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

Midterm Assessment

11 Mar 2024 **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-THREE (23) pages**, and the **ANSWER SHEET** comprises of **FIFTEEN (15) 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. The questions are provided in the question booklet, and if there is any inconsistency in the question, please refer to the question booklet. If there is any inconsistency in the answers, refer to the answer sheet.
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: Tree Sort [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**
Nutmeg	Cedar	Maple	Cedar	Ash	Cedar	Ash	Ash
Cedar	Date	Juniper	Date	Beech	Date	Beech	Beech
Pine	Nutmeg	Linden	Elm	Cedar	Juniper	Cedar	Cedar
Date	Pine	Ilex	Juniper	Juniper	Nutmeg	Date	Date
Juniper	Elm	Date	Fir	Elm	Pine	Juniper	Elm
Elm	Juniper	Hazel	Gingko	Linden	Elm	Elm	Fir
Linden	Linden	Kauri	Linden	Fir	Linden	Linden	Gingko
Fir	Fir	Fir	Beech	Gingko	Fir	Fir	Hazel
Gingko	Gingko	Gingko	Ash	Maple	Gingko	Gingko	Ilex
Oak	Oak	Cedar	Hazel	Date	Oak	Oak	Juniper
Beech	Beech	Beech	Kauri	Ilex	Beech	Pine	Kauri
Ash	Ash	Ash	Maple	Hazel	Ash	Nutmeg	Linden
Hazel	Hazel	Elm	Ilex	Kauri	Hazel	Hazel	Maple
Kauri	Kauri	Nutmeg	Nutmeg	Nutmeg	Kauri	Kauri	Nutmeg
Maple	Maple	Oak	Oak	Oak	Maple	Maple	Oak
Ilex	Ilex	Pine	Pine	Pine	Ilex	Ilex	Pine
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 [22 marks]

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

$$T(n) = 20n^2 \log^2 n + 40n \log n + 17$$

.

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:

$$T(n) = \sum_{j=0}^{\log n} 2^j$$

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]

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

$$T(n) = \sum_{j=0}^{n} j$$

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]

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

1. True
2. False

[2 marks]

E. Which of the following is a proper definition of big-O notation? Given two (non-negative) functions f and g, we say that $f = O(g)$ if and only if:

1. $\exists\, c > 0, \forall n_0 > 0, \forall n \geq n_0 \ :\ f(n) \leq cg(n)$
2. $\exists\, c > 0, \forall n \geq 0 \ :\ f(n) \leq cg(n)$
3. $\exists\, n_0 > 0, \forall n \geq n_0, \exists c > 0 \ :\ f(n) \leq cg(n)$ [2 marks]
4. $\exists\, c > 0, \exists\, n_0 > 0, \forall n \geq n_0 \ :\ f(n) \leq cg(n)$
5. $\exists\, n_0 > 0, \forall n \geq n_0 \ :\ f(n) \leq g(n)$

F. Consider the following function that is defined in a piecewise manner:

$$f(n) = \begin{cases} n^3 & \text{if } n \text{ is even} \\ n & \text{if } n \text{ is odd} \end{cases}$$

Which of the following is true?

1. $f(n) = O(n^2)$
2. $f(n) = \Omega(n^2)$
3. Both are true: $f(n) = O(n^2)$ and $f(n) = \Omega(n^2)$. [2 marks]
4. None of the above are true.

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

$$T(n) = 4T(n/4) + 7$$

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]

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

$$T(n) = T(2n/3) + 2040$$

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]

I. What is the worst-case asymptotic running time of the following code, as a function of n, when you execute `Loopy(n, A)`? (Give the tightest bound possible of the available options.)

```
public void Loopy(int n, int[] A) {
    for (int i=1; i<n; i++) {
        int j=i;
        while ((j > 1) && (A[j] < A[j-1])) {
            doSomething(A, j, j-1); // doSomething runs in O(1) time
            j = j-1;
        }
    }
}
```

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

[2 marks]

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

```
public int rollerCoaster(int n, int m) {
    if (n < m) return m;

    int sum = 0;
    for (int i=1; i<n; i++) {
        sum = sum + m;
    }
    return sum + rollerCoaster(n/2, m) + rollerCoaster(n/2, m);
}
```

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

[2 marks]

K. Which recurrence best describes the runtime of the following `partyTime` algorithm? Give the recurrence as a function of n, the input to `partyTime`. Assume that the functions `doWork` and `doParty` run in $O(1)$ time. (Do not worry about the correctness of the algorithm.)

```
boolean partyTime(int n){
    // no time left
    if (n < 5) return false;

    // first, we study a little bit...
    studyTime(n/3);

    // now we deserve to take a break
    for (int i=0; i<n; i++) {
        doParty(); // doParty takes O(1) time
    }
    return true;
 }

void studyTime(int hours){
    // Ok, let's do some work
    for int (j = 0; j < 3; j++){
        doWork(); // doWork takes O(1) time
        // Shhh, don't tell!
        partyTime(hours);
    }
}
```

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

[2 marks]

Question 3: Faster and faster... [17 marks]

A. The "movement-cost" of an algorithm is the number of swaps/moves executed (ignoring, e.g., the comparisons). For this question, a movement is any operation that moves or copies an element in an array. What is the worst-case movement-cost of InsertionSort?

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]

B. The "movement-cost" of an algorithm is the number of swaps/moves executed (ignoring, e.g., the comparisons). For this question, a movement is any operation that moves or copies an element in an array. What is the worst-case movement-cost of SelectionSort?

