Quiz 1

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on the top of every page of this quiz booklet.
- You have 120 minutes to earn a maximum of 120 points. Do not spend too much time on any one problem. Skim them all first, and attack them in the order that allows you to make the most progress.
- **You are allowed one double-sided letter-sized sheet with your own notes**. No calculators, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. Pages will be scanned and separated for grading. If you need more space, write "Continued on S1" (or S2, S3) and continue your solution on the referenced scratch page at the end of the exam.
- Do not waste time and paper rederiving facts that we have studied in lecture, recitation, or problem sets. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required. Be sure to argue that your **algorithm is correct**, and analyze the **asymptotic running time of your algorithm**. Even if your algorithm does not meet a requested bound, you **may** receive partial credit for inefficient solutions that are correct.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points
0: Information	2	2
1: Modified Priorities	2	8
2: Recurrent Recurrences	2	8
3: Tree Trials	2	8
4: Missing Balls	2	20
5: Dentucky Kerby	1	20
6: Tidying Twigs	2	30
7: CartoonCon	1	24
Total		120

Name: ______________________________

MIT Athena: ______________________________

Problem 0. [2 points] **Information** (2 parts)

(a) [1 point] Write your name and email address on the cover page.

(b) [1 point] Write your name at the top of each page.

Problem 1. [8 points] **Modified Priorities** (2 parts)

Suppose `PQ` is a Python class that implements max priority queue operations with the following running times, where k is the number of items in the priority queue at the time of the operation:

Operation	Description	Running Time
`PQ()`	return an empty priority queue	**worst-case** $\Theta(1)$
`build(A)`	build with items from `A`	**worst-case** $\Theta(\|A\|)$
`insert(x)`	add item `x` to priority queue	**amortized** $\Theta(1)$
`delete_max()`	remove item with largest key	**expected** $\Theta(\log k)$, **worst-case** $\Theta(k)$

Suppose `B` is an array containing $n \gg 10$ items. For each of the following pieces of code, state both its **worst-case** and **expected** running times in terms of n, and state whether each is amortized.

(a) [4 points]

i) Worst-case:

```python
Q = PQ()
for x in B:
    Q.insert(x)
Q.delete_max()
```

ii) Expected:

(b) [4 points]

i) Worst-case:

```python
Q = PQ()
Q.build(B)
i = 0
while 10*i < len(B):
    Q.delete_max()
    i = i + 1
```

ii) Expected:

Problem 2. [8 points] **Recurrent Recurrences** (2 parts)

Argue **upper** and **lower** bounds for the following recurrences. (More points will be awarded for tighter bounds.) Assume that $T(1) = \Theta(1)$, and that $T(a) < T(b)$ for all $a < b$.

(a) [4 points] $T(n) = 2\,T(\frac{n}{2}) + O(n^2)$

i) Upper bound:

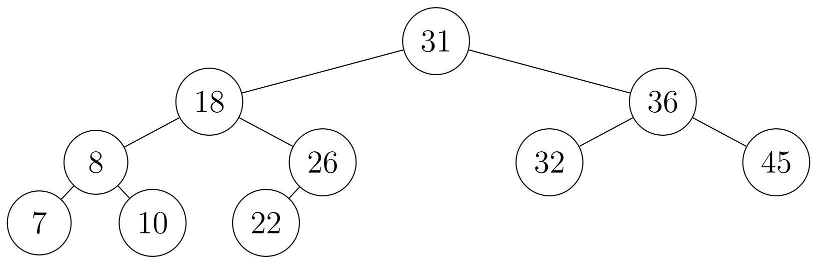
ii) Lower bound:

(b) [4 points] $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + \Theta(n)$

i) Upper bound:

ii) Lower bound:

Problem 3. [8 points] **Tree Trials**



The above binary tree T satisfies the binary search tree property. Please solve the following problems about T.

(a) [4 points] T is **packed** so could be stored in memory as a heap array. State this array.

(b) [4 points] Perform one or more rotations to T (drawing the resulting tree after each rotation) to produce a new tree T' which is (1) height-balanced, (2) maintains the binary search tree property, and (3) contains 18 at the root. (T' will not be packed)

Problem 4. [20 points] **Missing Balls** (2 parts)

Jebron Lames is a professional basketball player who has played n games in his career. After each game he plays, Jebron signs a ball from the game, labels it with its **consecutive game number** i and the **positive number of points** p_i he scored that game, and stores the ball in his trophy room. The day after playing game number n, Jebron comes home to find his balls jumbled in disarray; someone has robbed his trophy room! Help Jebron file his police report. For each of the following parts, state whether your algorithms achieve worst-case, expected, and/or amortized running times.

