CS2040S — Data Structures and Algorithms
School of Computing
National University of Singapore

Midterm Assessment

6 Mar 2023 **Time allowed:** 2 hours

Instructions — please read carefully:

1. Do not open the midterm until you are directed to do so.
2. Read **all** the instructions first.
3. The quiz is closed book. You may bring one double-sided sheet of A4 paper to the quiz. (You may not bring any magnification equipment!) You may NOT use a calculator, your mobile phone, or any other electronic device.
4. The **QUESTION SET** comprises **EIGHT (8) questions** and **TWENTY (20) pages**, and the **ANSWER SHEET** comprises of **TEN (10) pages**.
5. The time allowed for solving this test is **2 hours**.
6. The maximum score of this test is **100 marks**. The weight of each question is given in square brackets beside the question number.
7. All questions must be answered correctly for the maximum score to be attained.
8. All questions must be answered in the space provided in the **ANSWER SHEET**; no extra sheets will be accepted as answers.
9. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.
10. An excerpt of the question may be provided above the answer box. It is to aid you to answer in the correct box and is not the exact question. You should refer to the original question in the question booklet.
11. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).
12. Unless otherwise stated in the question, when we ask for the worst-case big-O running time of an algorithm we always mean to give the tightest possible answer.
13. Unless otherwise stated in the question, we will assume that operators behave as per Java (e.g., 5/2 will evaluate to 2). However, pseudocode does not necessarily satisfy Java syntax (unless stated otherwise) and things that do not compile as legal Java are not necessarily bugs (as long as they are clear pseudocode).
14. Unless otherwise stated in the question, we are not concerned with overflow errors, e.g., storing the value $2^{245,546,983}$ in an integer.

GOOD LUCK!

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: Name Jumble [12 marks]

The first column in the table below contains an unsorted list of words. The last column contains a sorted list of words. Each intermediate column contains a partially sorted list.

Each intermediate column was constructed by beginning with the unsorted list at the left and running one of the sorting algorithms that we learned about in class, stopping at some point before it finishes. Each algorithm is executed exactly as described in the lecture notes. One column has been sorted using a fake sorting algorithm. (Recursive algorithms recurse on the left half of the array before the right half. QuickSort uses the first element as the pivot and uses in-place 2-way partitioning.)

Unsorted	**A**	**B**	**C**	**D**	**E**	**F**	**Sorted**
Ken	Alice	Elly	Alice	Ori	Alice	Alice	Alice
Mo	Gia	Gia	Bob	Mo	Gia	Gia	Bob
Gia	Hal	Alice	Carol	Ned	Ken	Hal	Carol
Alice	Jo	Hal	Dan	Ina	Mo	Jo	Dan
Hal	Ken	Jo	Hal	Hal	Hal	Ina	Elly
Jo	Mo	Dan	Jo	Ken	Ina	Fanny	Fanny
Ned	Ned	Ina	Ned	Lily	Jo	Carol	Gia
Ina	Ina	Fanny	Ina	Elly	Ned	Bob	Hal
Fanny	Fanny	Carol	Fanny	Fanny	Fanny	Ken	Ina
Carol	Carol	Bob	Gia	Carol	Carol	Mo	Jo
Bob	Bob	Ken	Mo	Bob	Bob	Dan	Ken
Ori	Ori	Ori	Ori	Gia	Ori	Lily	Lily
Pat	Pat	Pat	Pat	Jo	Pat	Elly	Mo
Dan	Dan	Ned	Ken	Dan	Dan	Ned	Ned
Lily	Lily	Lily	Lily	Alice	Lily	Ori	Ori
Elly	Elly	Mo	Elly	Pat	Elly	Pat	Pat
Unsorted	**A**	**B**	**C**	**D**	**E**	**F**	**Sorted**

Identify, below, which column was (partially) sorted with which of the following algorithms:

1. BubbleSort
2. SelectionSort
3. InsertionSort
4. MergeSort
5. QuickSort (first element pivot)
6. None of the above.

Hint: Do not just execute each sorting algorithm, step-by-step, until it matches one of the columns. Instead, think about the invariants that are true at every step of the sorting algorithm.

A. What sort was used on Column A? [2 marks]

B. What sort was used on Column B? [2 marks]

C. What sort was used on Column C? [2 marks]

D. What sort was used on Column D? [2 marks]

E. What sort was used on Column E? [2 marks]

F. What sort was used on Column F? [2 marks]

Question 2: Asymptotically Approaching Answers [18 marks]

A. Choose the tightest possible bound from the available options for the following function:

$$T(n) = 17n\log^2(n^2)$$

1. $O(\log n)$
2. $O(n)$
3. $O(n\log n)$
4. $O(n^2)$
5. $O(n^3)$
6. $O(2^n)$

[2 marks]

B. Choose the tightest possible bound from the available options for the following function:

$$T(n) = 4.2n\sqrt{n} + \frac{17n}{\log^2(n)}$$

1. $O(n)$
2. $O(n\log n)$
3. $O(n\log^2 n)$
4. $O(n^2)$
5. $O(n^3)$
6. $O(2^n)$

[2 marks]

C. True or false: $2^{\log^2 n} = O(n^{17})$

1. True
2. False

[2 marks]

D. Choose the tightest possible bound from the available options for the following recurrence, assuming that $T(1) = 1$:

$$T(n) = T(n/3) + 7n$$

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n\log n)$
5. $O(n\log^2 n)$
6. $O(n^2)$
7. $O(n^3)$
8. $O(2^n)$

[3 marks]

E. Choose the tightest possible bound from the available options for the following recurrence, assuming that $T(1) = 1$:

$$T(n) = 3T(n/3) + 3n$$

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n \log^2 n)$
6. $O(n^2)$
7. $O(n^3)$
8. $O(2^n)$

[3 marks]

F. What is the asymptotic running time of the following code, as a function of n, when you execute `doubleLoopy(n)`? (Give the tightest bound possible of the available options.)

```
public int doubleLoopy(int n){
    int k = 0;
    for (int i=0; i<n/2; i++)
        k = k + 1;
    if (k>1) {
        doubleloopy(k);
        doubleloopy(k);
    }
    return 42;
}
```

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n^2)$
6. $O(n^3)$
7. $O(2^n)$

[3 marks]

G. Which recurrence best describes the runtime of the following algorithm? (Assume that $A[u \ldots v]$ creates a new array containing a copy of only the slots in the range $[u, v]$. Do not worry about the cost of creating or copying the array, or about the correctness of the algorithm.)

```
loopyloop(A[0..n-1]){
    if (n < 17) return 42;
    int count = 0;
    int total = 0;
    int x = n/2;
    for (int i = 0; i < n; i++)
        for (int j=i; j<n; j++)
            count = count+1;
            if (count == x/2)
                total = total + loopyloop(A[0..x/2]);
            if (count == n-1)
                total = total + loopyloop(A[3*x/2..n-1]);
    return count;
 }
```

1. $T(n) \leq T(n/2) + O(n)$.
2. $T(n) \leq T(n/2) + O(n^2)$.
3. $T(n) \leq 2T(n/2) + O(n^2)$.
4. $T(n) \leq T(n/4) + O(n)$.
5. $T(n) \leq T(n/4) + O(n^2)$.
6. $T(n) \leq 4T(n/4) + O(n)$.
7. $T(n) \leq 2T(n/4) + O(n^2)$.
8. None of the above.

