Midterm Exam

Summer 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 must 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	12
Asymptotic Analysis	12
Recurrences	8
Trees	9
Heaps	4
Algorithms	8
Extra Credit	(+1)
Total	53

Section 1: ADTs and Data Structures

(3 pts) **Question 1: ADT Applications**
For each scenario below, identify which ADT is most appropriate.

Options: Stack, Queue, Priority Queue, Ordered Dictionary

Each option may be used zero, one, or multiple times. Each part is worth 1 point.

1. Sasquatch is exploring a maze. At each intersection, it chooses one unexplored path and saves the other available paths for later. If the chosen path leads to a dead end, it retraces its route until it reaches the nearest previous intersection that still has an unexplored path.

   Stack

2. During the World Cup, Jessica assigns each team a championship score based on its current performance. She wants to identify the team with the highest score at any time. No particular ordering is required among teams with the same score.

   Priority Queue

3. Kevin is trying to build muscle and records the amount of protein in each food in his refrigerator. Before dinner, he calculates a target protein range for the meal. He wants to retrieve all foods whose protein amounts fall within that range. Assume no two foods have exactly the same protein amount.

   Ordered Dictionary

(4 pts) **Question 2: Campus Printing System**
A campus printing system stores pending print jobs. Faculty jobs must always be printed before student jobs. Within each group, jobs must be printed in the order in which they were submitted.

1. What ADT or combination of ADTs would you use? You may use more than one instance of an ADT. Specify any ordering rule required by your design.

   Use two queues: one queue for faculty jobs and one queue for student jobs. The ordering rule is that faculty jobs are always printed before student jobs, and jobs within each queue are printed in FIFO order.

   An alternative valid solution is to use one priority queue with a comparison rule that gives faculty jobs higher priority than student jobs, and gives earlier submissions higher priority among jobs in the same group.

2. Complete the following using 1 – 2 short sentences for each operation:

   `submit(job):`

   Enqueue the job into the queue corresponding to its submitter.

   `printNext():`

   Dequeue from the faculty queue if nonempty; otherwise dequeue from the student queue.

(5 pts) **Question 3: Circular Array**

Suppose I wish to implement my circular array so that the index of `back` remains the same after resizing. For example, if I had the list $[a_3, a_4, a_5, a_1, a_2]$, where `front` pointed to a_1 and `back` pointed to a_5, then after resizing, my array should look like

$$[a_3, a_4, a_5, 0, 0, 0, 0, 0, a_1, a_2],$$

so that the back of my queue has the same index as before.

```
1  public void enqueue(T val) {
2      if (size == data.length) {
3          T[] newArr = (T[]) new Object[2*data.length];
4          for (int i = 0; i < back % data.length ; i++) {
5              newArr[i] = data[____BLANK 1____];
6          }
7          for (int i = ______BLANK 2_____; i < newArr.length ; i++) {
8              newArr[i] = data[____BLANK 3____];
9          }
10         data = newArr;
11         front = ______BLANK 4______;
12     }
13 \\ Update back and insert
14     back = ____BLANK 5____;
15     data[back] = val;
16     size++;
17 }
```

1. Fill in blank 1: i
2. Fill in blank 2: front + data.length
3. Fill in blank 3: i - data.length

4. Fill in blank 4: front + data.length/2

5. Fill in blank 5: back + 1

Section 2: Asymptotics

(4 pts) **Question 4: Big-O Proof**
Prove that $4n^3 + 6n + 3 \in O(n^3)$.

Let $c = 13$ and $n_0 = 1$. Observe that for all $n \geq n_0$, we have that $6n \leq 6n^3$ and $3 \leq 3n^3$. Therefore, we have that for all $n \geq n_0$:

$$4n^3 + 6n + 3 \leq 4n^3 + 6n^3 + 3n^3 = 13n^3 = cn^3,$$

as desired.

(8 pts) **Question 5: Code Analysis**
Give the best and worst case running times for each algorithm below. In each case, assume the size of every input data structure is n (so if the inputs are a tree and a heap, then both have n elements).

