CSE 373 A 19 Sp: Midterm May 3rd, 2019

Name_____

Student ID#_____

INSTRUCTIONS

- This test is closed book, closed calculators, closed electronics.
- You are permitted a single sheet of 8.5" x 11" notes.
- **Do not start the exam until told to do so**. You will receive a 10-point deduction if you work before the exam starts or keep working after the instructor calls for papers.
- Please write your answers neatly in the provided space.
- Please note we will only be scanning the front of each page, so please **do not put any work you wish to be graded on the back of a page** and **leave room in the margins.**
- If you have a question, please raise your hand to ask the course staff for clarification.
- Some mathematical definitions have been provided for you separate from this packet, you do not need to turn this page in, so please don't put any answers on this sheet.

ADVICE

- Write down assumptions, thoughts and intermediate steps so you can get partial credit. Clearly circle your final answer.
- The questions are not necessarily in order of difficulty, skip around.
- **Pay attention to your time and spend it wisely**

Order	Topic
1	Hashing
2	Trees
3	Closed Form of Recurrences
4	Big O Definitions
5	Code Modeling
6	Design Decisions
7	Extra Credit

1. HASHING

(A) Imagine you are given the following set of values to insert into a hash table. Assume this hash table:

- Hashes values using the function $\boldsymbol{h(x) = x}$
- Stores each value at $\boldsymbol{index = h(x)\ \%\ table\_size}$
- Uses **separate chaining** to resolve collisions, adding new values to the **back** of the list
- Has an initial internal capacity of 5
- **Triggers a resizing** of the internal array for the next value **after** reaching λ = 1

Draw the internal state of the hash table as you add the values in the given order. One of these values will trigger a resizing of the array, at this point be sure to transfer the values over so the larger array will represent the final state after adding all values.

values to add: **5, 7, 8, 4, 2, 10, 12, 25, 18**

Figure 1 – Internal array of hash table before resizing

0	1	2	3	4
5		**7 -> 2**	**8**	**4**

Figure 2 – Internal array of hash table after resizing

0	1	2	3	4	5	6	7	8	9
10		**2 -> 12**		**4**	**5 ->25**		**7**	**8 -> 18**	

(B) Imagine a hash table has an initial capacity of 10. During each resizing, the capacity, or *table_size*, of the internal array is doubled. Integer values are inserted into the table using the hash function $h(x) = 5x$. Then each value will be stored at $index = h(x)\ \%\ table\_size$

(i) Why is this design suboptimal? Explain briefly.

Hash function turns all values into multiples of 5 which will cause lots of collisions because the table size will always be a multiple of 10. This means that we will only use indices 0 and 5 initially, and resizing by 2 doesn't fix the problem (only 1/5 of our available space is used).

(ii) Describe a change that could improve this design and briefly explain how it would impact the runtime.

Change the initial table size OR the hash function so that they are not creating multiples of the same value, OR change the resizing procedure to resize to bigger prime numbers, so you could either change the table size to something initially prime like 11 instead of 10 or change the hash function to be something that is not creating easy multiples of table size like 3x or change the resizing strategy to be something more particular that lands on prime numbers for the capacity or numbers that have fewer factors as the capacity.

2. TREES

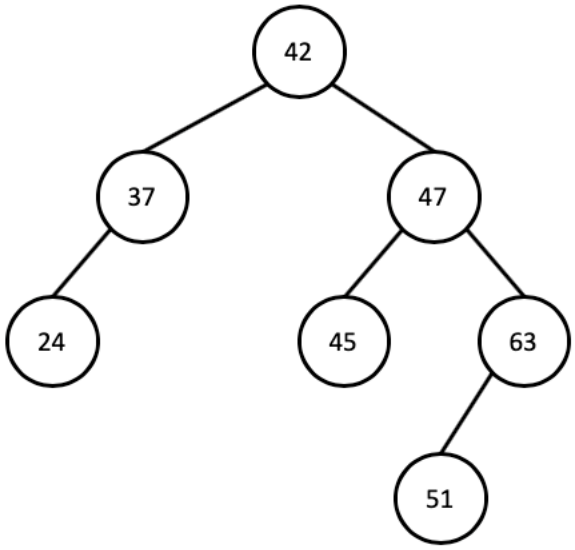

Figure 3 – Initial state of AVL Tree

(A) Imagine the value 55 is inserted into the AVL tree shown in Figure 1.

(i) What node becomes 55's parent after insertion and before any rotations?

51

(ii) Which node(s) become imbalanced due to the insertion of 55? This should include all nodes that would fail the AVL balance requirement.

63, 47, 42

(iii) Redraw the tree after any rotations triggered by the insertion of 55. Show your work to receive partial credit.