[3 marks]

Question 3: How fast is it? [14 marks]

A. What is the worst-case running time of InsertionSort on an array of size n containing only 2 different elements (e.g., an array containing only the digits 0 and 1):

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n \log^2 n)$
6. $O(n^2)$
7. $O(n^3)$
8. $O(2^n)$

[2 marks]

B. What is the running time of MergeSort (as described in class) on an array in which the first half and the second half are already sorted, e.g., an array like $\{2,4,6,8,1,3,5,7\}$:

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n \log^2 n)$
6. $O(n^2)$
7. $O(n^3)$
8. $O(2^n)$

[2 marks]

C. In a casino, often card tables combine multiple decks of cards (so that counting cards is harder). For example, at a blackjack table, they might use 6 decks of cards. Assume you are sorting k identical decks of cards, where each deck has n cards. (There are kn cards in total.) Since the decks are identical, when the sorting is done, all the k copies of each card will be together.

Assume you are running Paranoid QuickSort on this big deck of kn cards. The pivot is being chosen uniformly at random, and you are using an efficient 3-way partitioning scheme. What is the expected worst-case running time of the sort? *(Hint: think about the base case where the recursion terminates.)*

1. $\Theta(n)$
2. $\Theta(n \log k)$
3. $\Theta(n \log n)$
4. $\Theta(nk)$
5. $\Theta(nk \log k)$
6. $\Theta(nk \log n)$
7. $\Theta(nk \log nk)$
8. None of the above.

[3 marks]

D. Assume that comparing two strings of length k_1 and k_2 takes $\min(k_1, k_2)$ time. The worst-case running time for inserting a string of length L into an AVL tree of size n where all the keys in the tree have length L is:

1. $O(1)$
2. $O(L)$
3. $O(\log n)$
4. $O(L \log n)$
5. $O(\log n + L)$
6. $O(nL)$

[2 marks]

E. What is the worst-case number of rotations during an insert into an AVL tree?

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n^2)$

[2 marks]

F. Given a sorted array of keys, how fast can you build an AVL tree, using the most efficient technique you know of?

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n \log^2 n)$
6. $O(n^2)$
7. $O(n^3)$

[3 marks]

Question 4: Invariants and Algorithms [17 marks]

A. Some Sort of Algorithm. Consider the following (pseudo)code (where `swap(a, i, j)` swaps the items at index i and j in array a):

```
void SomeSort(int[] array)
    int size = array.length;
    for (int i=1; i<size; i++)
        for (int j=0; j<i; j++)
            if (array[i] >= array[j])
                swap(array, i, j);
```

Which of the following is a good loop invariant for the outer loop (i.e., a property that is always true immediately after each iteration of the outer loop). Choose one.

1. For all k such that $k < i$: $A[k] \leq A[k+1]$.
2. For all k such that $k < i$: $A[k] \geq A[k+1]$.
3. The subarray $A[0..i]$ contains the $i+1$ smallest elements in the array.
4. The subarray $A[0..i-1]$ contains the i smallest elements in the array.
5. The subarray $A[0..i]$ contains the $i+1$ largest elements in the array.
6. The subarray $A[0..i-1]$ contains the i largest elements in the array.
7. None of the above.

[3 marks]

B. Is the SomeSort algorithm stable?

1. Stable
2. Not stable

[2 marks]

C. Which of the following are invariants for an AVL tree (evaluated at the end of every operation, i.e., after all rotations and updates are complete)? Assume height is defined as in class, where a leaf has height 0.

I. If node u and v are siblings, then $|height(u) - height(v)| < 2$.

II. If node u is the parent of node v, then $|height(u) - height(v)| < 2$.

III. If node u is the parent of node v, then $|height(u) - height(v)| > 0$.

IV. If node u has height h, then the number of nodes in the subtree rooted at u is at most 2^h.

[3 marks]

D. Which of the following are invariants for an (a,b)-tree (i.e., properties that are true at the end of every operation)? Assume height is as defined as in an AVL tree, where a leaf has height 0.

I. If node u and v are siblings, then $|height(u) - height(v)| < 1$.

II. If node u and v are siblings, then $|height(u) - height(v)| < 2$.

III. If non-root node u has height h, then the subtree rooted at u contains at least a^h nodes.

IV. If non-root node u has height h, then the subtree rooted at u contains at least b^h nodes.

[3 marks]

E. Consider a new type of tree, which we will call a mutated-(a,b)-tree. It satisfies the following properties: (i) every node has at most degree b, and (ii) every leaf is the same distance from the root. Based only on these properties, can you conclude that every mutated-(a,b)-tree balanced?

1. Always balanced.
2. Not always balanced.

[2 marks]

F. Consider the following (pseudocode) implementation (which behaves like Java) of a "special search":

```
1.    int specialSearch(int[] A, int key, int low, int high)
2.        if ((A == null) or (A.length == 0)) return NOT_FOUND;
3.        if (low >= high)
4.            if (A[low] == key) return low;
5.            else return -1;
6.        mid = low + (high-low)/2;
7.        first = specialSearch(A, key, low, mid-1);
8.        if (first != -1) return first;
9.        second = specialSearch(A, key, mid, high);
10.       return second;
```

Which of the following statements best characterizes the `specialSearch` algorithm?

