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 every page of this quiz booklet.
- You have 120 minutes to earn 100 points. Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- This quiz booklet contains 7 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your quiz at the end of the exam period.
- This quiz is closed book. You may use **one** $8\frac{1}{2}'' \times 11''$ or A4 crib sheet (both sides). No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Parts	Points	Grade	Grader
1	3	10		
2	2	20		
3	1	15		
4	1	15		
5	1	20		
6	2	20		
Total		100		

Name: ______________________________ [1 point]

Friday Recitation:	Krzysztof	Alex	Zoran	Rishabh
	11 AM	**12 PM**	**3 PM**	**4 PM**

Problem 1. Asymptotic orders of growth [9 points] (3 parts)

For each of the three pairs of functions given below, rank the functions by increasing order of growth; that is, find any arrangement g_1, g_2, g_3 of the functions satisfying $g_1 = O(g_2)$, $g_2 = O(g_3)$. $\lg n$ represents logarithm of n to the base 2.

(a)

$$f_1(n) = 8\sqrt{n}, \quad f_2(n) = 25^{1000}, \quad f_3(n) = (\sqrt{3})^{\lg n}$$

(b)

$$f_1(n) = \frac{1}{100}, \quad f_2(n) = \frac{1}{n}, \quad f_3(n) = \frac{\lg n}{n}$$

(c)

$$f_1(n) = 2^{\lg^3 n}, \quad f_2(n) = n^{\lg n}, \quad f_3(n) = \lg n!$$

Problem 2. Balanced Augmented BST [20 points] (2 parts)

In this question, we use balanced binary search trees for keeping a directory of MIT students. We assume that the names of students have bounded constant length so that we can compare two different names in $O(1)$ time. Let n denotes the number of students.

(a) Say we have a binary search tree with students' last names as the keys with lexicographic dictionary ordering. Let k be the number of students whose last name starts with a given prefix x.

For example, if the tree contains 5 names $\{$ABC, ABD, ADA, ADB, ADC$\}$ there are k=2 names, ABC and ABD respectively, starting with the prefix x=AB. How can we output a list of all those students in $O(k + \log n)$ time?

(b) Give an algorithm to return the number of nodes k with names starting with a given prefix x in $O(\log n)$ time. You are allowed to augment the binary search tree nodes.

Problem 3. Employee Dictionary [15 points] (1 part)

We are interested in building a phonebook for the employees of a small company of size < 64, using a *dictionary*. Our hash table will have size $m = 64$, and the employees' last names will be used as keys.

The longest employee name is 15 characters long, so to simplify our task we convert every last name into a 15-character string by introducing *space characters* in front of the first character. For instance, the last name 'Devadas' will be padded with 8 space characters preceding the leading 'D'. Now we encode every letter of the English alphabet—together with the space character—into a unique 6-bit binary number, so that every last name can be represented uniquely by a 90-bit binary number.

Consider the following hash functions for our dictionary. Which of them is likely to perform the best? Justify your choice: For the hash functions you rejected, describe a collection of last names for which the hash function performs poorly and explain why. For the hash function of your choice, explain why it is better than the other hash functions. The input k to the following functions is the 90-bit binary number corresponding to a last name:

1. $h(k) = k \mod 64$;
2. $h(k) = ((a \cdot k) \mod 2^{18}) >> 12$, for $a = 4097$;
3. $h(k) = ((a \cdot k) \mod 2^{90}) >> 84$, for $a = 1 + 2^6 \cdot 54605$;

Note: Observe that $4097 = 1 + 2^{12}$ and $54605 = 1101010101001101_2$. The $>>$ operator is the right shift operator.

Problem 4. Asymptotic Runtime Analysis [15 points] (1 part)

Professor Devadas has an idea to assign students to recitations based on their Quiz 1 grades. He wants to split all N students into 4 recitations such that the maximum difference between the Quiz 1 grades of any two students in the same recitation is as small as possible. To figure out if this idea is reasonable, he wants to know how small the staff can make this maximum difference.

Let the number of students be N and suppose that Quiz 1 scores are all integers in the range $1 \dots M$. Professor Daskalakis suggests the following Python code, which takes the value of M and a pre-sorted list of quiz scores:

```
def assign_recitations(M, scores):
    N = len(scores)
    a = 1
    b = M
    while a < b:
        c = (a+b)/2
        groups = 1
        group_low = scores[0]
        for i in range(N):
            if scores[i] > group_low + c:
                groups = groups + 1
                group_low = scores[i]
        if groups <= 4:
            b = c
        else:
            a = c+1
    return a
```

What is the asymptotic running time of this algorithm? Express your answer as a function of N and/or M using Θ-notation.

Problem 5. Airplane Scheduling [20 points] (1 part)

Logan airport management asks you to compute the airport "load" during one day, which is the maximum number of planes that are on the ground at the same time. You are provided the data for n airplanes' arrival and departure times. To simplify computation, the day is divided into m time segments and only these segments are recorded. For the plane i, $start[i]$ denotes the time segment of arrival and $end[i]$ denotes the time segment of departure; $1 \leq start[i] \leq end[i] \leq m$. A plane is considered to be on the ground during arrival and departure time segments, as well as all segments in between. For example, if $n = 5$, $m = 10$ and planes' $(arrival, departure)$ pairs are $\{(5, 7), (1, 3), (8, 10), (2, 5), (4, 9)\}$, then the airport load is 3 (during time segment 5 planes 1, 4, and 5 are on the ground).

Give an $O(m + n)$ time algorithm to find the airport load. If you provide an algorithm with worse time complexity you will be given partial credit.

Problem 6. Multiplying two N-bit Numbers [20 points] (2 parts)

Let X and Y be two n-bit numbers in base B and we would like to compute their product $Z = XY$. A naive divide-and-conquer technique can work as follows:
Divide X into two $(n/2)$-bit numbers X_1 and X_0 and similarily Y into Y_1 and Y_0. Let $b = B^{n/2}$.

$$\begin{aligned} X &= X_1 b + X_0 \\ Y &= Y_1 b + Y_0 \\ Z &= (X_1 b + X_0)(Y_1 b + Y_0) \\ Z &= Z_2 b^2 + Z_1 b + Z_0 \end{aligned}$$

where $Z_2 = X_1Y_1, Z_1 = X_1Y_0 + X_0Y_1$ and $Z_0 = X_0Y_0$ can be obtained recursively. Assume addition of two n-bit numbers takes $\Theta(n)$ time. Also assume that multiplying an n-bit number by B^m also takes $\Theta(n + m)$ time.

(a) Prove that the above naive divide-and-conquer algorithm runs in $\Theta(n^2)$ time.

(b) It turns out that we only need 3 multiplications to compute Z_2,Z_1 and Z_0 instead of 4, required in the previous case, at the cost of some extra additions as follows:
$Z_2 = X_1Y_1$, $Z_0 = X_0Y_0$ and $Z_1 = (X_1 + X_0)(Y_1 + Y_0) - Z_2 - Z_0$
What is the time complexity of this algorithm in terms of n?

SCRATCH PAPER

SCRATCH PAPER

SCRATCH PAPER