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]

C. Imagine a deck of n distinct cards (i.e., no duplicates) that is initially sorted. We then separate the deck into groups of 10 cards, shuffle each group of 10 cards (i.e., an arbitrary ordering of the cards), and then reassemble the groups in the same order. (That is, the ordering of cards only changed within each group, not between groups.)

For example, if the sorted deck contains the numbers from 1 to 20, then the modified deck (consisting of two randomly permuted groups of size 10) might look like:

$$7, 10, 3, 8, 4, 2, 6, 9, 1, 5, 13, 20, 18, 12, 17, 14, 11, 16, 15, 19$$

What is the worst-case expected running time of Paranoid QuickSort (with random pivot selection and in-place partitioning) on this deck?

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]

D. Imagine a deck of n distinct cards (i.e., no duplicates) that is initially sorted. We then separate the deck into groups of 10 cards, shuffle each group of 10 cards (i.e., an arbitrary

ordering of the cards), and then reassemble the groups in the same order. (That is, the ordering of cards only changed within each group, not between groups.)

For example, if the sorted deck contains the numbers from 1 to 20, then the modified deck (consisting of two randomly permuted groups of size 10) might look like:

$$7, 10, 3, 8, 4, 2, 6, 9, 1, 5, 13, 20, 18, 12, 17, 14, 11, 16, 15, 19$$

What is the worst-case running time of InsertionSort on this deck?

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. What is the worst-case number of rotations during an insertion or a deletion in 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. When inserting a key into a $(2,4)$-tree with n nodes, what is the worst-case number of nodes that may need to be split? (Assume nodes are only split when they need to be, not proactively.)

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

[3 marks]

Question 4: Invariants and Algorithms [20 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=0; i<size; i++) // outer loop
        for (int j=0; j<size-1; j++)
            if (array[j] < array[j+1])
                swap(array, j+1, j);
        for (int j=size-1; j>0; j--)
            if (array[j] > array[j-1])
                swap(array, j-1, 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).

I. The subarray $A[0..i]$ contains the $i+1$ smallest elements in the array.

II. The subarray $A[0..i]$ contains the $i+1$ largest elements in the array.

III. The subarray $A[n-i-1, n-1]$ contains the $i+1$ smallest elements in the array.

IV. The subarray $A[n-i-1, n-1]$ contains the $i+1$ largest elements in the array.

[3 marks]

B. True or false: every balanced tree is height-balanced?

1. True
2. False

[2 marks]

C. True or false: In an AVL tree, every non-leaf node u with height h has a child v with height exactly $h-1$.

1. True
2. False

[2 marks]

D. True or false: In an AVL tree, for every node u with height h, the number of nodes in the subtree rooted at u is at most $2^h + 1$.

1. True
2. False

[2 marks]

E. True or false: In an AVL tree, for every node u with height h, the number of nodes in the subtree rooted at u is at least $2^{h/2}$.

1. True
2. False

[2 marks]

F. Consider a new type of tree, a k-bushy tree. In a k-bushy tree, each node has up to k children. We say that a k-bushy tree is k-height balanced if for every node v and for every child u of v: $height(u) \geq height(v) - k$. If a k-bushy tree is k-height balanced, then which of the following provides the best bound on the height of a tree with n nodes (as a function of n and k)? Post-exam-amendment: as in an AVL tree, consider a null child to have height -1. This conditions is not stated quite correctly.

1. $h \leq 2\log_k(n)$
2. $h \leq 2\log_k(n/k)$
3. $h \leq 2\log(n/k)$
4. $h \leq 2\log(n)/k$
5. $h \leq k\log(n)$
6. $h \leq k\log(n/k)$
7. $h \leq k\log_k(n)$
8. $h \leq k\log_k(n/k)$
9. $h \leq k\log(n)/k$

[3 marks]

G. Consider the following (pseudocode) implementation (which behaves like Java) of a routine to test whether an array is sorted. A "Summary" data type holds stores three values: `int min`, `int max`, and `boolean sorted`. Assume the algorithm is invoked properly with an array A containing $n > 1$ integers, with `low`$= 0$ and `high`$= n-1$.

```
1.    Summary testSort(int[] A, int low, int high)
2.        if (low == high)
3.              summary.min = A[low];
4.              summary.max = A[high];
5.              summary.sorted = true;
6.              return summary;
7.        mid = low + (high-low)/2;
8.        sLeft = testSort(A, low, mid);
9.        sRight = testSort(A, mid+1, high);
10.
11.       summary.min = min(sLeft.min, sRight.min);
12.       summary.max = max(sLeft.max, sRight.max);
13.
14.       if (sLeft.max > sRight.min) summary.sorted = false;
15.       else summary.sorted = (sLeft.sorted && sRight.sorted);
```

```
16.
17.       return summary;
```

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

1. The algorithm terminates and works correctly: when invoked on array A, it returns a summary such that `summary.sorted=true` if and only if A is sorted.
2. The algorithm terminates and works sometimes: when invoked on array A, it always returns a summary such that `summary.sorted=true` if A is sorted, but sometimes returns `summary.sorted=true` when A is not sorted.
3. The algorithm terminates and works sometimes: when invoked on array A, it always returns a summary such that `summary.sorted=false` if A is not sorted, but sometimes returns `summary.sorted=false` when A is sorted.
4. The algorithm does not always terminate.
5. None of the above is a good description of what A guarantees.

