Midterm Exam

Spring 2026

Name Answer Key

Net ID _____ (@uw.edu)

Academic Integrity: You may not use any resources on this exam except for your one-page (front and back) reference sheet, writing instruments, your own brain, and the exam packet itself. This exam is otherwise closed notes, closed neighbor, closed electronic devices, etc.. The last two pages of this exam provide a list of potentially helpful identities as well as room for scratch work (respectively). Please detach those last two pages from the exam packet. No markings on these last two pages will be graded. Your answer for each question should fit in the answer box provided.

Instructions: Before you begin, **Put your name and UW Net ID at the top of this page.** Make sure that your name and ID are LEGIBLE. Please ensure that all of your answers appear within the boxed area provided.

Section	Max Points
ADTs and Data Structures	7
Asymptotic Analysis	14
Heaps	9
AVL Trees	10
Algorithms	6
Extra Credit	(+1)
Total	46

Section 1: ADTs

(4 pts)**Question 1: Circular Array Analysis**

You are implementing a `CircularArrayQueue` class. The class will have the following fields:

– `data`: the array which stores the elements of the queue.

– `front`: the index of the least recently added element

– `back`: the index of the most recently added element

– `size`: the number of elements in the queue.

When the `data` array is full, you must resize it to twice its current capacity.

In the first three subquestions below you will complete the following `enqueue` implementation by putting the missing code in the answer boxes corresponding to each blank.

```
1     public void enqueue(T val) {
2         if (size == data.length) {
3             // Q 1.1: Resize the array
4             T[] newArr = (T[]) new Object[____blank 1____];
5
6             // Q 1.2: Copy elements over
7             for (int i = 0; i < size; i++) {
8                 newArr[i] = data[____blank 2____];
9             }
10
11            data = newArr;
12            front = 0;
13            back = size - 1;
14        }
15
16        // Q 1.3: Update back and insert
17        back = ____blank 3____;
18        data[back] = val;
19        size++;
20    }
```

1. Code for blank 1: data.length * 2

2. Code for blank 2: (front + i) % data.length

3. Code for blank 3: (back + 1) % data.length

4. Give the worst-case running time for a single `enqueue` operation, then describe the conditions that lead to the worst-case scenario.

   Worst-case Running Time: $\Theta\left(\boxed{n}\right)$

   Scenario Description:

   When the array is full, an enqueue triggers a resize. The system must allocate a new array and copy all n existing elements into the new memory space. This is a linear scan of the old array.

(3 pts)**Question 2: ADT Applications**
For each scenario below, identify which of the ADT options provided is the most appropriate choice.
Options: Stack, Queue, Priority Queue, Ordered Dictionary

1. You are implementing the "Undo/Redo" functionality for Visual Studio Code. Each time a change is applied, the state of the code editor is saved. When the user hits Ctrl+Z, the system must revert to the state immediately preceding the current one.

   Stack

2. The Suzzallo library needs a new inventory system. You're helping to build a system to support looking up books by their unique 13-digit ISBN. Additionally, the manager needs to frequently print a report of all books within a specific ISBN range (e.g., all computer science textbooks starting at 332-...) in numerical order.

   Ordered Dictionary

3. You've just been hired for an internship at Happle! You've been tasked to develop a data-syncing service for the hiPhone 20. Every time a file is modified, it is added to a list of files to be transmitted to the hiCloud for backup. In the system preferences, users can indicate which type of files they most care to back up so that those can be transmitted first. Any file that's marked as "system critical" must be moved to the absolute front of the list to be synced immediately.

   Priority Queue

Section 2: Asymptotic Analysis

(5 pts)**Question 3: Tree Method**
Suppose that the running time of an algorithm is expressed by the recurrence relation:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + 2n$$

$$T(1) = 1$$

For the following questions, use the tree method to solve the recurrence relation. We have broken up the process into subquestions to guide you through your answer. You may assume that $n/3$ is always an integer (and therefore $2n/3$ is an integer as well). You may also ignore the work done at the lowest level of the tree.

1. Sketch the tree in space below. Include at least the first 3 levels of the tree (the root, its children, and its grandchildren), make clear what the input size is for each recursive call as well as the work per call.

   ...

2. Indicate exactly the total amount of work done at level i of the tree (define the root to be level 0). Include all constants and non-dominant terms.

   $2n$ work per level.

3. Indicate the height of the tree.

   Longest branch is on $T(2n/3)$, so height is $\log_{3/2} n$.