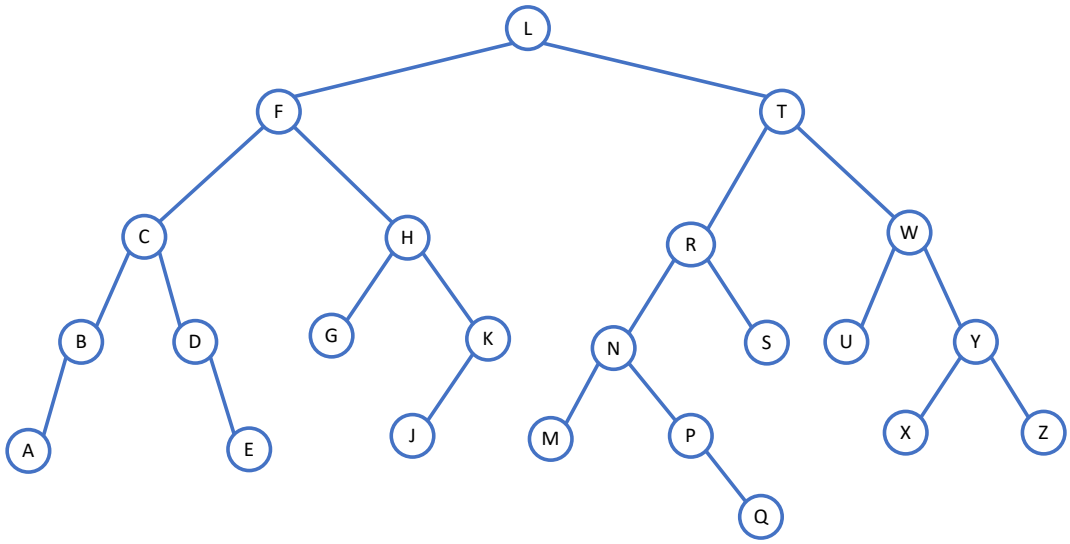
1. The algorithm works correctly: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will return the index of k if k is in A and return -1 otherwise.
2. The algorithm sometimes fails to find a key: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes return -1 even when k is in A.

3. The algorithm sometimes returns the wrong index: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes return an index j where $A[j] \neq k$.
4. The algorithm does not terminate: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes never return.
5. None of the above options is a good description of the situation.

[4 marks]

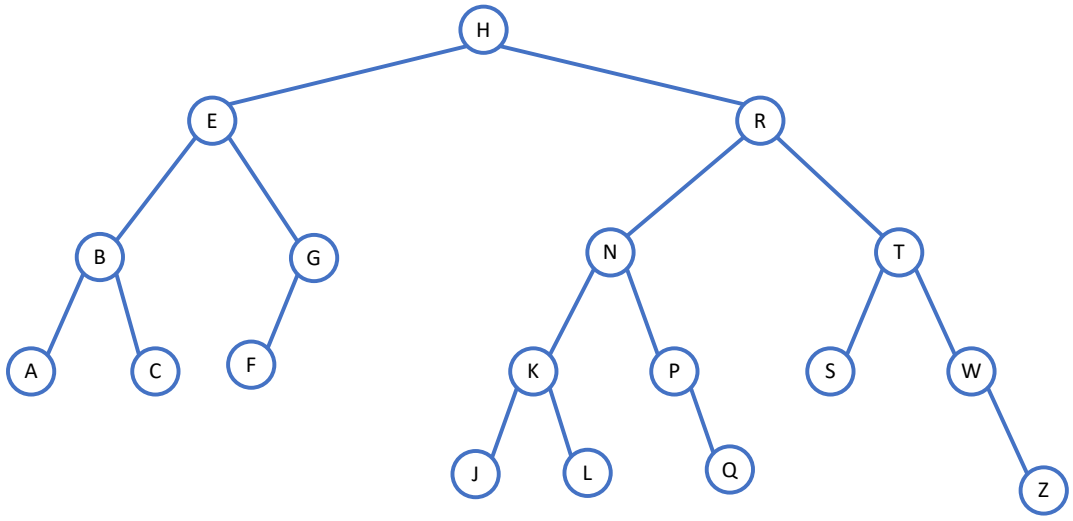
Question 5: Be the TreePuter [14 marks]

A. The tree below is an AVL tree, where some node has just been inserted, but no rebalance operations have yet been executed. The keys are letters of the alphabet, where A is smallest and Z is largest. Which node(s) are out of balance, if any? If more than one node is out-of-balance, choose the highest node in the tree that is out-of-balance. (Only choose one answer.)



[4 marks]

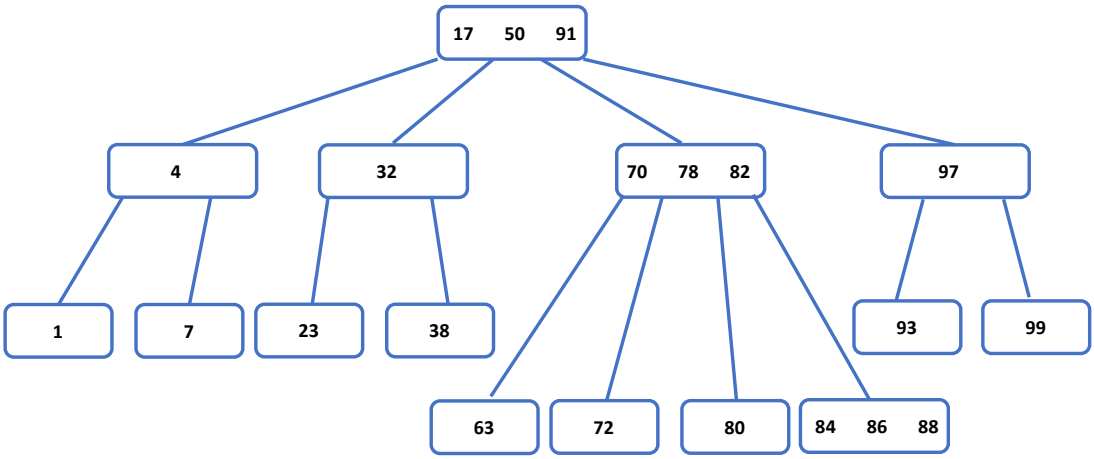
B. The tree below is an AVL tree. Assume that we insert *M* into the tree. Which rotation(s) occur? (Identify a rotation with the root of the subtree rotated. For example, a `right-rotate(E)` would move *E* down and *B* up.)



1. right-rotate(R)
2. left-rotate(H)
3. right-rotate(N)
4. left-rotate(T)
5. right-rotate(R), left-rotate(H)
6. left-rotate(H), right-rotate(R)
7. left-rotate(K), right-rotate(N)
8. right-rotate(N), left-rotate(K)
9. right-rotate(K), left-rotate(H)
10. left-rotate(T), right-rotate(R)
11. No rotations occur.
12. None of the above.

[4 marks]

C. The tree below is a $(2,4)$ tree. Assume that a key with value 90 is inserted into this tree. How many nodes split during the insertion?



1. 0
2. 1
3. 2
4. 3
5. 4
6. 5
7. 6

[2 marks]

D. After the insertion of value 90 is completed in the $(2,4)$-tree, what are the keys stored in the root? (Assume that when a node splits, if there are an odd number of remaining keys, the smaller number of keys go to the left node and the larger number of keys go to the right node.)