(B) Assume an AVL tree has n nodes:

(i) What is the maximum number of rotations that could be triggered by a single insertion?

2

(ii) What is the tight big-O for the worst-case runtime of locating where to place a new value in the tree?

O(logn)

(iii) What is the tight big-O for the worst-case runtime of the rotations triggered by inserting into an AVL tree?

O(1)

(C) Imagine you are given the following set of values to insert into a Binary Search Tree:

values to insert: 'a', 'b', 'c', 'd', 'e', 'f', 'g'

(i) In what order would you insert these values that would result in the **worst-case** runtime when traversing the tree after creation?

Value	'a'	'b'	'c'	'd'	'e'	'f'	'g'
Order of insertion	**1**	**2**	**3**	**4**	**5**	**6**	**7**

(ii) In what order would you insert these values that would result in the **best-case** runtime when traversing the tree after creation?

Value	'a'	'b'	'c'	'd'	'e'	'f'	'g'
Order of insertion	**4**	**2**	**5**	**1**	**6**	**3**	**7**

One example ordering is shown above for cii), but several exist as long as the BST's height is as small as possible.

3. CLOSED FORM OF A RECURRENCE

Consider the following recurrence:

$$T(n) = \begin{cases} 4 \; when \; n = 1 \\ 8T\left(\frac{n}{2}\right) + n^2 \; otherwise \end{cases}$$

(A) Find an exact closed form of this recurrence using the **tree method**.
As in class, consider the root level $i = 0$. This means at $i = 0$ the input is n and the number of nodes is 1.

(i)	What is the total number of nodes at level *i*?	$\mathbf{8^i}$
(ii)	What is the size of the **input** to each node at level *i*?	$\mathbf{\left(\frac{n}{2^i}\right)}$
(iii)	What is the total **work** done across the *i*-th *recursive* level?	$\mathbf{8^i \left(\frac{n}{2^i}\right)^2}$
(iv)	What is the value of *i* on the last level of the tree (the level of the leaf nodes)?	$\mathbf{\sum_{i=0}^{\log_2 n - 1} 8^i \left(\frac{n}{2^i}\right)^2}$
(v)	What is a mathematical expression in terms of *i* for the total work done across all the *recursive* levels?	$\mathbf{\log_2 n}$
(vi)	How many leaf nodes are on the last level of the tree?	$\mathbf{8^{\log_2 n} = n^3}$
(vii)	What is the total **work** done across the last level of the tree?	$\mathbf{4n^3}$
(viii)	What is an expression in terms of *i* for the total **work** done by *T(n)?*	$\mathbf{\sum_{i=0}^{\log_2 n - 1} 8^i \left(\frac{n}{2^i}\right)^2 + 4n^3}$

(ix) Give the closed form of the total work done by *T(n)*. You must show your work to receive credit.

$$T(n) = \sum_{i=0}^{\log_2 n-1} 8^i \left(\frac{n}{2^i}\right)^2 + 4n^3 = n^2 \left(\sum_{i=0}^{\log_2 n-1} \frac{8^i}{4^i}\right) + 4n^3 = n^2 \left(\sum_{i=0}^{\log_2 n-1} 2^i\right) + 4n^3$$

$$= n^2 \frac{\left(2^{\log_2 n} - 1\right)}{2-1} + 4n^3 = n^2(n-1) + 4n^3 = n^3 - n^2 + 4n^3 = 5n^3 - n^2$$

(x) What is the simplified tight big-O bound for *T(n)*? _____O(n^3)_____

(B) Use Master Theorem to validate your answer above. Write out the appropriate inequality and the resulting Θ to show how you applied the theorem.

$$\log_b a = \log_2 8 = 3$$
$$c = 2$$
$$3 > 2 \rightarrow \theta(n^3)$$

4. BIG-O DEFINITIONS

For each entry below give the simplified tight big-O bound and then answer the corresponding true or false question about that function.

Function	Simplified Tight Big-O	True or False? (Circle)	
$a(n) = 4\log_2 n$	$O(\log n)$	$a(n)$ *is in* $O(n)$	T
$b(n) = 4n + 3$	$O(n)$	$b(n)$ *is in* $\Omega(n \log n)$	F
$c(n) = n^2 - 3n$	$O(n^2)$	$c(n)$ *is in* $O(n^2)$	T
$d(n) = n^3 + 3^n$	$O(3^n)$	$d(n)$ *is in* $\Omega(n^2)$	T
$e(n) = 10C_1 + C_2$	$O(1)$	$e(n)$ *is in* $O(n)$	T

$f(n) = \frac{n}{2} + \log_2(2n)$	**$O(n)$**	$f(n)\ is\ in\ \Omega(\log n)$	**T**

5. CODE MODELING

(A) Consider the following code for the `sortGrades` method:

```
1  public DoubleLinkedList<Integer> sortGrades(DoubleLinkedList<Integer> input){
2     for (int i = 0; i < input.size(); i++) {
3        int newLoc = i;
4        int itemToSort = input.delete(i);
5        while (newLoc > 0 && input.get(newLoc-1) > itemToSort) {
6           newLoc--;
7        }
8        input.insert(newLoc, itemToSort);
9     }
10    return input;
11 }
```

Answer the following questions about the runtime of the `sortGrades` method. Consider the parameter `input` to contain "n" `ints`.