[3 marks]

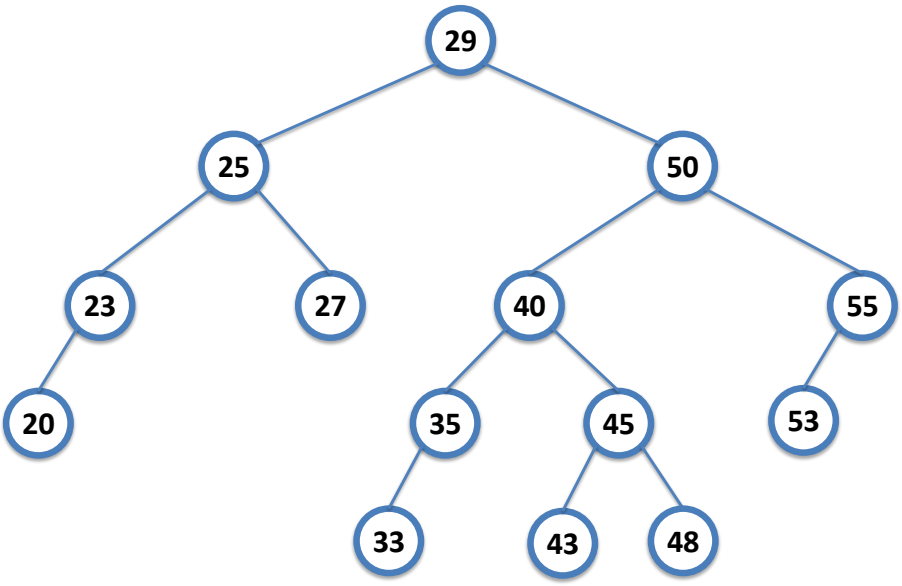
H. What is the worst-case running time of the `testSearch` routine? (For the purpose of this question, consider only inputs on which the function terminates—if there are inputs on which it does not terminate, ignore them.) Assume the algorithm is invoked properly with an array A containing $n > 1$ integers, with `low`$= 0$ and `high`$= n-1$.

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

[3 marks]

Question 5: Oak and Ash [14 marks]

A. The tree below is an AVL tree. After key 42 is inserted (and before rebalancing occurs), which node(s), if any, are out of balance? 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.)



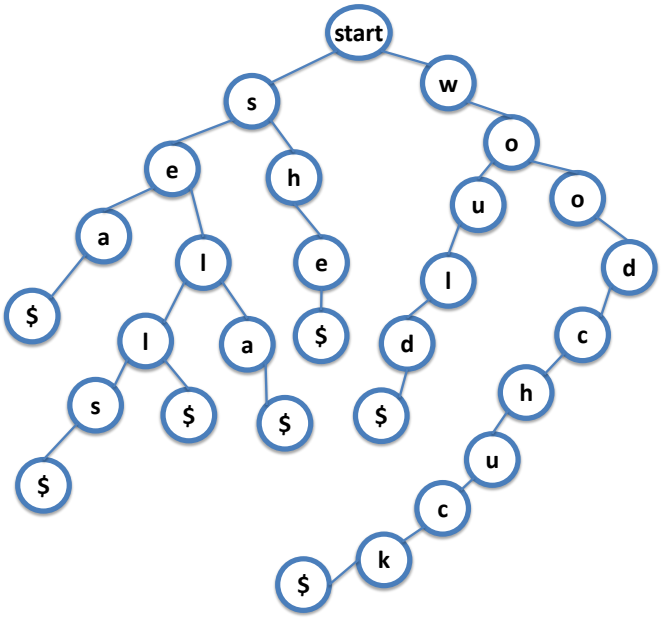
[2 marks]

B. For the same tree as the previous question, after key 42 is inserted, Which rotation(s) occur, if any? (Identify a rotation with the root of the subtree rotated. For example, a `right-rotate(29)` would move 29 down and 25 up.)

1. left-rotate(29)
2. right-rotate(50)
3. left-rotate(40)
4. right-rotate(45)
5. left-rotate(29), right-rotate(40)
6. left-rotate(29), left-rotate(50)
7. right-rotate(50), left-rotate(40)
8. right-rotate(45), left-rotate(40)
9. left-rotate(40), right-rotate(50)
10. No rotations occur.
11. None of the above.

[2 marks]

C. Consider the following trie. How many strings are stored in the trie?



[2 marks]

D. True or false: the string "sell" is in the trie?

1. True
2. False

[2 marks]

E. True or false: the string "sells" is in the trie?

1. True
2. False

[2 marks]

F. True or false: the string "wood" is in the trie?

1. True
2. False

[2 marks]

G. In general, the number of nodes in a trie (not counting the start node) is: Post-exam-amendment: do not count trie "nodes" that only contain "null terminator" characters. This question was unclear as stated during the exam.

1. Greater than the sum of the lengths of all the strings in the trie, collectively.
2. Less than or equal to the sum of the lengths of all the strings in the trie, collectively.
3. Cannot be determined—it depends on the strings.

[2 marks]

Question 6: One, two, three, four... [7 marks]

Randolph Key wants to augment an AVL tree so that you can determine, for a given node u, how many nodes there are in the tree that have keys larger than u's key. For example, if the tree contained the integers $\{1,2,3,4,5,6,7,8,9,10\}$, then if you executed a `largerThan(7)` query, it would return 3.

Randolph has already augmented the tree to store the `size` of each subtree in each node: `u.size` contains the number of nodes in the subtree rooted at u. And Randolph has already implemented the following skeleton code. You just need to fill in the last few missing pieces. For each blank, choose a correct option so that they entire function works correctly.

