CSE 332 : 23Wi Midterm

Name:	**UW Email:** @uw.edu

Instructions

- The allotted time is **60** minutes. Please do not turn the page until staff says to do so.
- This is a closed-book and closed-notes exam. You are NOT permitted to access electronic devices including calculators.
- Read the directions carefully, especially for problems that require you to show work or provide an explanation.
- We can only give partial credit for work that you've written down.
- Unless otherwise noted, every time we ask for an O, Ω, or Θ bound, it must be simplified and tight.
- For answers that involve bubbling ◯ or ☐, make sure to fill in the shape completely: ● or ■.
- If you run out of room on a page, indicate where the answer continues. Try to avoid writing on the very edges of the pages: we scan your exams and edges often get cropped off.
- A formula sheet has been included at the end of the exam.

Advice

- If you feel like you're stuck on a problem, you may want to skip it and come back at the end if you have time.
- Look at the question titles on the cover page to see if you want to start somewhere other than problem 1.
- Relax and take a few deep breaths. You've got this! :-)

Question	Max points
1. Grab Bag	14
2. Code Analysis	16
3. $\mathcal{O}$, Ω, and Θ, oh, my!	12
4. Write a recurrence	9
5. Solve a recurrence	10
6. AVL Trees	10
7. Heaps	12
8. B-Trees	12
Total	**95**

1. Grab Bag [14 points]

For questions asking you about runtime, give a **simplified, tight** Big-$\mathcal{O}$ bound. This means that, for example, $\mathcal{O}(5n^2 + 7n + 3)$ (not simplified) or $\mathcal{O}(2^{n!})$ (not tight enough) are unlikely to get points. Unless otherwise specified, all logs are base 2. You may also leave you answer as an unsimplified formula like $7 \cdot 10^3$ when appropriate. You do not need to show your work for these problems.

(a) **Worst-case runtime** to find the second largest value in an AVL tree with N elements.

(a) ____________________

(b) An AVL tree of height h has a **minimum** number of nodes x and an AVL tree of height $(h-2)$ has a **minimum** number of nodes y. What is the **minimum** number of nodes for an AVL tree of height $(h+1)$? Express your answer in terms of x and y. Assume height h is an integer greater than 2.

(b) ____________________

(c) Suppose we have 1000 key-value pairs with the keys being each integer from 0 to 999 and values being arbitrary strings. What is the **minimum** number of trie nodes needed to store all 1000 key-value pairs in a trie data structure? Assume the root represents the empty string and that the integers are given without any leading zeroes.

(c) ____________________

(d) Give a simplified, tight Big-$\mathcal{O}$ bound for:
$f(N) = (\log N^2)^2 + \log(\log N^2)$

(d) ____________________

(e) We implement a `findMax` operation for a binary min heap with N elements. We start at the root node and use an iterative process. At each iterative step, we pick the child node with the bigger value, and traverse to that child. The iterative process ends when we are at a leaf node. The node we eventually traverse to will be the node with the maximum value. **True or False:** `findMax` correctly finds the maximum value of the heap.

(e)

◯ **TRUE**

◯ **FALSE**

(f) Given a B-tree of height 3, with $M = 3$ and $L = 2$, what is the **MINIMUM** possible total number of key-value pairs in the entire tree?

(f) ____________________

(g) Given a B-tree of height 3, with $M = 3$ and $L = 2$, what is the **MAXIMUM** possible total number of key-value pairs in the entire tree?

(g) ____________________

2. Code Analysis [16 points]

Describe the worst-case running time for the following pseudocode functions in Big-$\mathcal{O}$ notation in terms of the variable n. Your answer **MUST** be tight and simplified. **You do not have to show work or justify your answers for this problem.**

(a)
```
int climbingSteps(int n) {
    int steps = 0;
    for (int i = 1; i < n * n * n; i *= 3) {
        steps++;
    }
    for (int i = n; i > 1; i /= 2) {
        steps--;
    }
    return steps;
}
```

(a) ____________________

(b)
```
int collisionPoint(int n) {
    int start = n + 3;
    int end = n * (3 * n + 3) + 2;
    while (start < end) {
        start++;
        end = end - n;
    }
    return start;
}
```

(b) ____________________

(c)
```
int confusedNGL(int n) {
    int G = 1;
    int L = 332;
    if (n <= G) {
        return n + G + L;
    }
    if (n < G * L) {
        for (int i = G; i < L + n; i++) {
            G++;
        }
        return confusedNGL(n - G + L);
    }
    return confusedNGL(n / 2) + confusedNGL(G) + L;
}
```

(c) ____________________

(d)
```
int truthOrDare(int n) {
    int roulette = n;
    while (roulette != 0) {
        for (int i = 0; i < n * n; i++) {
            if (i % 2 == 1) {
                print(“I love CSE332!”);
            } else if (i % 2 == 0) {
                for (int j = i; j >= 0; j--) {
                    print(“I'd rather do all my chores...”);
                }
            }
        }
        roulette-- ;
    }
    return roulette;
}
```

(d) ____________________

3. $\mathcal{O}, \Omega,$ and Θ, oh my! [12 points]

For each of the following statements, indicate whether it is always true, sometimes true, or never true. You do **NOT** need to include an explanation. Assume that the domain and co-domain of all functions in this problem are natural numbers(1, 2, 3 ...).