1. (17, 50, 91)
2. (50)
3. (17, 50, 78, 91)
4. (50, 78, 82)
5. (86)
6. (90)
7. (17, 50, 90, 91)
8. (50, 78)
9. None of the above.

[2 marks]

E. After the insertion of value 90 is completed in the $(2,4)$-tree, what is the height of the tree?

1. 0
2. 1
3. 2
4. 3
5. 4
6. 5
7. 6

[2 marks]

Question 6: The HeaptRee [14 marks]

There comes a time in every coder's life when they must design a new type of tree. For Rowan Timber, that day is today! Rowan's idea is that trees get hot in the sun, and so we should store a temperature in every node.

Each node in Rowan's tree has two parts:

- key: The tree is sorted by the key. The tree is a valid binary search tree with respect to the keys.
- temp: The temperature indicates how hot the node is. A tree should be hottest closest to the root, so Rowan's tree will satisfy the following temperature property: if node v has temperature $v.temp$, and if u is a child of v, then $u.temp < v.temp$.

A. Rowan has been studying balanced trees, and hopes that these two properties together will ensure that the tree is balanced. As Rowan's algorithm consultant, what advice would you give? For a worst-case choice of keys and temperatures, what is the worst-case height of a tree containing n nodes that satisfies the binary search tree property (with respect to the keys) and the temperature property?

1. $O(1)$
2. $O(\log n)$
3. $O(\log^2 n)$
4. $O(\sqrt{n})$
5. $O(n)$
6. $O(n \log n)$
7. $O(n^2)$

[3 marks]

B. In order to show that the tree has good balance properties, Rowan has been studying the various balance properties we have seen in CS2040S, e.g., when a tree is balanced, height-balanced, and weight-balanced. Rowan is especially interested in a version of weight-balance similar to that which you saw on Problem Set 5 for Scapegoat trees:

1. The weight of a node u is equal to the number of nodes in the subtree rooted at u. We denote this $u.weight$.
2. A node u is $3/4$-weight-balanced if, for every child v of u: $v.weight \leq \frac{3}{4} u.weight$
3. Tree T is $3/4$-weight balanced if every node in T is $3/4$ weight-balanced.

Which of the following statements are true? (The definitions of balanced and height-balanced are as discussed in class.)

I. Every $(3/4)$-weight-balanced tree is balanced.

II. Every $(3/4)$-weight-balanced tree is height-balanced.

III. Every height-balanced tree is $(3/4)$-weight-balanced.

[3 marks]

C. Assume that every node in tree T is $(3/4)$-weight-balanced. Which of the following is the best approximate upper bound on the height of tree T?

1. $\log_2(n)+1$
2. $2\log_2(n)+1$
3. $\log_{4/3}(n)+1$
4. $\log_{3/4}(n)+1$
5. $\log_{3/2}(n)+1$
6. $(4/3)\log_2(n)+1$
7. $(4/3)\log_{4/3}(n)+1$
8. $(4/3)\log_{3/4}(n)+1$
9. $(3/4)\log_{3/4}(n)+1$
10. None of the above is a reasonable upper bound on the height.

[3 marks]

D. Rowan next ponders what would happen if the temperatures were chosen uniformly at random. For example, the temperatures might be chosen as real numbers in the range of $(0, 100)$ with sufficient digits of decimal precision so that there are no duplicates. What is the (approximate) probability that the root node is $(3/4)$-weight-balanced? *(Hint: Think about choosing the pivot in QuickSort.)*

1. 0
2. $1/4$
3. $1/3$
4. $1/2$
5. $2/3$
6. $3/4$
7. 1
8. None of the above.

[3 marks]

E. Rowan does not know exactly what to do with that information, and so decides to defer further analysis of the new data structure until the CS2040S midterm is over. Rowan then considers how to insert a node into the new type of tree. Consider the following proposed algorithm for inserting a new node with a given key and temperature:

1. Insert the node as a leaf using a traditional binary tree insert based on the key (and ignoring the temperature).
2. Start at the leaf where the node was inserted, and walk up the tree checking if $v.temp > v.parent.temp$. If that condition holds (and hence the temperature property is violated), then fix it as follows:
   - If v is a left child of $v.parent$, do a right rotation of $v.parent$.
   - If v is a right child, do a left rotation of $v.parent$.
   - After the first such rotation, return.

Does this algorithm work correctly?

1. Yes, after the rotation, the temperature ordering property and binary search tree property both hold.
2. No, after the rotation, there (still) may be temperature violations.
3. No, after the rotation, there may be binary-search-tree property violations.

[2 marks]

Question 7: The IntervalRay of Doom [11 marks]

Ina Tervalfan has never taken CS2040S, but needs a data structure to store a collection of intervals of the form (a,b) where a and b are integers. She decides to design a new data structure called an IntervalRay, i.e., an array of intervals. The IntervalRay consists of an array I with one entry for each interval. Each entry $I[j]$ contains four values: (i) left: the left endpoint of the interval, (ii) right: the right endpoint of the interval, (iii) special: to be discussed later, and (iv) count: the number of intervals whose right interval is to the left of interval $I[j].left$. We will assume for this problem that each interval has unique endpoints.

Ina has several design decisions to make for this data structures. She has already determined that the IntervalRay is sorted by the right endpoint, i.e., $I[j].right < I[j+1].right$. Your job is to help her with the remaining design decisions so as to produce a data structure that, once completed, can support the following query operation: `findInterval(x)` returns an interval that contains point x, if such an interval exists in the IntervalRay, and returns *null* otherwise. Notice that the data structure is not dynamic, i.e., it does not support inserting or deleting intervals.

Note that your job is to choose a set of answers that result in a functioning, correct, efficient data structure. Some answers may be correct for a different design, but will not be compatible with the other choices you have to make. Look at all the parts of this question before choosing your answer.

A. The first decision is what to store in the special variable in each cell. Ina is considering the following options to store in $I[j].special$:

1. The maximum right endpoint of any interval stored in $I[k]$ where $k \leq j$.
2. The maximum right endpoint of any interval stored in $I[k]$ where $k \geq j$.
3. The minimum left endpoint of any interval stored in $I[k]$ where $k \leq j$.
4. The minimum left endpoint of any interval stored in $I[k]$ where $k \geq j$.

Which of these will be most useful for Ina's algorithm?