You can assume that a node x has five subfields: a key, a value, a size, a left child (possibly null) and a right child (possibly null). You can assume that keys are distinct. The operation is initiated by calling `largerThan(key, root)`. The function should return the number of keys in the subtree of `Node x` that are strictly larger than `key`.

```
int largerThan(int key, Node x)
    if (x == null) return 0;
    if (key < x.key) {
        return [ A ] + largerThan(key, [ B ]);
    }
    if (key > x.key) {
        return [ C ] + largerThan(key, [ D ]);
    }
    if (key == x.key) {
        if ( [ E ] == null) {
            return [ F ] ;
        }
        else return [ G ] ;
    }
```

A. A = ??

1. 0
2. 1
3. `x.left`
4. `x.right`
5. `x.key`
6. `x.size`
7. `x.left.key`
8. `x.right.key`
9. `x.left.size`
10. `x.right.size`
11. `x.left.size+1`
12. `x.right.size+1`

[1 mark]

B. B = ?

1. 0
2. 1
3. `x.left`
4. `x.right`
5. `x.key`
6. `x.size`

7. `x.left.key`
8. `x.right.key`
9. `x.left.size`
10. `x.right.size`
11. `x.left.size+1`
12. `x.right.size+1`

[1 mark]

C. C = ?

1. 0
2. 1
3. `x.left`
4. `x.right`
5. `x.key`
6. `x.size`
7. `x.left.key`
8. `x.right.key`
9. `x.left.size`
10. `x.right.size`
11. `x.left.size+1`
12. `x.right.size+1`

[1 mark]

D. D = ?

1. 0
2. 1
3. `x.left`
4. `x.right`
5. `x.key`
6. `x.size`
7. `x.left.key`
8. `x.right.key`
9. `x.left.size`
10. `x.right.size`
11. `x.left.size+1`
12. `x.right.size+1`

[1 mark]

E. E = ?

1. 0
2. 1
3. `x.left`
4. `x.right`
5. `x.key`
6. `x.size`
7. `x.left.key`
8. `x.right.key`
9. `x.left.size`
10. `x.right.size`
11. `x.left.size+1`
12. `x.right.size+1`

[1 mark]

F. F = ?

1. 0
2. 1
3. `x.left`
4. `x.right`
5. `x.key`
6. `x.size`
7. `x.left.key`
8. `x.right.key`
9. `x.left.size`
10. `x.right.size`
11. `x.left.size+1`
12. `x.right.size+1`

[1 mark]

G. G = ?

1. 0
2. 1
3. `x.left`
4. `x.right`
5. `x.key`
6. `x.size`
7. `x.left.key`
8. `x.right.key`
9. `x.left.size`
10. `x.right.size`
11. `x.left.size+1`
12. `x.right.size+1`

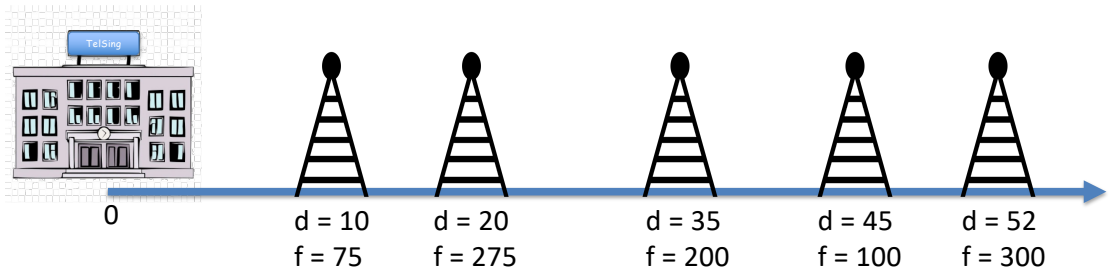
[1 mark]

Question 7: What's the frequency... [8 marks]

Ina Tervalfan is back again trying to fix the problems of intervals in the world. She took CS2040S years ago and is now working for the major telecom provider TelSing. She is expecially interested in the problem of cell phone coverage: she would like to ensure that no two consecutive towers are transmitting at frequencies that are too close. If two nearby towers use similar frequencies, there is interference and coverage is poor. To ensure good service, TelSing wants to choose frequencies that are far enough part.

Her boss sketches the problem as follows:

> You can think of our network as deployed on a long road, starting at our corporate headquarters (location 0) and extending into the distance. A tower t has two attributes: its frequency f_t and its location d_t—i.e., its distance from headquarters. (For the purpose of this problem, we will think of both of these values as real numbers stored as floats.) We need you to build a data structure that supports two operations:
>
> - `build(float freq, float dist)`: builds a new tower with the specified frequency and location. Every tower will be built at a unique location (but some towers may use the same frequency).
> - `freqGap()`: returns the value of the minimum gap in frequencies between adjacent towers. (Two towers are adjacent if there is no tower between them.)
>
> For example, in our network diagram below, you can see we have five towers deployed, and the minimum frequency gap is 75: location 20 and location 35. (If two pairs of adjacent towers have the same minimum gap, that is okay: just return the gap of either of them.) In the future, we'll ask you implement `demolish`, which will allow us to remove towers, and `updateFrequency`, which will allow us to change the frequency of a tower. But today, let's get a minimum viable product up and running with just the two functions above, okay? And make that data structure efficient: we plan to build a lot of towers!



Your job, below, is to help Ina to design this data structure. You may want to read all the parts first, as an answer is only correct if it is part of a correct design for all the parts. (Your answers must fit together to create a single functional data structure.)

A. Ina decides to use an AVL tree as the basic data structure. Her first decision is what to use as the key for the AVL tree. Should the tree be sorted by distance or frequency?