(a) [10 points] Given a list of the game numbers from all jumbled balls that were not stolen, describe an efficient[1] algorithm to compile a list of all game numbers from balls missing from Jebron's trophy room. (Remember to provide arguments for **correctness** and **running time** for any algorithm you describe on this quiz.)

[1]By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

(b) [10 points] Older and higher scoring balls are worth more; specifically, the value v_i of ball i is $v_i = 3^{p_i}(n-i)^2$. Given a list of game number-point pairs (i, p_i) for the missing balls sorted by game number, describe an efficient algorithm to compute the **median** value of the missing balls. Assume that: at least half the balls are missing, the number of missing balls is odd, and that the maximum number of points Jebron has scored in any game is at most $23 \lg n$.

Problem 5. [20 points] **Dentucky Kerby**

The Dentucky Kerby is one of the most selective horse races in America. To qualify, a horse must participate in multiple qualifying races during the season. The **results** of a given qualifying race is a list R containing the names of the $|R|$ horses that participated in that race in the rank order in which they finished[2]. At the end of the season, the k horses with the lowest average rank over all qualifying races will be invited to race in the Dentucky Kerby (ties may be broken arbitrarily). Note that k is chosen at the start of the season, and does not change during the season. Describe a database supporting the following operations, where n is the number of horses stored in the database at the time of the operation. For each operation, state whether your running time is worst-case, expected, and/or amortized.

`record_race(R)`	record results `R` from a new qualifying race in $O(\lvert\texttt{R}\rvert \log n)$ time
`top_k()`	return names of k horses with lowest average rank in $O(k)$ time

[2]For example, the name of the second-place horse in a qualifying race would appear as the second name in the results listing, with that horse achieving rank 2 in that qualifying race.

Problem 6. [30 points] **Tidying Twigs** (2 parts)

Gryan Briffin is a dog who has n favorite fetching sticks. Each stick has a **unique** positive integer length $\ell_i \in L$ where L is a list of all stick lengths, $(\ell_0, \ell_1, \ldots, \ell_{n-1})$. Gryan is a tidy dog and wants to store the sticks neatly in the corner of the room, using exactly two of the sticks to demarcate a rectangular storage area on the ground (in which to store the remaining $n - 2$ sticks). There will be enough room to store the sticks if Gryan can find two border sticks ℓ_a and ℓ_b such that the rectangular area $\ell_a \times \ell_b$ formed by the border is **at least as large** as the total length $T = \sum_{\ell_i \in L} \ell_i$ of all the sticks[3]. For each of the following parts, state whether your algorithms achieve worst-case, expected, and/or amortized running times.

(a) [15 points] Given L, describe an $O(n \log n)$-time algorithm to find the lengths of two sticks ℓ_a and ℓ_b such that $\ell_a \times \ell_b - T$ is minimized, subject to $\ell_a \times \ell_b \geq T$ (or return that no pair ℓ_a and ℓ_b exist such that $\ell_a \times \ell_b \geq T$).

[3]Gryan doesn't know that comparing areas to lengths may not be a good idea...

(b) [15 points] Gryan, too impatient for the previous algorithm, now just wants to check whether any pair can form a rectangle with area **equal** to the total length. Given L, describe an $O(n)$-time algorithm to determine whether there exist two sticks ℓ_a and ℓ_b such that $\ell_a \times \ell_b = T$.

Problem 7. [24 points] **CartoonCon**

Mundall Ranroe is organizing CartoonCon, a national convention for cartoonists like herself. Event scheduling for the convention is crowd-sourced. If an attendee would like to host an event during the convention, they will submit to the convention website: the event's name, and two integers denoting the starting and ending times for the event. Two time ranges **overlap** if there exists a time that is in both ranges that is not an endpoint of either range[4]. While events may be scheduled to overlap, Mundall would like to help attendees determine whether a particular time range is currently **available**, i.e., no existing event overlaps the range. Describe a database supporting the following operations, each in worst-case $O(\log n)$ time, where n is the number of events scheduled at the time of the operation.

`schedule(e, s, t)`	add event named `e` starting at time `s` and ending at time `t`
`unschedule(e)`	remove event named `e` from the schedule (if it exists)
`check_avail(s, t)`	return whether an existing event overlaps the time range from `s` to `t`

Hint: Try storing events in a modified Set data structure keyed on **starting times**.

[4]Specifically, given two time ranges (a, b) and (c, d) where $a < b$ and $c < d$, the ranges would not overlap if $b \leq c$ or $d \leq a$, but would overlap in all other cases (for example, if $c < b < d$ or $a \leq c < d \leq b$).

SCRATCH PAPER 1. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S1" on the problem statement's page.

SCRATCH PAPER 2. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S2" on the problem statement's page.

SCRATCH PAPER 3. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S3" on the problem statement's page.