4. Give a simplified Θ bound on the solution. A simplified expression should have neither constant coefficients nor non-dominant terms.

   $\Theta\left(T(n) = \sum_{i=0}^{\log_{3/2} n - 1} 2n = 2n \cdot \sum_{i=0}^{\log_{3/2} n - 1} 1 = 2n(\log_{3/2} n) \Rightarrow O(n \log n) \right)$

(6 pts)**Question 4: Code Analysis**

Give a Θ bound on the best-case and worst-case running times for the following functions. (Hint: queues and priority queues may have duplicate elements, dictionaries may not.)

1.
```
public int dominicus(CircularArrayQueue<Integer> queue){
    BinaryMinHeap<Integer> heap = new BinaryMinHeap<>();
    heap.insert(-1);
    while(!queue.isEmpty()){
        int x = queue.dequeue()
        heap.insert(x);
    }
    return heap.peek();
}
```

Best case: $\Theta\left(\boxed{n}\right)$ Worst case: $\Theta\left(\boxed{n \log n}\right)$

2.
```
public int lancelot(AVLTree<Integer,Integer> avl) {
    BinaryMinHeap<Integer> heap = new BinaryMinHeap<>();
    lancelot(avl.root, heap);
    return heap.peek();
}
private void lancelot(TreeNode<Integer> curr, BinaryMinHeap<Integer> heap){
    if(curr != null){
        lancelot(curr.left, heap);
        heap.insert(curr.key);
        lancelot(curr.right, heap);
    }
    else{
        heap.insert(-1);
    }
}
```

Best case: $\Theta\left(\boxed{n}\right)$ Worst case: $\Theta\left(\boxed{n}\right)$

3.
```
public void charlemagne(BinaryMinHeap<Integer> heap) {
    AVLTree<Integer, Integer> avl = new AVLTree<>();
    avl.insert(-1);
    while(!heap.isEmpty()){
        int x = heap.extract()
        avl.insert(x, x);
    }
    return avl.size();
}
```

Best case: $\Theta\left(\boxed{n}\right)$ Worst case: $\Theta\left(\boxed{n \log n}\right)$

(3 pts)**Question 5: c and n_0**

For each subquestion we give $f(n)$, $g(n)$, and either c or n_0. Our goal is to identify a pair of c, n_0 which we could use to show $f(n) = O(g(n))$. If we gave a value for c then select the **smallest value** for n_0 (from the options provided) with which we could show $f(n) = O(g(n))$. If we gave a value for n_0, then select the **smallest value** for c (from the options provided) with which we could show $f(n) = O(g(n))$. Indicate your answer by filling in the corresponding circle to its left.

1. $f(n) = n$, $g(n) = 2n$, $n_0 = 1$

   ○ $c = 0$ ○ $c = \frac{1}{4}$ ● $c = \frac{1}{2}$ ○ $c = 1$ ○ $c = 2$

2. $f(n) = log_4(n)$, $g(n) = log_{12}(n)$, $n_0 = 1$

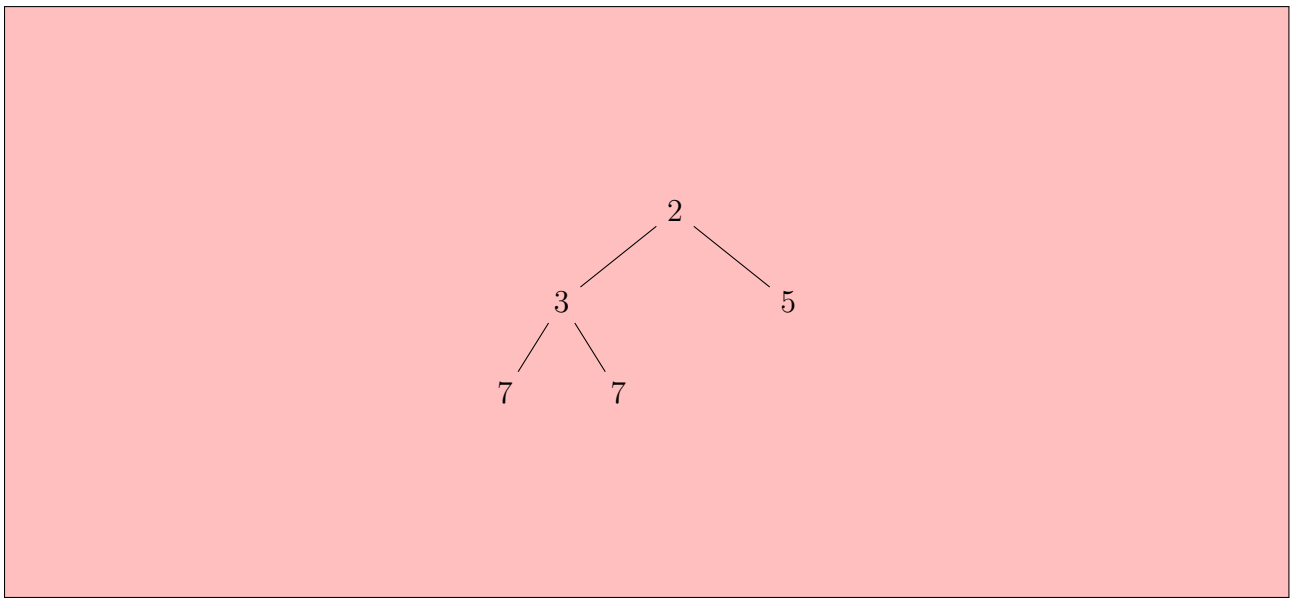
   ○ $c = 0$ ○ $c = \frac{1}{log_4(12)} \approx 0.56$ ○ $c = 1$ ● $c = \frac{1}{log_{12}(4)} \approx 1.8$