1. distance
2. frequency

[1 mark]

B. Ina quickly implements the AVL tree with the specified key, and storing both the frequency `u.freq` and distance `u.dist` of the tower at node u of the tree. Now she can quickly and efficiently build towers! She realizes, though, that she does not have enough information to efficiently query for the minimum frequency gap. To do that, she is going to need to store some additional information at each node in the tree.

First, she decides that for every node u, she is going to store `u.gap`: the minimum frequency gap of any adjacent towers y and z in the subtree rooted at u. What other information would be useful to store at a node u?

I. the size of the subtree rooted at u (`u.size`)

II. the minimum distance between two adjacent towers in the subtree rooted at u (`u.minAdjDist`)

III. the location of the tower neareast to u (`u.nbr`)

IV. the minimum and maximum frequencies in the subtree rooted at u (`u.minFreq`, `u.maxFreq`)

V. the minimum and maximum distances in the subtree rooted at u (`u.minDist`, `u.maxDist`)

VI. the frequencies of the towers at minimum and maximum distances in the subtree rooted at u (`u.minDistFreq`, (`u.maxDistFreq`))

[2 marks]

Ina next wants to design a function to compute the minimum frequency gap `u.gap` at node u in $O(1)$ time based only on the information stored at u and in u's left and right children (`u.left` and `u.right`). (Obviously this computation should not include `u.gap` itself!). You can assume that the children of u have all their data correct, and all the information at u is correct except for `u.gap`. (It is important that this information can be efficiently updated during insertions and rotations.)

For the purpose of the code below, assume that node u has both a left and a right child. (In the proper production code, we will check if the children exist before accessing them to avoid errors.) `Math.abs` computes the absolute value, `Math.max` the maximum of all its arguments, and `Math.min` the minimum of all its arguments.

```
float minFreqGap(Node u)
    leftInfo = Math.abs(u.left.[ A ] - [ B ]);
    rightInfo = Math.abs(u.right.[ C ] - [ D ]);

    gap = [ E ];

    return Math.min(gap, u.left.gap, u.right.gap);
```

C. A = ??

[1 mark]

D. B = ??

[1 mark]

E. C = ??

[1 mark]

F. D = ??

[1 mark]

G. E = ??

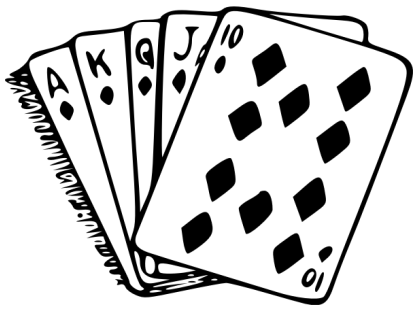
1. `Math.abs(leftInfo - rightInfo)`
2. `Math.abs(leftInfo + rightInfo)`
3. `Math.max(leftInfo, rightInfo)`
4. `Math.min(leftInfo, rightInfo)`

[1 mark]

Question 8: Just for Fun: The Dark Room [0 marks]

You are in a completely dark room. You can't see anything, not even your hands when you wave them in front of your face. You are sitting in a chair, and you can feel a table in front of you. On that table your hands run across what feels like a deck of cards. A voice overhead announces, "Before you is a standard deck of 52 cards. Thirteen of the cards are face up, the rest are face down. You will also find two boxes on the table. You must place one stack of cards in each of the two boxes (and each stack must contain at least one card). You will win if both boxes contain the same number of face up cards. Otherwise, you lose." You do not want to know what will happen if you lose.

What is your best strategy for winning? [0 mark]



— END OF PAPER —

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

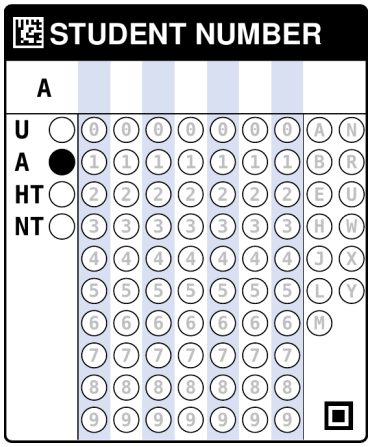
Midterm Assessment — Answer Sheet

2023/2024 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 **FIFTEEN (15) 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. The questions are provided in the question booklet, and if there is any inconsistency in the question, please refer to the question booklet. If there is any inconsistency in the answers, refer to the answer sheet.
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. Please fill in the bubbles properly. **Marks may be deducted** for unclear answers.



For Examiner's Use Only

Question	Marks
Q1	/ 12
Q2	/ 22
Q3	/ 17
Q4	/ 20
Q5	/ 14
Q6	/ 7
Q7	/ 8
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]

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

Question 2C 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 2D True or false: [2 marks]

◯ True ◯ False

Question 2E Definition of big-O notation: [2 marks]

1. ◯ $\exists\, c > 0, \forall n_0 > 0, \forall n \geq n_0 \;:\; f(n) \leq cg(n)$
2. ◯ $\exists\, c > 0, \forall n \geq 0 \;:\; f(n) \leq cg(n)$
3. ◯ $\exists\, n_0 > 0, \forall n \geq n_0, \exists c > 0 \;:\; f(n) \leq cg(n)$
4. ◯ $\exists\, c > 0, \exists\, n_0 > 0, \forall n \geq n_0 \;:\; f(n) \leq cg(n)$
5. ◯ $\exists\, n_0 > 0, \forall n \geq n_0 \;:\; f(n) \leq g(n)$

Question 2F How does f relate to n^2? [2 marks]

1. ◯ $f(n) = O(n^2)$
2. ◯ $f(n) = \Omega(n^2)$
3. ◯ Both are true: $f(n) = O(n^2)$ and $f(n) = \Omega(n^2)$.
4. ◯ None of the above are true.