[3 marks]

B. Ina is trying to understand her new data structure better, and so you suggest to her to think about the invariants that it might satisfy. Consider the following properties of the IntervalRay containing n intervals:

I. The special variable is monotonically non-decreasing, i.e., $I[k].special \leq I[k+1].special$ for all $k < n-1$.

II. The special variable is monotonically non-increasing, i.e., $I[k].special \geq I[k+1].special$ for all $k < n-1$.

III. The count variable is monotonically non-decreasing, i.e., $I[k].count \leq I[k+1].count$ for all $k < n-1$.

IV. The count variable is monotonically non-increasing, i.e., $I[k].count \geq I[k+1].count$ for all $k < n-1$.

Which of these properties are always true of the IntervalRay?

[3 marks]

C. After studying the monotonicity properties, Ina thanks that she can use binary search to implement the `findInterval`(x) query. She suggests the following possibilities. Which do you think will work?

1. Binary search I, comparing $I[j].left$ to x.
2. Binary search I, comparing $I[j].right$ to x.
3. Binary search I, comparing $I[j].special$ to x.
4. Binary search I, comparing $I[j].count$ to x.
5. None of these strategies work properly.

[3 marks]

D. Ina is not sure about the binary search strategy. So she instead proposes a simpler approach:

1. Sort the IntervalRay by left endpoint (i.e., $I[j].left$) using MergeSort. Then scan the array from left to right removing all intervals where $x < I[j].left$.
2. Sort the IntervalRay by right endpoint (i.e., $I[j].right$) using MergeSort. Then scan the array from left to right removing all intervals where $x > I[j].right$.
3. If there are any intervals left in the array, return that one.

What is your evaluation of this approach?

1. The approach works, and is a reasonable alternative, assuming that the binary search does not work.
2. The approach works, but even assuming binary search does not work, there are better solutions.
3. The approach does not work.

[2 marks]

CS2040S — Algorithms and Data Structures
School of Computing
National University of Singapore

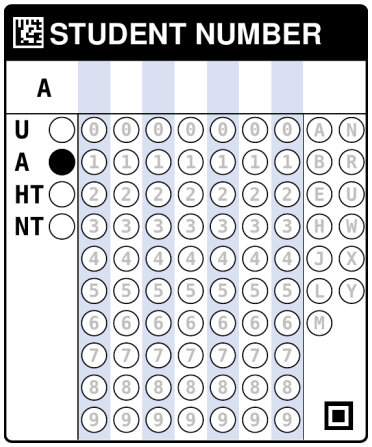
Midterm Assessment — Answer Sheet

2022/2023 Semester 2

Time allowed: 2 hours

Instructions (please read carefully):

1. Write down your **student number** on the right and using ink or pencil, shade the corresponding circle in the grid for each digit or letter. DO NOT WRITE YOUR NAME!
2. This answer booklet comprises **TEN (10) pages**, including this cover page.
3. All questions must be answered in the space provided; no extra sheets will be accepted as answers. You may use the extra page behind this cover page if you need more space for your answers.
4. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.
5. An excerpt of the question may be provided to aid you in answering in the correct box. It is not the exact question. You should still refer to the original question in the question booklet.
6. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).
7. **Marks may be deducted** for unreadable answers.



For Examiner's Use Only

Question	Marks
Q1	/ 12
Q2	/ 18
Q3	/ 14
Q4	/ 17
Q5	/ 14
Q6	/ 14
Q7	/ 11
Q8	/ 0
Total	/100

This page is intentionally left blank.
Use it ONLY if you need extra space for your answers, and indicate the **question number clearly** as well as in the original answer box. **Do NOT** use it for your rough work.

Question 1A What sort was used on Column A? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

Question 1B What sort was used on Column B? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

Question 1C What sort was used on Column C? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

Question 1D What sort was used on Column D? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

Question 1E What sort was used on Column E? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

Question 1F What sort was used on Column F? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

Question 2A Tightest bound from the available options: [2 marks]

◯ $O(\log n)$ ◯ $O(n)$ ◯ $O(n \log n)$ ◯ $O(n^2)$ ◯ $O(n^3)$ ◯ $O(2^n)$

Question 2B Tightest bound from the available options: [2 marks]

◯ $O(n)$ ◯ $O(n \log n)$ ◯ $O(n \log^2 n)$ ◯ $O(n^2)$ ◯ $O(n^3)$ ◯ $O(2^n)$

Question 2C True or false: [2 marks]

◯ True ◯ False

Question 2D Tightest bound from the available options: [3 marks]

◯ $O(1)$ ◯ $O(n)$ ◯ $O(n \log^2 n)$ ◯ $O(n^3)$
◯ $O(\log n)$ ◯ $O(n \log n)$ ◯ $O(n^2)$ ◯ $O(2^n)$

Question 2E Tightest bound from the available options: [3 marks]

◯ $O(1)$ ◯ $O(n)$ ◯ $O(n \log^2 n)$ ◯ $O(n^3)$
◯ $O(\log n)$ ◯ $O(n \log n)$ ◯ $O(n^2)$ ◯ $O(2^n)$

Question 2F Asymptotic running time. [3 marks]

◯ $O(1)$ ◯ $O(n)$ ◯ $O(n^2)$ ◯ $O(2^n)$
◯ $O(\log n)$ ◯ $O(n \log n)$ ◯ $O(n^3)$

Question 2G Which recurrence best describes this function? [3 marks]

◯ $T(n) \leq T(n/2) + O(n)$. ◯ $T(n) \leq T(n/4) + O(n^2)$.
◯ $T(n) \leq T(n/2) + O(n^2)$. ◯ $T(n) \leq 4T(n/4) + O(n)$.
◯ $T(n) \leq 2T(n/2) + O(n^2)$. ◯ $T(n) \leq 2T(n/4) + O(n^2)$.
◯ $T(n) \leq T(n/4) + O(n)$. ◯ None of the above.

Question 3A Worst-case for InsertionSort on array containing only 2 elements: [2 marks]

◯ $O(1)$ ◯ $O(n)$ ◯ $O(n \log^2 n)$ ◯ $O(n^3)$
◯ $O(\log n)$ ◯ $O(n \log n)$ ◯ $O(n^2)$ ◯ $O(2^n)$

Question 3B MergeSort on partially sorted array: [2 marks]

- ◯ $O(1)$
- ◯ $O(n)$
- ◯ $O(n\log^2 n)$
- ◯ $O(n^3)$
- ◯ $O(\log n)$
- ◯ $O(n\log n)$
- ◯ $O(n^2)$
- ◯ $O(2^n)$

Question 3C Paranoid QuickSort on a deck of kn cards: [3 marks]

- ◯ $\Theta(n)$
- ◯ $\Theta(nk)$
- ◯ $\Theta(nk\log(nk))$
- ◯ $\Theta(n\log(k))$
- ◯ $\Theta(nk\log(k))$
- ◯ $\Theta(n\log(n))$
- ◯ $\Theta(nk\log(n))$
- ◯ None of the above.

