf, the largest element in subtree rooted at $i.left <$ the smallest element in subtree rooted at $i.right$. (3 marks)
2 Analysis (15 marks)

Prove (the statement is correct) or disprove (the statement is wrong) the following statements below. If you want to prove it, provide the proof (preferred) or at least a convincing argument. If you want to disprove it, provide at least one counter example. 3 marks per each statement below (1 mark for saying correct/wrong, 2 marks for explanation):

Note: You are only given a small amount of space below (i.e. do not write too long-winded answer)!

1. In a max heap which is a full binary tree of more than 1 vertex, the 2nd smallest item must be a leaf. {Note: A full binary tree is one where every vertex other then the leaves has 2 children.}

2. The total number of vertices $T$ in a binary tree with $m$ leaves, where $m \geq 1$, is such that $2 \leq T \leq 2m - 1$.

3. Given any BST tree of size $n$, for any vertex $i$ in the tree, in order to find its successor, we need to compare with at most $O(\log n)$ other vertices.

4. Inserting $k$ items into a sorted array of $n$ items will always have a time complexity greater than that of inserting $k$ items into an AVL tree of $n$ items.
5. Every AVL tree insert operation only requires at most 1 rebalancing operation (1 of the 4 cases as outlined in lecture 3) to maintain the AVL property for the whole tree.

3 Application Questions (23 Marks)

Please write pseudo-code for all your algorithms. There should not be any black-boxes in your pseudo-code. This means that if you use any function that is not defined in the lecture notes, then you must define them.

Q1. Modify a bit: (8 marks)

a.) Give an efficient algorithm to determine if two max heaps which contains integers from [1..1000000] have the same content. Assume that values in the heaps are all unique. (4 marks)
b.) Two trees $T_1$ and $T_2$ are isomorphic if $T_1$ can be transformed into $T_2$ (or vice versa) by swapping left and right children of (some of the) vertices in $T_1$. For example the two trees below are isomorphic because they are the same if the children of $a$, $b$, and $g$ but not the other vertices are swapped. Give a polynomial time algorithm to decide if two trees are isomorphic. What is the running time of your algorithm? (4 marks)
Q2. Crank up the brain juices: (8 marks)

Design an algorithm to find the median in a bBST of $n$ elements as efficiently as possible. The median of $n$ elements is defined as follows: If $n$ is odd, median = $((n+1)/2)^{th}$ element. If $n$ is even, median = $[(n/2)^{th}$ element + $(n/2 + 1)^{th}$ element]/2. (Hint: The best algorithm can do this in $O(\log n)$ time. If your algorithm has a time complexity larger than this, then you will only get partial marks).
–You can continue writing your answer for Q2 here–
Q3. Crank up even more brain juice: (7 marks)

Given a set of integers, design an algorithm to get the $k^{th}$ smallest element as efficiently as possible if we are restricted to only the min heap data structure. (Hint: The most efficient algorithm can do this in $O(k \log k)$ time. If your algorithm has a time complexity larger than this, then you will only get partial marks).
--You can continue writing your answer for Q3 here--
Candidates, please do not touch this table!

<table>
<thead>
<tr>
<th>Section</th>
<th>Maximum Marks</th>
<th>Student’s Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>50</strong></td>
<td></td>
</tr>
</tbody>
</table>