(a) Let $f(n)$ be the **worst-case** runtime for inserting n elements into a binary search tree.
Then $f(n)$ is in $O(nlog(n))$.

◯ **Always True**

◯ **Sometimes True**

◯ **Never True**

(b) Let $f(n)$ be the **worst-case** runtime for `percolateUp()` when inserting a single element into a binary min heap.
Then $f(n)$ is in $\Omega(N)$.

◯ **Always True**

◯ **Sometimes True**

◯ **Never True**

(c) If $f(n)$ is $\Theta(g(n))$ and $g(n)$ is $\Omega(h(n))$, then $f(n)$ is $O(h(n))$.

◯ **Always True**

◯ **Sometimes True**

◯ **Never True**

4. Write a recurrence [9 points]

Give the recurrence relation (the exact mathematical model) of the following function. Use variables appropriately for constants (e.g. c_0, c_1, c_2, etc.) in your recurrence (you do not need to attempt to count the exact number of operations).

YOU DO NOT NEED TO SOLVE this recurrence

```
int DrWho(int n) {
    int d = 0;
    if (n < 10) {
        for (int i = 0; i <= n * n; i++) {
            d++;
        }
        print("daleks here!");
        return d;
    } else {
        int count = DrWho(n/2);
        for (int i = n; i >= 1; i  /= 2) {
            print("don't blink!");
        }
        if (DrWho(n/3) < 0) {
            for (int i = 0; i <= n; i++) {
                print("where is the TARDIS");
            }
        }
        return count + DrWho(n/3);
    }
}
```

$$T(n) = \begin{cases} \rule{8cm}{0.4pt} & \text{for } n < 10 \\ \rule{8cm}{0.4pt} & \text{for } n \geq 10 \end{cases}$$

Yay!! You do NOT **need to solve *this* recurrence...**

5. Solve a Recurrence [10 points]

Suppose the running time of an algorithm satisfies the recurrence given below. Find the closed form for $T(N)$. **You may assume N is a large power of 8.** Your answer should **not** be in Big-$\mathcal{O}$ notation. Show the exact constants and bases of logarithms in your answer (e.g. do NOT use c_0, c_1, c_2 in your answer).

Your final answer must **NOT** have any summation symbols or recursion – you may find the list of summations and logarithm identities on the last page of the exam to be useful.

You must show your work and put your final answer on the line below to receive *any* credit.

$$T(n) = \begin{cases} 1 & \text{if n = 1} \\ 8T\left(\frac{n}{8}\right) + n^2 & \text{otherwise} \end{cases}$$

Closed Form: ________________________

6. AVL Trees [10 points]

(a) (2 points) We used an array to store the binary min heap data structure. Explain in **one sentence** on why it might be a bad idea to use an array for storing an AVL tree. Your answer **MUST** include reasoning on space usage.

(b) (4 points) Suppose we are tracking the fluctuating stock price of company XYZ for the year 2023. We want to support **two** methods. The first method is `insert(int day, float price)`, which records the stock price for the given day (there can be only one stock price per day). It is guaranteed that `day` will be **strictly increasing** each time this method is called. The second method is `lookup(int day)`. The method returns the stock price of a given day in the past (it is guaranteed that the entry exists).

Explain in **2 - 3** sentences why using a sorted array might be preferable to using an AVL tree to keep track of the stock data entries. Discuss **BOTH** insert and lookup in your reasoning.

insert:

lookup:

(c) (4 points) We have an AVL tree that contains the integers $1, 2, 3, 4, 5, 6, 7$. **How many** valid AVL trees can be formed with those numbers? There's a simpler way to count than enumerating each valid AVL tree one by one. It will probably take too much time to enumerate all of them manually. Hint: think about the possible values the root can take. Give your answer as a **single number**. Showing your work is not required.

Number of valid AVL trees: ________________________

7. Heaps and FIFO Queues [12 points]

Suppose you have k FIFO queues, together holding N total integers. Each of the k FIFO queues happen to contain values sorted in **increasing** order (i.e. a call to dequeue gives you the smallest value in that FIFO queue). There is no specific ordering between the k FIFO queues (e.g. the values in FIFO queue 1 may all be less than the values in FIFO queue 2, greater than, or some mix).