Question 2G Tightest bound from the available options: [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 2H Tightest bound from the available options: [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 2I Asymptotic running time. [2 marks]

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

Question 2J Asymptotic running time. [2 marks]

- ◯ $O(n)$
- ◯ $O(n \log(m/n))$
- ◯ $O(m \log n)$
- ◯ $O(n \log n)$
- ◯ $O((n/m) \log(n/m))$
- ◯ $O(n^2)$
- ◯ $O(n \log m)$
- ◯ $O((m/n) \log(m/n))$
- ◯ $O(n \log(n/m))$
- ◯ $O((n/m) \log(n))$

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

- ◯ $T(n) \leq T(n/3) + O(n)$.
- ◯ $T(n) \leq 3T(n/3) + O(1)$.
- ◯ $T(n) \leq T(n/3) + O(n^2)$.
- ◯ $T(n) \leq T(n/3) + O(1)$.
- ◯ $T(n) \leq 3T(n/3) + O(n^2)$.
- ◯ $T(n) \leq 3T(n/3) + O(n)$.
- ◯ $T(n) \leq 3T(n) + O(n)$.
- ◯ None of the above.

Question 3A Worst-case movement-cost for InsertionSort: [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 3B Worst-case movement-cost for SelectionSort: [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 3C QuickSort on specially arranged deck: [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 3D InsertionSort on specially ordered deck: [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 3E Rotations on insertion or deletion. [2 marks]

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

Question 3F Inserting into a $(2,4)$-tree. [3 marks]

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

Question 4A Good loop invariant? [3 marks]

- ◯ Option I.
- ◯ Option II.
- ◯ Option III.
- ◯ Option IV.
- ◯ Option I and IV.
- ◯ Option II and III.
- ◯ Option II and IV.
- ◯ None of the above.

Question 4B True or false: balanced [2 marks]

◯ True ◯ False

Question 4C True or false: heights [2 marks]

◯ True ◯ False

Question 4D True or false: max number of nodes [2 marks]

◯ True ◯ False

Question 4E True or false: min number of nodes [2 marks]

◯ True ◯ False

Question 4F Height of a k-balanced k-bushy tree. [3 marks]

◯ $h \leq 2\log_k(n)$ ◯ $h \leq 2\log(n)/k$ ◯ $h \leq k\log_k(n)$
◯ $h \leq 2\log_k(n/k)$ ◯ $h \leq k\log(n)$ ◯ $h \leq k\log_k(n/k)$
◯ $h \leq 2\log(n/k)$ ◯ $h \leq k\log(n/k)$ ◯ $h \leq k\log(n)/k$

Question 4G What does testSort guarantee? [3 marks]

◯ The algorithm terminates and works correctly: when invoked on array A, it returns a summary such that summary.sorted=true if and only if A is sorted.

◯ The algorithm terminates and works sometimes: when invoked on array A, it always returns a summary such that summary.sorted=true if A is sorted, but sometimes returns summary.sorted=true when A is not sorted.

◯ The algorithm terminates and works sometimes: when invoked on array A, it always returns a summary such that summary.sorted=false if A is not sorted, but sometimes returns summary.sorted=false when A is sorted.

◯ The algorithm does not always terminate.

◯ None of the above is a good description of what A guarantees.

Question 4H testSearch performance? [3 marks]

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

Question 5A Highest out-of-balance node? [2 marks]

◯ 29 ◯ 27 ◯ 35 ◯ 43
◯ 25 ◯ 40 ◯ 45 ◯ 48
◯ 50 ◯ 55 ◯ 53 ◯ None of these listed.
◯ 23 ◯ 20 ◯ 33

Question 5B How to balance? [2 marks]

- ◯ left-rotate(29)
- ◯ right-rotate(50)
- ◯ left-rotate(40)
- ◯ right-rotate(45)
- ◯ left-rotate(29), right-rotate(40)
- ◯ left-rotate(29), left-rotate(50)
- ◯ right-rotate(50), left-rotate(40)
- ◯ right-rotate(45), left-rotate(40)
- ◯ left-rotate(40), right-rotate(50)
- ◯ No rotations occur.
- ◯ None of the above.

Question 5C Number of strings in the trie? [2 marks]

- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ 11
- ◯ 12 or more

Question 5D True or false: sell [2 marks]

◯ True ◯ False

Question 5E True or false: sells [2 marks]

◯ True ◯ False

Question 5F True or false: wood [2 marks]

◯ True ◯ False

Question 5G Number of nodes in a trie [2 marks]

- ◯ Greater than the sum of the lengths of all the strings
- ◯ Less than or equal to the sum of the lengths of all the strings
- ◯ Cannot be determined

Question 6A A = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

Question 6B B = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

Question 6C C = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

Question 6D D = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

Question 6E E = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

Question 6F F = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `x.left`
- ◯ `x.right`
- ◯ `x.key`
- ◯ `x.size`
- ◯ `x.left.key`
- ◯ `x.right.key`
- ◯ `x.left.size`
- ◯ `x.right.size`
- ◯ `x.left.size+1`
- ◯ `x.right.size+1`

Question 6G G = ?