3. $f(n) = log_8(n)$, $g(n) = (\log_8 n)^2$, $c = 1$

   ○ $n_0 = 1$ ○ $n_0 = 2$ ○ $n_0 = 4$ ● $n_0 = 8$ ○ $n_0 = 16$

Section 3: Heaps

(3 pts)**Question 6: Heap Inserts**
Draw the Binary Min-Heap that results from performing the following operations in order: insert(1), insert(3), insert(5), insert(7), insert(2), insert(7), extract()



(3 pts)**Question 7: Heap Arithmetic**
Assume that we have a Binary Min Heap with 2048 elements. Also assume that the binary min heap is stored using **1-indexed array representation** (so the root appears at index 1). (hint: $2048 = 2^{11}$)

Answer the following about this min heap.

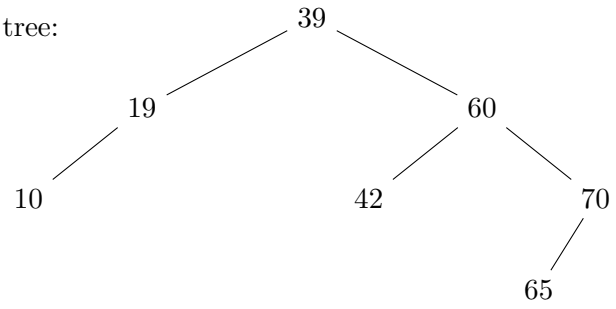
1. What is the height of the heap (Hint: A heap with one node has height 0)? 11
2. At what index would we find the sibling of index 1000? 1001
3. How many elements are on the last level of the heap? 1

(3 pts)**Question 8: Possible places for elements**
Suppose we have a **Binary Max Heap** that contains 4 elements A, B, C, D such that $A < B < C < D$.

1. Which of the following element(s) could possibly be the **root**? Select all that apply.
   ○ A ○ B ○ C ● D ○ None of these
2. Which of the following element(s) could possibly be **exactly one edge from the root**? Select all that apply.
   ● A ● B ● C ○ D ○ None of these
3. Which of the following element(s) could possibly be exactly **two edges from the root**? Select all that apply.
   ● A ● B ○ C ○ D ○ None of these

Section 4: AVL Trees

(8 pts)**Question 9: Rotations**
Consider the following AVL tree:



For each question below, indicate the *problem node* and *type of rotation* that would occur if the given insertion(s) are done on the tree above. If there are multiple insertions, the first insertion will *not* result in a rotation, the second insertion *may* result in a rotation. Each question should be considered completely independently (i.e. "reset" to the image between questions).

For "Problem node", write the key corresponding to the problem node. For "Rotation type" Write one of the following to indicate your answer:

– **R** for a single right rotation

– **L** for a single left rotation

– **LR** for a left-then-right double rotation

– **RL** for a right-then-left double rotation

– **NR** for no rotation (write this for the "Problem node" box as well)

1. `insert(63)`

   Problem node: 70 Rotation type: R

2. `insert(75)`
   `insert(76)`

   Problem node: 60 Rotation type: L

3. `insert(41)`

   Problem node: NR Rotation type: NR

4. `insert(50)`
   `insert(45)`

   Problem node: 42 Rotation type: RL

(2 pts)**Question 10: AVL Tree Arithmetic**
Answer the following about the number of nodes in an AVL tree.