1. Consider an algorithm which either populates the heap via an in-order traversal or a reverse in-order traversal depending on the roots value:

```
public void TreeToHeap(TreeNode root, MinHeap heap) {
    if(root != null){
        if((root.key % 2) == 0) {
            TreeToHeap(root.right, heap);
            heap.insert(root);
            TreeToHeap(root.left, heap);
        } else {
            TreeToHeap(root.left, heap);
            heap.insert(root);
            TreeToHeap(root.right, heap);
        }

    }
}
```

Best Case Running Time: $O(n)$

Worst Case Running Time: $O(n \log n)$

2. Consider an algorithm which empties a heap into an AVL tree:

```
public void HeapToTree(Tree avl, MinHeap heap) {
    while (!heap.isEmpty()) {
        avl.insert(heap.extract())
    }
}
```

Best Case Running Time: $O\left(\boxed{n \log n}\right)$

Worst Case Running Time: $O\left(\boxed{n \log n}\right)$

(3 pts) **Question 6: Amortized Analysis**

Suppose I implement the resize function for my ArrayList by increasing the size of my array from n to $n + \lfloor n^{1/4} \rfloor$. Give a tight upper bound on the amortized complexity of the add() method using this resizing strategy.

Suppose my list has n elements, and length $n + n^{1/4}$ (so I just performed a resize). After $n^{1/4}$ adds, I must resize, which takes $n + n^{1/4}$ time. So the amortized complexity is $O((n + n^{1/4})/n^{1/4})) = O(n^{3/4})$

Section 3: Recurrences

(8 pts) **Question 7: Code Analysis and Tree Method**
Give a recurrence for the worst-case asymptotic runtime of the following method as a function of n. The base case is already provided. You do not need to include constants. You will have to solve this recurrence.

```
1   public int funFunction(int n){
2       if(n < 10) {
3           return 100;
4       }
5       int hooray = funFunction(n/2);
6       if (hooray > 10) {
7           for (int i = 0; i < n; i++){
8               hooray += 1;
9           }
10          return 2 * funFunction(2*n/3);
11      } else {
12          for (int i = 0; i < n; i++) {
13              for (int j = i; j < n; j++) {
14                  hooray += 1;
15              }
16          }
17          return funFunction(n/3);
18      }
19  }
```

$$T(n) = \begin{cases} c_0 & \text{if } n < 10, \\ T\left(\frac{n}{2}\right) + T\left(\frac{2n}{3}\right) + n & \text{if } n \geq 10. \end{cases}$$

For the following questions, use the tree method to solve the recurrence relation.

1) Sketch the tree in space below. Include at least the first 3 levels of the tree (i.e. the root, its children, its grandchildren), make clear the input size 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.

$n \cdot \left(\frac{7}{6}\right)^i$

3) Indicate the highest level of the tree in which the base cases occur. In other words, what is the length of the shortest path from the root to a leaf in your tree? Give your answer precisely (not as a $O(\cdot)$ bound) – for full credit your answer must be within ± 1 of the correct answer.

$\log_2(n/10)$

4) Indicate the lowest level of the tree in which the base cases occur. In other words, what is the length of the longest path from the root to a leaf in your tree? Give your answer precisely (not as a $O(\cdot)$ bound) – for full credit your answer must be within ± 1 of the correct answer.

$\log_{3/2}(n/10)$

5) Give a simplified O bound on $T(n)$. When simplified, n should not appear in any exponents. (Hint: keep in mind the leaf depths of your tree. In addition, the log rule $a^{\log_b(c)} = c^{log_b(a)}$ may be helpful!)

$O\left(\sum_{i=0}^{\log_{3/2} n - 1} n \cdot \left(\frac{7}{6}\right)^i \rightarrow n \cdot \left(\frac{7}{6}\right)^{\log_{3/2} n} = n^{1+\log_{3/2}(7/6)}\right)$

Section 4: Trees

(6 pts) **Question 8: AVL Madness**

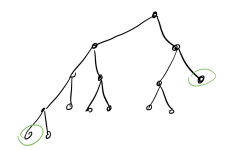
1. Let $\mathsf{depth}(v)$ denote the distance of node v to the root (so $\mathsf{depth}(root) = 0$). What is the maximum difference of depths between any two leaves of a valid AVL tree T? Recall that a leaf is any node that has no children.

$$\max_{\text{leaves } \ell_1,\ell_2} \mathsf{depth}(\ell_1) - \mathsf{depth}(\ell_2) =$$

$\log_\phi(n)/2$??