Question 3D Worst-case running time for inserting a string of length? [2 marks]

- ◯ $O(1)$
- ◯ $O(\log n)$
- ◯ $O(\log n + L)$
- ◯ $O(L)$
- ◯ $O(L\log n)$
- ◯ $O(nL)$

Question 3E Rotations on insertion. [2 marks]

- ◯ $O(1)$
- ◯ $O(n)$
- ◯ $O(n^2)$
- ◯ $O(\log n)$
- ◯ $O(n\log n)$

Question 3F Building an AVL tree. [3 marks]

- ◯ $O(1)$
- ◯ $O(n)$
- ◯ $O(n\log^2 n)$
- ◯ $O(n^3)$
- ◯ $O(\log n)$
- ◯ $O(n\log n)$
- ◯ $O(n^2)$

Question 4A Good loop invariant? [3 marks]

- ◯ For all k such that $k < i$: $A[k] \leq A[k+1]$.
- ◯ For all k such that $k < i$: $A[k] \geq A[k+1]$.
- ◯ The subarray $A[0..i]$ contains the $i+1$ smallest elements in the array.
- ◯ The subarray $A[0..i-1]$ contains the i smallest elements in the array.
- ◯ The subarray $A[0..i]$ contains the $i+1$ largest elements in the array.
- ◯ The subarray $A[0..i-1]$ contains the i largest elements in the array.
- ◯ None of the above.

Question 4B Stable or not? [2 marks]

- ◯ Stable
- ◯ Not stable

Question 4C Invariants for an AVL tree. [3 marks]

- ◯ Only I.
- ◯ Only II.
- ◯ Only III.
- ◯ Only IV.
- ◯ I and II.
- ◯ I and III.
- ◯ I and IV.
- ◯ II and III.
- ◯ II and IV.
- ◯ III and IV.
- ◯ I, II, and III.
- ◯ I, II, and IV.
- ◯ II, III, and IV.
- ◯ I, II, III, and IV.
- ◯ None of the above is accurate.

Question 4D Invariants for an (a,b)-tree. [3 marks]

- ◯ Only I.
- ◯ Only II.
- ◯ Only III.
- ◯ Only IV.
- ◯ I and II.
- ◯ I and III.
- ◯ I and IV.
- ◯ II and III.
- ◯ II and IV.
- ◯ III and IV.
- ◯ I, II, and III.
- ◯ I, II, and IV.
- ◯ II, III, and IV.
- ◯ I, II, III, and IV.
- ◯ None of the above is accurate.

Question 4E Balanced or not? [2 marks]

- ◯ Balanced
- ◯ Not balanced

Question 4F Special search? [4 marks]

◯ The algorithm works correctly: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will return the index of k if k is in A and return -1 otherwise.

◯ The algorithm sometimes fails to find a key: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes return -1 even when k is in A.

◯ The algorithm sometimes returns the wrong index: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes return an index j where $A[j] \neq k$.

◯ The algorithm does not terminate: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes never return.

◯ None of the above options is a good description of the situation.

Question 5A Highest out-of-balance node? [4 marks]

◯ F ◯ H ◯ L ◯ M ◯ N

◯ P ◯ Q ◯ R ◯ S ◯ T

◯ U ◯ W ◯ X ◯ Y ◯ Z

◯ None of these listed.

Question 5B How to balance? [4 marks]

◯ right-rotate(R)
◯ left-rotate(H)
◯ right-rotate(N)
◯ left-rotate(T)
◯ right-rotate(R), left-rotate(H)
◯ left-rotate(H), right-rotate(R)
◯ left-rotate(K), right-rotate(N)
◯ right-rotate(N), left-rotate(K)
◯ right-rotate(K), left-rotate(H)
◯ left-rotate(T), right-rotate(R)
◯ No rotations occur.
◯ None of the above.

Question 5C How many split? [2 marks]

◯ 0 ◯ 1 ◯ 2

◯ 3 ◯ 4 ◯ 5

◯ 6

Question 5D What keys are in the root node? [2 marks]

- ◯ (17, 50, 91)
- ◯ (50)
- ◯ (17, 50, 78, 91)
- ◯ (50, 78, 82)
- ◯ (86)
- ◯ (90)
- ◯ (17, 50, 90, 91)
- ◯ (50, 78)
- ◯ None of the above.

Question 5E What is the height of the tree? [2 marks]

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6

Question 6A Worst-case tree height: [3 marks]

- ◯ $O(1)$
- ◯ $O(\log n)$
- ◯ $O(\log^2 n)$
- ◯ $O(\sqrt{n})$
- ◯ $O(n)$
- ◯ $O(n \log n)$
- ◯ $O(n^2)$

Question 6B Which statements are true? [3 marks]

- ◯ I only.
- ◯ II only.
- ◯ III only.
- ◯ I and II.
- ◯ I and III.
- ◯ II and III.
- ◯ I and II and III.
- ◯ None of the three statements are true.

Question 6C Upper bound on (3/4)-weight-balanced tree? [3 marks]

- ◯ $\log_2(n)+1$
- ◯ $2\log_2(n)+1$
- ◯ $\log_{4/3}(n)+1$
- ◯ $\log_{3/4}(n)+1$
- ◯ $\log_{3/2}(n)+1$
- ◯ $(4/3)\log_2(n)+1$
- ◯ $(4/3)\log_{4/3}(n)+1$
- ◯ $(4/3)\log_{3/4}(n)+1$
- ◯ $(3/4)\log_{3/4}(n)+1$
- ◯ None of the above is a reasonable upper bound on the height.

Question 6D Random temperatures? [3 marks]

- ◯ 0
- ◯ 1/4
- ◯ 1/3
- ◯ 1/2
- ◯ 2/3
- ◯ 3/4
- ◯ 1
- ◯ None of the above.

Question 6E Does it work? [2 marks]

- ◯ Yes, after the rotation, the temperature ordering property and binary search tree property both hold.
- ◯ No, after the rotation, there (still) may be temperature violations.
- ◯ No, after the rotation, there may be binary-search-tree property violations.

Question 7A What is the best use of the special variable? [3 marks]

- ◯ The maximum right endpoint of any interval stored in $I[k]$ where $k \leq j$.
- ◯ The maximum right endpoint of any interval stored in $I[k]$ where $k \geq j$.
- ◯ The minimum left endpoint of any interval stored in $I[k]$ where $k \leq j$.
- ◯ The minimum left endpoint of any interval stored in $I[k]$ where $k \geq j$.
- ◯ None of these are useful.