1. What is the maximum number of nodes in an AVL tree of height 4 (Hint: a 1-node tree has height 0)?

   31

2. What is the minimum number of nodes in an AVL tree of height 4?

   12

Section 5: Algorithms

(6 pts)**Question 11: Next Biggest Value After Here**

Welcome to the very last section of the exam! You are almost there! For this section we will look at two different attempts to solve the following problem that we call "next biggest value after here".

Input: an integer array called `a` of n **unique** values

Output: a new integer array called `b`, also of length n, where each `b[i]` contains the next biggest value of `a` that is after index `i`.

More precisely, for each index `i` we want to set `b[i] = a[j]` where `j>i` and `a[j]` is the minimum value where `a[i]<a[j]`. We will set `b[i]=null` if there is no `j>i` where `a[i]<a[j]`.

For example, if the input array is `a = [2, 5, 3, 4]`, the output array should be `b = [3, null, 4, null]`.

The following algorithm would correctly solve this problem in $\Theta(n^2)$ time:

```
public Integer[] nextBiggestAfter(int[] a){
    Integer[] b = new Integer[a.length]; //initialize b
    for(int i = 0; i < a.length; i++){
        b[i] = Integer.MAX_VALUE; // initialize b[i] to the maximum integer
        for(int j = i+1; j < a.length; j++){ // for each index j after i
            if(a[j] > a[i]){ // if the value is greater than a[i]
                b[i] = Math.min(a[j], b[i]);
            }
        }
        // if b[i] is still the max int, nothing after i was larger than a[i].
        if(b[i] == Integer.MAX_VALUE)
            b[i] = null;
    }
    return b;
}
```

Questions are on the following page.

1. In an attempt to design a faster algorithm, Jacklyn tried using a Binary Max Heap. She came up with the following algorithm, which is incorrect:

   Start from the back of the array and iterate forward. For each index `i`, insert `a[i]` into the Binary Max Heap then set `b[i]` to be the parent of `a[i]` (if `a[i]` does not have a parent, set `b[i]=null`). Here is some psuedocode for this algorithm:

```
public int[] nextBiggestAfter(int[] a){
    int[] b = new int[a.length];
    BinaryMaxHeap<Integer> heap = new BinaryMaxHeap<>();
    for(int i = a.length-1; i >= 0; i--){
        b[i] = heap.insertAndReturnParent(a[i]);
    }
    return b;
}
```

   Give a counterexample that demonstrates that her algorithm is incorrect. In your answer, identify the input list, the correct output, and the output of Jacklyn's algorithm. (Hint: a counterexample exists with as few as 3 elements in the array).

   $[8, 9, 10]$ which would return $[10, 10, null]$ instead of $[8, 10, null]$

2. While the heap may not have worked, there is a correct algorithm that uses an AVL Tree. In 3-5 sentences or in psuedocode, describe an algorithm that uses an AVL tree to correctly solve the problem that is asymptotically faster than n^2. (Hint: you may refer to any methods that AVL trees had on Exercise 4 without re-implementing them.)

   Going from back to front, insert $a[i]$ into an AVL tree, then set $b[i] = nextKey(a[i])$.

3. Give the worst case running time of your algorithm

   $n \log n$

Section 6: Extra Credit

(+0.5 pts)**Question Extra 1: Before**
In the space below, draw a picture of how you were feeling coming into this exam.

:$

(+0.5 pts)**Question Extra 2: After**
Now draw a picture of how you are feeling coming out of it.

:)

Scratch Work

Nothing written on this page will be graded.

Identities

Nothing written on this page will be graded.

Summations

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \text{ for } |x| < 1$$

$$\sum_{i=0}^{n-1} = \sum_{n}^{i=1} = n$$

$$\sum_{i=0}^{n} i = 0 + \sum_{n}^{i=1} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$$

$$\sum_{i=0}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$$

$$\sum_{i=0}^{n-1} x^i = \frac{1-x^n}{1-x}$$

$$\sum_{i=0}^{n-1} \frac{1}{2^i} = 2 - \frac{1}{2^{n-1}}$$

Logs

$$x^{\log_x(n)} = n$$

$$\log_a(b^c) = c \log_a(b)$$

$$a^{\log_b(c)} = c^{log_b(a)}$$

$$\log_b(a) = \frac{\log_d(a)}{\log_d(b)}$$