Note: Assume that when scanning the list for a given index, `DoubleLinkedList`'s `delete`, `get` and `insert` methods use an optimized search that will begin either at the `front` or `back` depending on which is closer to the index passed in.

(i)	In terms of `n`, what is the maximum number of nodes that will be visited by the `delete` method on line 4 during any single iteration of the outer for loop?	**n/2 (or n/2 + 1 for odd length)**
(ii)	In terms of `n`, what is the maximum number of iterations of the `while` loop between lines 5 and 7 when `i = input.size()-1`	**n-1**
(iii)	In terms of `n`, what is the maximum number of iterations of the `for` loop between lines 2 and 9?	**n**
(iv)	What is the simplified tight big-O bound for the worst-case runtime of `sortGrades`?	**O(n^3)**

(B) Consider the following code for the `assignGrades` method:

```
1  public static void assignGrades(DoubleLinkedList<Integer> input) {
2     ArrayDictionary<Double, DoubleLinkedList<Integer>> grades =
                  new ArrayDictionary<Double, DoubleLinkedList<Integer>>();
3     Iterator<Integer> itr = input.iterator();
4     for (double gpa = 0.0; gpa < 4.0; gpa += 0.5) {
5        DoubleLinkedList<Integer> scores = new DoubleLinkedList<Integer>();
6        for (int i = 0; i < input.size()/10; i++) {
7           scores.add(itr.next());
8        }
9        grades.put(gpa, scores);
10    }
11    DoubleLinkedList<Integer> topScores = new DoubleLinkedList<Integer>();
12    while (itr.hasNext()) {
13       topScores.add(itr.next());
14    }
15    grades.put(4.0, topScores);
16 }
```

Answer the following questions about the runtime of the `assignGrades` method. Consider the parameter `input` to contain "`n`" `ints`.

(i)	In terms of `n`, what is the maximum number of iterations of the `for` loop between lines 6 and 8 during any single iteration of the `for` loop between lines 4 and 10?	**n/10**
(ii)	What is the simplified tight big-O bound for the worst-case runtime of the `add` method on line 7?	**O(1)**
(iii)	What is the simplified tight big-O bound for the worst-case runtime of the `put` method on line 9 during the very **last** iteration of the `for` loop between lines 4 and 10?	**O(1)**
(iv)	In terms of `n`, what is the maximum number of iterations of the `while` loop between lines 12 and 14?	**2n/10**
(v)	What is the simplified tight big-O bound for the worst-case runtime of `assignGrades`?	**O(n)**

6. DESIGN DECISIONS

For each of the following scenarios described below select one of the following ADTs that would best suit the situation and **briefly** justify your choices.

Note: each ADT should be used exactly once.

ADTs: **List, Stack, Queue, Dictionary**

(A) You are writing a portion of a program to manage the NFL draft. In the draft teams rotate through 7 rounds of picking in which the team with the worst record for the previous year picks first, followed then by the second and so on until the team that won the super bowl picks last. The order of picking is the same in each round. Which ADT would you choose to manage the order of the teams as they rotate through their turn each round?

Queue, because maintain order and once you pull one team out the front you can put them right back in the queue for the next round

(B) You are writing a program to manage the arrangement of songs at a music festival. Which ADT would you choose to store the order of songs as you build out the show? The process of designing the show will be iterative and you will need the ability to move, add and remove songs before you get to your final design.

List so you can maintain and manipulate order

(C) You are writing a program to manage unread emails in an inbox. When a new mail arrives, it should appear at the very top of the inbox and as the mails are read, they should be removed from the unread view. Which ADT would you choose to manage the ordering of the mails?

Stack to maintain the most recent mail at the top pushing older mails closer to the bottom

(D) You are writing a program to return search results for an online store. When a user types in a query you should return all the products that include the search term in their title. Which ADT would you choose to manage the products and their titles?

Dictionary that maps product titles to the actual product

7. EXTRA CREDIT (1 POINT)

Draw an artistic representation of your current emotional state as a data structure

(extra space provided)