Question 7B Which properties are true of the IntervalRay? [3 marks]

- ◯ Only I.
- ◯ Only II.
- ◯ Only III.
- ◯ Only IV.
- ◯ I and II.
- ◯ I and III.
- ◯ II and III.
- ◯ II and IV.
- ◯ I, II and III.
- ◯ II, III and IV.
- ◯ None of the above indicates properties that are always true.

Question 7C Which do you think will work? [3 marks]

- ◯ Binary search I, comparing $I[j].left$ to x.
- ◯ Binary search I, comparing $I[j].right$ to x.
- ◯ Binary search I, comparing $I[j].special$ to x.
- ◯ Binary search I, comparing $I[j].count$ to x.
- ◯ None of these strategies work properly.

Question 7D What is your evaluation of this approach? [2 marks]

◯ The approach works, and is a reasonable alternative, assuming that the binary search does not work.

◯ The approach works, but even assuming binary search does not work, there are better solutions.

◯ The approach does not work.

Question 8 [0 marks]

The Dark Room

— E N D O F A N S W E R S H E E T —

Question 1A What sort was used on Column A? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

(InsertionSort)

Question 1B What sort was used on Column B? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

(QuickSort)

Question 1C What sort was used on Column C? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

(SelectionSort)

Question 1D What sort was used on Column D? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

(None of the above: HeapSort)

Question 1E What sort was used on Column E? [2 marks]

- ◯ BubbleSort
- ◯ SelectionSort
- ◯ InsertionSort
- ◯ MergeSort
- ◯ QuickSort (first element pivot)
- ◯ None of the above.

(MergeSort)

Question 1F What sort was used on Column F? [2 marks]

◯ BubbleSort	◯ MergeSort
◯ SelectionSort	◯ QuickSort (first element pivot)
◯ InsertionSort	◯ None of the above.

(BubbleSort)

Question 2A Tightest bound from the available options: [2 marks]

◯ $O(\log n)$ ◯ $O(n)$ ◯ $O(n \log n)$ ◯ $O(n^2)$ ◯ $O(n^3)$ ◯ $O(2^n)$ $O(n^2)$

Question 2B Tightest bound from the available options: [2 marks]

◯ $O(n)$ ◯ $O(n \log n)$ ◯ $O(n \log^2 n)$ ◯ $O(n^2)$ ◯ $O(n^3)$ ◯ $O(2^n)$ $O(n^2)$

Question 2C True or false: [2 marks]

◯ True ◯ False False: $2^{\log^2 n} = n^{\log n} \neq O(n^{17})$

Question 2D Tightest bound from the available options: [3 marks]

◯ $O(1)$	◯ $O(n)$	◯ $O(n \log^2 n)$	◯ $O(n^3)$
◯ $O(\log n)$	◯ $O(n \log n)$	◯ $O(n^2)$	◯ $O(2^n)$

$O(n)$

Question 2E Tightest bound from the available options: [3 marks]

◯ $O(1)$	◯ $O(n)$	◯ $O(n \log^2 n)$	◯ $O(n^3)$
◯ $O(\log n)$	◯ $O(n \log n)$	◯ $O(n^2)$	◯ $O(2^n)$

$O(n \log n)$

Question 2F Asymptotic running time. [3 marks]

◯ $O(1)$	◯ $O(n)$	◯ $O(n^2)$	◯ $O(2^n)$
◯ $O(\log n)$	◯ $O(n \log n)$	◯ $O(n^3)$	

$O(n \log n)$

Question 2G Which recurrence best describes this function? [3 marks]

- ◯ $T(n) \leq T(n/2) + O(n)$.
- ◯ $T(n) \leq T(n/2) + O(n^2)$.
- ◯ $T(n) \leq 2T(n/2) + O(n^2)$.
- ◯ $T(n) \leq T(n/4) + O(n)$.
- ◯ $T(n) \leq T(n/4) + O(n^2)$.
- ◯ $T(n) \leq 4T(n/4) + O(n)$.
- ◯ $T(n) \leq 2T(n/4) + O(n^2)$.
- ◯ None of the above.

$T(n) \leq 2T(n/4) + O(n^2)$

Question 3A Worst-case for InsertionSort on array containing only 2 elements: [2 marks]

- ◯ $O(1)$
- ◯ $O(\log n)$
- ◯ $O(n)$
- ◯ $O(n \log n)$
- ◯ $O(n \log^2 n)$
- ◯ $O(n^2)$
- ◯ $O(n^3)$
- ◯ $O(2^n)$

$O(n^2)$

Question 3B MergeSort on partially sorted array: [2 marks]

- ◯ $O(1)$
- ◯ $O(\log n)$
- ◯ $O(n)$
- ◯ $O(n \log n)$
- ◯ $O(n \log^2 n)$
- ◯ $O(n^2)$
- ◯ $O(n^3)$
- ◯ $O(2^n)$

$O(n \log n)$

Question 3C Paranoid QuickSort on a deck of kn cards: [3 marks]

- ◯ $\Theta(n)$
- ◯ $\Theta(n \log(k))$
- ◯ $\Theta(n \log(n))$
- ◯ $\Theta(nk)$
- ◯ $\Theta(nk \log(k))$
- ◯ $\Theta(nk \log(n))$
- ◯ $\Theta(nk \log(nk))$
- ◯ None of the above.

$O(nk \log n)$

Question 3D Worst-case running time for inserting a string of length? [2 marks]

- ◯ $O(1)$
- ◯ $O(L)$
- ◯ $O(\log n)$
- ◯ $O(L \log n)$
- ◯ $O(\log n + L)$
- ◯ $O(nL)$

$O(L \log n)$

Question 3E Rotations on insertion. [2 marks]

◯ $O(1)$ ◯ $O(n)$ ◯ $O(n^2)$
◯ $O(\log n)$ ◯ $O(n \log n)$

$O(1)$

Question 3F Building an AVL tree. [3 marks]

◯ $O(1)$ ◯ $O(n)$ ◯ $O(n \log^2 n)$ ◯ $O(n^3)$
◯ $O(\log n)$ ◯ $O(n \log n)$ ◯ $O(n^2)$

$O(n)$

Question 4A Good loop invariant? [3 marks]

◯ For all k such that $k < i$: $A[k] \leq A[k+1]$.
◯ For all k such that $k < i$: $A[k] \geq A[k+1]$.
◯ The subarray $A[0..i]$ contains the $i+1$ smallest elements in the array.
◯ The subarray $A[0..i-1]$ contains the i smallest elements in the array.
◯ The subarray $A[0..i]$ contains the $i+1$ largest elements in the array.
◯ The subarray $A[0..i-1]$ contains the i largest elements in the array.
◯ None of the above.