2. Draw a valid AVL tree that achieves the maximum you described above.

Originally, we thought the answer was 2 here, see the below example:



However, we realized after the exam that this example can be leveraged to create even further examples. The best examples we have so far achieve

$$\max_{\ell_1,\ell_2} \mathsf{depth}(\ell_1) - \mathsf{depth}(\ell_2) = (height + 1)/2.$$

In general, the height could be $\log_\phi(n)$, where ϕ is the golden ratio.
This may not be the correct answer still.

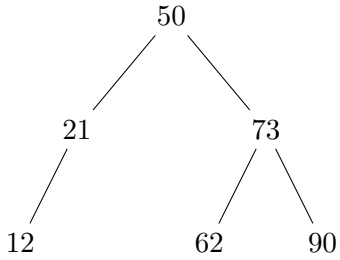
3. Suppose we relax our definition of AVL trees to require only that the difference in height of the two subtrees of every node is at most C (normally, $C = 1$).

   For this relaxed definition, what is

$$\max_{\text{leaves } \ell_1,\ell_2} \mathsf{depth}(\ell_1) - \mathsf{depth}(\ell_2) \quad ?$$

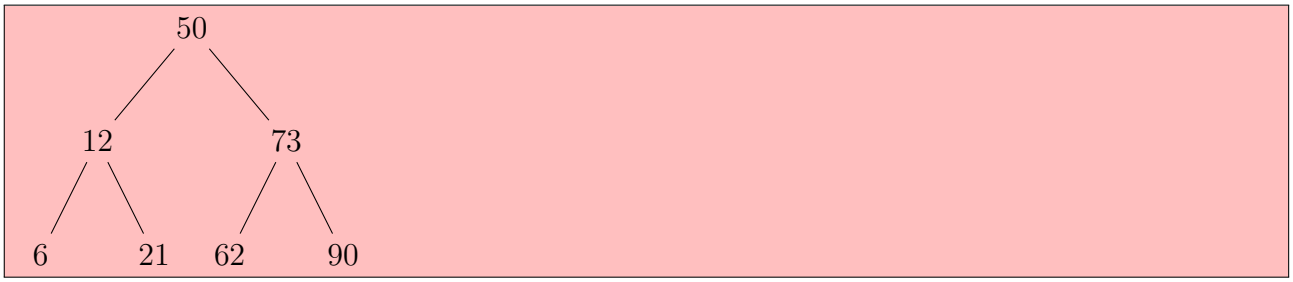
??

(3 pts) **Question 9: AVL Rotations**

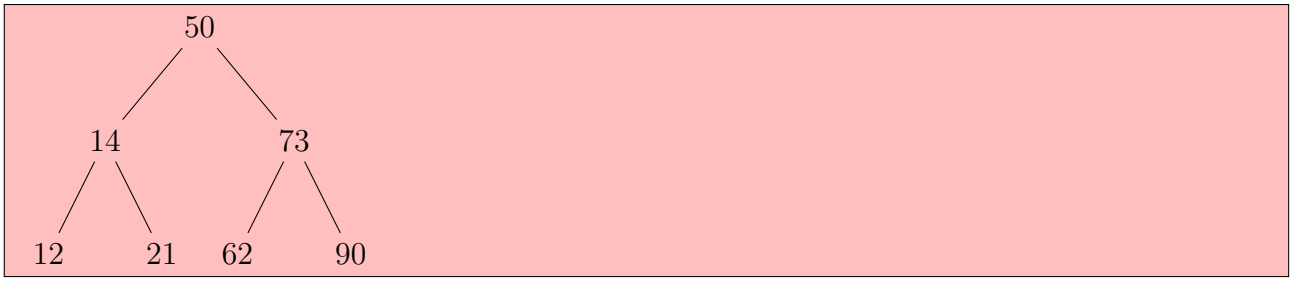


Draw out the tree resulting from applying the following insertions to the above tree, each tree should only be a single element away from the one above (don't include part (a)'s insertion in part (b)'s tree).