Your goal is to merge the k FIFO queues into a single combined FIFO queue that is also sorted in **increasing** order (i.e. a call to dequeue on the combined FIFO queue gives you the smallest value of all N values).

For the following questions, assume a call to dequeue on any of the k FIFO queues has a worst case running time of $O(1)$. For the following questions, for heap operations, assume `insert` and `deleteMin` operations take $O(logN)$ in the worst case for a heap containing N elements. For runtime questions, give your answer in **tight** Big-$\mathcal{O}$ notation. You do **NOT** need to show your work or justify your answer for runtime.

(a) In your first attempt, you:

(i) dequeue all the elements from the first FIFO queue and insert them into a binary min heap. Then dequeue all the elements from the second FIFO queue and insert them into the same binary min heap. Repeat this process for each of the k queues until all N values have been inserted into the binary min heap.

(ii) Call `deleteMin` on the heap and enqueue the value obtained onto your combined FIFO queue. Repeat this process until all N values are added to the combined FIFO queue.

Does this solution successfully accomplish the task? (yes/no) _________ Justify your answer in $1-2$ **sentences.**

What is the Big-$\mathcal{O}$ runtime of this algorithm? $\mathcal{O}(\boxed{})$

(b) You come up with a new strategy. Assuming the k FIFO queues are equal in length, you follow this process:

(i) dequeue 1 element from the first FIFO queue and `insert` it into a heap. dequeue 1 element from the second FIFO queue and `insert` it into the same heap. Do this for each of the k FIFO queues. (e.g. A total of k dequeues and k `insert`s)

(ii) Then call `deleteMin` on the heap and enqueue the value obtained onto the combined FIFO queue. Repeat this until the heap is empty. (e.g. A total of k `deleteMin`s and k enqueues)

Repeat steps (i) and (ii) until the k FIFO queues and the heap are all empty.

Does this solution successfully accomplish the task? (yes/no) _________ Justify your answer in $1-2$ **sentences.**

What is the Big-$\mathcal{O}$ runtime of this algorithm? $\mathcal{O}(\boxed{})$

(c) Your final attempt is a variation on the previous strategy:

(i) `dequeue` 1 element from the first FIFO queue and `insert` it into a heap. `dequeue` 1 element from the second FIFO queue and `insert` it into the same heap. Do this for each of the k FIFO queues. (e.g. A total of k `dequeues` and k `inserts`)

(ii) Then call `deleteMin` on the heap and `enqueue` the value obtained onto your combined FIFO queue. If the value just removed from the heap originated from FIFO queue X, if there are remaining elements in FIFO queue X, you `dequeue` another element from FIFO queue X and insert it into the heap. If FIFO queue X is empty, then you do not need to `dequeue` and `insert` another element into the heap.

Repeat step (ii) until all FIFO queues and the heap are empty.

Does this solution successfully accomplish the task? (yes/no) _________ **Justify your answer in** $1 - 2$ **sentences.**

What is the Big-$\mathcal{O}$ runtime of this algorithm? $\mathcal{O}(\boxed{})$

8. B-Trees [12 points]

(a) (3 points) B-trees were invented by Rudolf Bayer and Edward M. McCreight while working at Boeing Research Labs. Bayer and McCreight never explained what, if anything, what the B in B-tree stands for. However, the more you think about what the B in B-trees means, the better you understand B-trees. B can be interpreted as **broad** and **balanced**, two of the properties B-trees have. In **2 - 3** sentences, explain 1) why these two properties apply to B-trees and 2) how they give B-trees an advantage compared to a regular binary search tree when storing large amounts of data. **Be sure to address both properties.**

Broad:

Balanced:

(b) (2 points) In the implementation of a B-tree, each internal and leaf node will often have an extra pointer pointing back to its parent. Explain in **1 - 2** sentences why the pointer might be useful in the context of an `insert` operation.

(c) (4 points) Suppose the block (or page) size on a memory system is 128 bytes. **Assuming each internal node and each leaf node has an extra pointer pointing back to its parent**, find the appropriate values for M and L with the following information.

Key Size = 10 bytes
Pointer Size = 8 bytes
Data Size = 4 bytes per record (does **not** include the key)

M: ______________________ L: ______________________

(d) (3 points) Give a **tight** Big-$\mathcal{O}$ runtime of the **best-case** of the `insert` operation on a B-tree (as described in lecture and in the textbook). Keep all factors of M, L, and N in your answer. Do **not** use any other variables in your answer. Be sure to give the runtime of the best-case. **Put your answer in the box below**, you **DO NOT** need to show your work.

$\mathcal{O}(\boxed{\qquad\qquad\qquad\qquad})$