$A[k] \geq A[k+1]$

Question 4B Stable or not? [2 marks]

◯ Stable ◯ Not stable

Not stable.

Question 4C Invariants for an AVL tree. [3 marks]

- ◯ Only I.
- ◯ Only II.
- ◯ Only III.
- ◯ Only IV.
- ◯ I and II.
- ◯ I and III.
- ◯ I and IV.
- ◯ II and III.
- ◯ II and IV.
- ◯ III and IV.
- ◯ I, II, and III.
- ◯ I, II, and IV.
- ◯ II, III, and IV.
- ◯ I, II, III, and IV.
- ◯ None of the above is accurate.

I and III are correct.

Question 4D Invariants for an (a,b)-tree. [3 marks]

- ◯ Only I.
- ◯ Only II.
- ◯ Only III.
- ◯ Only IV.
- ◯ I and II.
- ◯ I and III.
- ◯ I and IV.
- ◯ II and III.
- ◯ II and IV.
- ◯ III and IV.
- ◯ I, II, and III.
- ◯ I, II, and IV.
- ◯ II, III, and IV.
- ◯ I, II, III, and IV.
- ◯ None of the above is accurate.

I and II and III are correct.

Question 4E Balanced or not? [2 marks]

- ◯ Balanced
- ◯ Not balanced

Not balanced.

Question 4F Special search? [4 marks]

◯ The algorithm works correctly: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will return the index of k if k is in A and return -1 otherwise.

◯ The algorithm sometimes fails to find a key: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes return -1 even when k is in A.

◯ The algorithm sometimes returns the wrong index: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes return an index j where $A[j] \neq k$.

◯ The algorithm does not terminate: when invoked on array A with $low = 0$ and $high = A.length - 1$, a search for a key k will sometimes never return.

◯ None of the above options is a good description of the situation.

Does not temrinate.

Question 5A Highest out-of-balance node? [4 marks]

◯ F ◯ H ◯ L ◯ M ◯ N

◯ P ◯ Q ◯ R ◯ S ◯ T

◯ U ◯ W ◯ X ◯ Y ◯ Z

◯ None of these listed.

R

Question 5B How to balance? [4 marks]

◯ right-rotate(R)
◯ left-rotate(H)
◯ right-rotate(N)
◯ left-rotate(T)
◯ right-rotate(R), left-rotate(H)
◯ left-rotate(H), right-rotate(R)
◯ left-rotate(K), right-rotate(N)
◯ right-rotate(N), left-rotate(K)
◯ right-rotate(K), left-rotate(H)
◯ left-rotate(T), right-rotate(R)
◯ No rotations occur.
◯ None of the above.

`right-rotate(R)`, `left-rotate(H)`

Question 5C How many split? [2 marks]

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6

3

Question 5D What keys are in the root node? [2 marks]

- ◯ (17, 50, 91)
- ◯ (50)
- ◯ (17, 50, 78, 91)
- ◯ (50, 78, 82)
- ◯ (86)
- ◯ (90)
- ◯ (17, 50, 90, 91)
- ◯ (50, 78)
- ◯ None of the above.

(50)

Question 5E What is the height of the tree? [2 marks]

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6

3

Question 6A Worst-case tree height: [3 marks]

- ◯ $O(1)$
- ◯ $O(\log n)$
- ◯ $O(\log^2 n)$
- ◯ $O(\sqrt{n})$
- ◯ $O(n)$
- ◯ $O(n \log n)$
- ◯ $O(n^2)$

$O(n)$

Question 6B Which statements are true? [3 marks]

- ◯ I only.
- ◯ II only.
- ◯ III only.
- ◯ I and II.
- ◯ I and III.
- ◯ II and III.
- ◯ I and II and III.
- ◯ None of the three statements are true.

I only is true.

Question 6C Upper bound on (3/4)-weight-balanced tree? [3 marks]

- ◯ $\log_2(n)+1$
- ◯ $2\log_2(n)+1$
- ◯ $\log_{4/3}(n)+1$
- ◯ $\log_{3/4}(n)+1$
- ◯ $\log_{3/2}(n)+1$
- ◯ $(4/3)\log_2(n)+1$
- ◯ $(4/3)\log_{4/3}(n)+1$
- ◯ $(4/3)\log_{3/4}(n)+1$
- ◯ $(3/4)\log_{3/4}(n)+1$
- ◯ None of the above is a reasonable upper bound on the height.

$\log_{4/3}(n)+1$

Question 6D Random temperatures? [3 marks]

- ◯ 0
- ◯ 1/4
- ◯ 1/3
- ◯ 1/2
- ◯ 2/3
- ◯ 3/4
- ◯ 1
- ◯ None of the above.

1/2

Question 6E Does it work? [2 marks]

- ◯ Yes, after the rotation, the temperature ordering property and binary search tree property both hold.
- ◯ No, after the rotation, there (still) may be temperature violations.
- ◯ No, after the rotation, there may be binary-search-tree property violations.

No, there may still be temperature violations

Question 7A What is the best use of the special variable? [3 marks]

- ◯ The maximum right endpoint of any interval stored in $I[k]$ where $k \leq j$.
- ◯ The maximum right endpoint of any interval stored in $I[k]$ where $k \geq j$.
- ◯ The minimum left endpoint of any interval stored in $I[k]$ where $k \leq j$.
- ◯ The minimum left endpoint of any interval stored in $I[k]$ where $k \geq j$.
- ◯ None of these are useful.

4 : The minimum left endpoint stored in cells $k \geq j$.

Question 7B Which properties are true of the IntervalRay? [3 marks]

- ◯ Only I.
- ◯ Only II.
- ◯ Only III.
- ◯ Only IV.
- ◯ I and II.
- ◯ I and III.
- ◯ II and III.
- ◯ II and IV.
- ◯ I, II and III.
- ◯ II, III and IV.
- ◯ None of the above indicates properties that are always true.

Only I: The special variable is monotonically non-decreasing.

Question 7C Which do you think will work? [3 marks]

- ◯ Binary search I, comparing $I[j].left$ to x.
- ◯ Binary search I, comparing $I[j].right$ to x.
- ◯ Binary search I, comparing $I[j].special$ to x.
- ◯ Binary search I, comparing $I[j].count$ to x.
- ◯ None of these strategies work properly.

3 : Binary search comparing $I[j].special$ to x.

Question 7D What is your evaluation of this approach? [2 marks]

- ◯ The approach works, and is a reasonable alternative, assuming that the binary search does not work.
- ◯ The approach works, but even assuming binary search does not work, there are better solutions.
- ◯ The approach does not work.

The approach works, but there are better solutions. Notably, a linear search is faster as it does not require sorting.