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `x.left`
- ◯ `x.right`
- ◯ `x.key`
- ◯ `x.size`
- ◯ `x.left.key`
- ◯ `x.right.key`
- ◯ `x.left.size`
- ◯ `x.right.size`
- ◯ `x.left.size+1`
- ◯ `x.right.size+1`

Question 7A Key for the AVL tree

[1 marks]

- ◯ distance
- ◯ frequency

Question 7B Additional data to be stored

[2 marks]

- ◯ I
- ◯ II
- ◯ III
- ◯ IV
- ◯ V
- ◯ VI
- ◯ I and II
- ◯ I, II, and III
- ◯ I, II, and IV
- ◯ I, II, and V
- ◯ I, II, and VI
- ◯ II and III
- ◯ II, III, and IV
- ◯ II, III, and V
- ◯ II and IV
- ◯ II and V
- ◯ II and VI
- ◯ III and IV
- ◯ III and V
- ◯ IV and V
- ◯ IV and VI
- ◯ IV, V, and VI

Question 7C A = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

Question 7D B = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

Question 7E C = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

Question 7F D = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

Question 7G E = ??

[1 marks]

- ◯ `Math.abs(leftInfo - rightInfo)`
- ◯ `Math.abs(leftInfo + rightInfo)`
- ◯ `Math.max(leftInfo, rightInfo)`
- ◯ `Math.min(leftInfo, rightInfo)`

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.

(MergeSort)

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

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

(None of the above (HeapSort))

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

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

(Bubblesort)

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

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

(QuickSort)

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

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

(InsertionSort)

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

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

(SelectionSort)

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

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

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

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

Question 2C 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 2D True or false: [2 marks]

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

Question 2E Definition of big-O notation: [2 marks]

1. ◯ $\exists\, c > 0, \forall n_0 > 0, \forall n \geq n_0 \;:\; f(n) \leq cg(n)$
2. ◯ $\exists\, c > 0, \forall n \geq 0 \;:\; f(n) \leq cg(n)$
3. ◯ $\exists\, n_0 > 0, \forall n \geq n_0, \exists c > 0 \;:\; f(n) \leq cg(n)$
4. ◯ $\exists\, c > 0, \exists\, n_0 > 0, \forall n \geq n_0 \;:\; f(n) \leq cg(n)$
5. ◯ $\exists\, n_0 > 0, \forall n \geq n_0 \;:\; f(n) \leq g(n)$

$\exists c > 0, \exists n_0 > 0, \forall n \geq n_0 \;:\; f(n) \leq cg(n)$

Question 2F How does f relate to n^2? [2 marks]

1. ◯ $f(n) = O(n^2)$
2. ◯ $f(n) = \Omega(n^2)$
3. ◯ Both are true: $f(n) = O(n^2)$ and $f(n) = \Omega(n^2)$.
4. ◯ None of the above are true.

Neither is true.

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

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

$O(n)$

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

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

$O(\log n)$

Question 2I Asymptotic running time. [2 marks]

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

$O(n^2)$

Question 2J Asymptotic running time. [2 marks]

◯ $O(n)$	◯ $O(n\log(m/n))$	◯ $O(m\log n)$
◯ $O(n\log n)$	◯ $O((n/m)\log(n/m))$	◯ $O(n^2)$
◯ $O(n\log m)$	◯ $O((m/n)\log(m/n))$	
◯ $O(n\log(n/m))$	◯ $O((n/m)\log(n))$	

$O(n\log(n/m))$

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

◯ $T(n) \leq T(n/3) + O(n)$.	◯ $T(n) \leq 3T(n/3) + O(1)$.
◯ $T(n) \leq T(n/3) + O(n^2)$.	◯ $T(n) \leq T(n/3) + O(1)$.
◯ $T(n) \leq 3T(n/3) + O(n^2)$.	◯ $T(n) \leq 3T(n/3) + O(n)$.
◯ $T(n) \leq 3T(n) + O(n)$.	◯ None of the above.

$T(n) \leq 3T(n/3) + O(n)$

Question 3A Worst-case movement-cost for InsertionSort: [3 marks]

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

$O(n^2)$

Question 3B Worst-case movement-cost for SelectionSort: [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 3C QuickSort on specially arranged deck: [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 3D InsertionSort on specially ordered deck: [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 3E Rotations on insertion or deletion. [2 marks]

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

$O(\log n)$

Question 3F Inserting into a $(2,4)$-tree. [3 marks]

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

$O(\log n)$

Question 4A Good loop invariant? [3 marks]

◯ Option I.
◯ Option II.
◯ Option III.
◯ Option IV.
◯ Option I and IV.
◯ Option II and III.
◯ Option II and IV.
◯ None of the above.

Options II and III.

Question 4B True or false: balanced [2 marks]

◯ True ◯ False False. Every height-balanced tree is balanced.

Question 4C True or false: heights [2 marks]

◯ True ◯ False True.

Question 4D True or false: max number of nodes [2 marks]

◯ True ◯ False False.

Question 4E True or false: min number of nodes [2 marks]

◯ True ◯ False True.

Question 4F Height of a k-balanced k-bushy tree. [3 marks]

◯ $h \leq 2\log_k(n)$	◯ $h \leq 2\log(n)/k$	◯ $h \leq k\log_k(n)$
◯ $h \leq 2\log_k(n/k)$	◯ $h \leq k\log(n)$	◯ $h \leq k\log_k(n/k)$
◯ $h \leq 2\log(n/k)$	◯ $h \leq k\log(n/k)$	◯ $h \leq k\log(n)/k$

$h \leq k\log_k(n)$

Question 4G What does testSort guarantee? [3 marks]

◯ The algorithm terminates and works correctly: when invoked on array A, it returns a summary such that summary.sorted=true if and only if A is sorted.