1. Insert element "6"



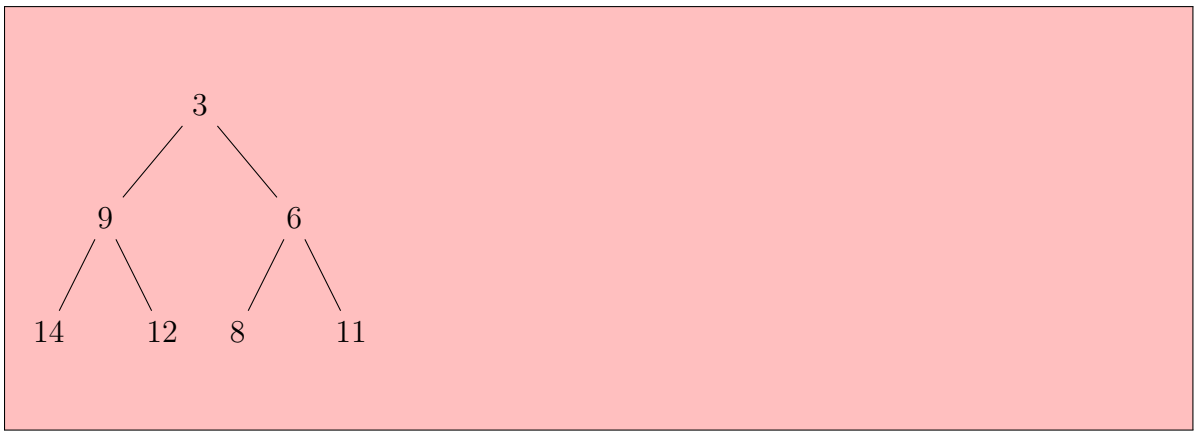
2. Insert element "14"



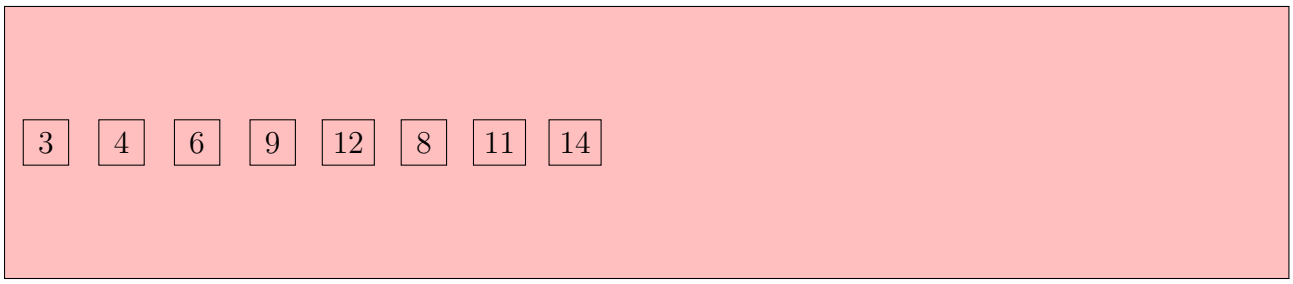
3. Insert element "55"

Section 5: Heaps

(2 pts) **Question 10: Array Representation of Heaps**
Draw a tree-like representation of this heap, currently represented as an array with the root at index 0: $[3, 9, 6, 14, 12, 8, 11]$



(2 pts) **Question 11: Modifying Heaps as Arrays**
Write the heap, as an array, that would result from inserting the element "4" to the heap above:



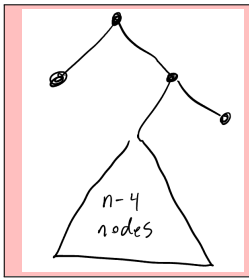
Section 6: Algorithms

(8 pts) **Question 12: Big Difference**
Give an algorithm that, given an ordered dictionary implemented with a binary search tree, returns the maximum difference of any two keys (assume keys are integers) present in the dictionary.

Traverse to the furthest left leaf to find the min, and traverse to the furthest right leaf to find the max. Return the max - min.

1. What is the best case runtime of your algorithm? $O($ 1 $)$

2. Sketch a tree for which your algorithm achieves your best case runtime.



3. What is the worst case runtime of your algorithm? $O($ n $)$

Section 7: Extra Credit

(1 pt) **Question 13: Fun!**

Name one of your instructor's hobbies. ...

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

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

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

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

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