◯ The algorithm terminates and works sometimes: when invoked on array A, it always returns a summary such that summary.sorted=true if A is sorted, but sometimes returns summary.sorted=true when A is not sorted.

◯ The algorithm terminates and works sometimes: when invoked on array A, it always returns a summary such that summary.sorted=false if A is not sorted, but sometimes returns summary.sorted=false when A is sorted.

◯ The algorithm does not always terminate.

◯ None of the above is a good description of what A guarantees.

The algorithm terminates and works correctly.

Question 4H testSearch performance? [3 marks]

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

$O(n)$

Question 5A Highest out-of-balance node? [2 marks]

◯ 29	◯ 27	◯ 35	◯ 43
◯ 25	◯ 40	◯ 45	◯ 48
◯ 50	◯ 55	◯ 53	◯ None of these listed.
◯ 23	◯ 20	◯ 33	

29

Question 5B How to balance? [2 marks]

- ◯ left-rotate(29)
- ◯ right-rotate(50)
- ◯ left-rotate(40)
- ◯ right-rotate(45)
- ◯ left-rotate(29), right-rotate(40)
- ◯ left-rotate(29), left-rotate(50)
- ◯ right-rotate(50), left-rotate(40)
- ◯ right-rotate(45), left-rotate(40)
- ◯ left-rotate(40), right-rotate(50)
- ◯ No rotations occur.
- ◯ None of the above.

`left-rotate(40)`, `right-rotate(50)`

Question 5C Number of strings in the trie? [2 marks]

- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ 11
- ◯ 12 or more

7

Question 5D True or false: sell [2 marks]

◯ True ◯ False True.

Question 5E True or false: sells [2 marks]

◯ True ◯ False True.

Question 5F True or false: wood [2 marks]

◯ True ◯ False False.

Question 5G Number of nodes in a trie [2 marks]

- ◯ Greater than the sum of the lengths of all the strings
- ◯ Less than or equal to the sum of the lengths of all the strings
- ◯ Cannot be determined

Less than or equal to the number of characters.

Question 6A A = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

`x.right.size+1`

Question 6B B = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

`x.left`

Question 6C C = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

`0`

Question 6D D = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

`x.right`

Question 6E E = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

`x.right`

Question 6F F = ??

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

`0`

Question 6G G = ?

[1 marks]

◯ 0	◯ `x.key`	◯ `x.left.size`
◯ 1	◯ `x.size`	◯ `x.right.size`
◯ `x.left`	◯ `x.left.key`	◯ `x.left.size+1`
◯ `x.right`	◯ `x.right.key`	◯ `x.right.size+1`

`x.right.size`

Question 7A Key for the AVL tree

[1 marks]

◯ distance	◯ frequency

`distance`

Question 7B Additional data to be stored [2 marks]

- ◯ I
- ◯ II
- ◯ III
- ◯ IV
- ◯ V
- ◯ VI
- ◯ I and II
- ◯ I, II, and III
- ◯ I, II, and IV
- ◯ I, II, and V
- ◯ I, II, and VI
- ◯ II and III
- ◯ II, III, and IV
- ◯ II, III, and V
- ◯ II and IV
- ◯ II and V
- ◯ II and VI
- ◯ III and IV
- ◯ III and V
- ◯ IV and V
- ◯ IV and VI
- ◯ IV, V, and VI

VI: the frequency of the minimum and maximum distance towers

Question 7C A = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

`maxDistFreq`

Question 7D B = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

`u.freq`

Question 7E C = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

`minDistFreq`

Question 7F D = ??

[1 marks]

- ◯ 0
- ◯ 1
- ◯ `freq`
- ◯ `dist`
- ◯ `size`
- ◯ `minDist`
- ◯ `nbr`
- ◯ `minFreq`
- ◯ `maxFreq`
- ◯ `minDist`
- ◯ `maxDist`
- ◯ `minDistFreq`
- ◯ `maxDistFreq`
- ◯ `u.freq`
- ◯ `u.dist`
- ◯ `u.size`
- ◯ `u.minDist`
- ◯ `u.nbr`
- ◯ `u.minFreq`
- ◯ `u.maxFreq`
- ◯ `u.left.freq`
- ◯ `u.left.dist`
- ◯ `u.left.size`
- ◯ `u.left.minDist`
- ◯ `u.left.nbr`
- ◯ `u.left.minFreq`
- ◯ `u.left.maxFreq`
- ◯ `u.left.minDist`
- ◯ `u.left.maxDist`
- ◯ `u.left.minDistFreq`
- ◯ `u.left.maxDistFreq`
- ◯ `u.right.freq`
- ◯ `u.right.dist`
- ◯ `u.right.size`
- ◯ `u.right.minDist`
- ◯ `u.right.nbr`
- ◯ `u.right.minFreq`
- ◯ `u.right.maxFreq`
- ◯ `u.right.minDist`
- ◯ `u.right.maxDist`
- ◯ `u.right.minDistFreq`
- ◯ `u.right.maxDistFreq`

`u.freq`

Question 7G E = ??

[1 marks]

- ◯ `Math.abs(leftInfo - rightInfo)`
- ◯ `Math.abs(leftInfo + rightInfo)`
- ◯ `Math.max(leftInfo, rightInfo)`
- ◯ `Math.min(leftInfo, rightInfo)`

`Math.min(leftInfo, rightInfo)`

Question 8

[0 marks]

The Dark Room

Cool solution!