University of Washington

CSE 373 Winter 2020 Midterm

Full Name:	Staff Solutions
UWNetID (not student number):	
Name of person to your Left | Right	

All work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CSE373 who haven't taken it yet. Violation of these terms could result in a failing grade. **(please sign)**

Staff Solutions

Do not turn the page until 3:30.

Instructions

- This exam contains 7 pages, including this cover page. Show scratch work for partial credit, but put your final answers in the boxes provided.
- The exam is closed book (no laptops, tablets, wearable devices, or calculators). You are allowed one page (US letter, single-sided) of *handwritten* notes.
- Please silence and put away all noise-making devices, such as cell phones.
- Fill checkboxes completely. ■ is good. ☑ is not.
- Short-answer questions rarely require more than 1 sentence, and *never* require more than 2 sentences.
- You have 50 minutes to complete this exam.

Advice

- Read questions carefully before starting. Skip questions that are taking a long time.
- Read *all* questions first and start where you feel the most confident.
- Take a deep breath. You can do this.

Question	1	2	3	4	5	6	7	Total
Possible Points	12	20	21	21	18	12	1	105

UWNetID: _____

Question 1:

Which ADT would be best for the following scenarios? For each subquestion, you may select only one ADT.

(i) [3 pts] **Map Routing.** You need to find the shortest path between a source and destination for a mapping application. You've implemented an algorithm which stores a set of possible route segments ranked by how "good" that segment is, and you need to find the "best" available segment quickly.

☐ List ☐ Stack ☐ Queue ☐ Deque ☐ Set ☐ Map ■ Priority Queue

(ii) [3 pts] **Personal Playlist**. You are implementing a playlist for your music collection. You want to store the songs in your favorite order, and to add / remove songs from any point in the playlist.

■ List ☐ Stack ☐ Queue ☐ Deque ☐ Set ☐ Map ☐ Priority Queue

(iii) [3 pts] **Shared Playlist**. Your personal playlist is a success! You are now implementing a multi-user shared playlist to use at parties. In the interest of fairness (because everybody has different opinions on what constitutes "good music"), songs can only be added to the back of the shared list and removed from the front.

☐ List ☐ Stack ■ Queue ☐ Deque ☐ Set ☐ Map ☐ Priority Queue

(iv) [3 pts] **Browser history**. Whenever a user visits a new URL, the old URL is added to the browser's history; whenever the user presses the "back button", the most recent URL is removed from the history and loaded into the browser. When the history becomes too long, URLs are removed starting from the oldest entries.

☐ List ☐ Stack ☐ Queue ■ Deque ☐ Set ☐ Map ☐ Priority Queue

UWNetID: _____

Question 2:

LLRB Trees and B-Trees are significantly more complicated than an unbalanced BST. And yet, all 3 data structures implement the same ADTs; they are designed to find, add, or remove a key quickly.

(i) [4 pts] What functionality or guarantee do LLRB Trees and B-Trees offer that justifies the added complexity?

The height of LLRB Trees and B-Trees are guaranteed to be Θ(log N). BSTs do not offer this guarantee.

(ii) [4 pts] Describe the type of input which demonstrates how an LLRB Tree differs from an unbalanced BST.

Sorted input to a BST would create a BST that resembled a linked list. An LLRB would still have a height Θ(log N), however.

(iii) [12 pts] Indicate the truth of the following statements, and provide justification for your answer.

		Justification
A B-Tree is a type of Binary Search Tree	☐ True ■ False	A B-Tree can have more than 2 children if L > 2.
An LLRB Tree is a type of Binary Search Tree	■ True ☐ False	LLRB Trees obey the BST invariant: for each node in the tree, all of its left descendants must be smaller than the node's value and all of its right descendants must be greater.
When they are built using randomized input, the find() operation for unbalanced BSTs, LLRB Trees, and B-Trees all have the same *overall asymptotic* bound	■ True ☐ False	We saw in lecture that a BST built with randomized input has a height of Θ(log N). Since LLRB Trees and B-Trees also have a height of Θ(log N), the overall asymptotic bound for find() in all 3 tree types is also Θ(log N).
A B-Tree with any value for L can be converted to an equivalent LLRB Tree, and vice versa *(Reminder: L the maximum number of keys per node)*	☐ True ■ False	Only B-Trees with L=3 (ie, a 2-3 Tree) have a 1:1 correspondence (a "bijection") with LLRB Trees.

Question 3:

```
public int f(int n) {
  int retval = 0;

  if (n <= 2) {
    return retval;
  }

  if (n % 3 == 0) {
    return f(n/3);
  } else if (n % 3 == 1) {
    for (int i = 0; i < n*n/2; i++) {
      retval++;
    }
  } else {
    for (int i = 0; i < n*n*n/3; i++) {
      retval++;
    }
  }
  return retval;
}
```

(i) [8 pts] Which of the following claims about the *overall asymptotic* runtime for this function are true? Fill in the boxes next to each expression that applies.

■ Ω(1)	□ Θ(1)	□ O(1)
■ Ω(log N)	□ Θ(log N)	□ O(log N)
□ Ω(N)	□ Θ(N)	□ O(N)
□ $\Omega(N^2)$	□ $\Theta(N^2)$	□ $O(N^2)$
□ $\Omega(N^3)$	□ $\Theta(N^3)$	■ $O(N^3)$

(ii) [5 pts] Given your answer in part (iii), does the *overall asymptotic* runtime have a Θ-bound?

Answer	Justification
□ Yes, overall Θ-bound exists ■ No, overall Θ-bound does not exist	There is no Ω-bound and O-bound in common, so there can be no Θ-bound.

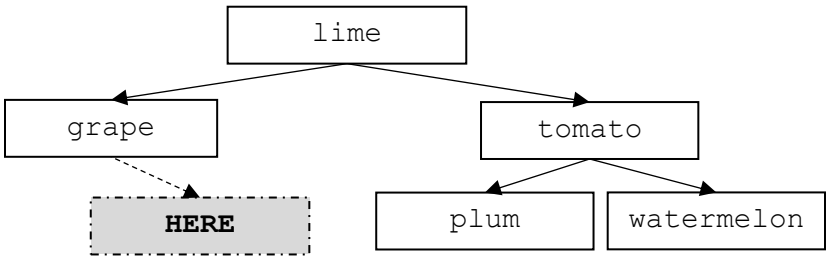
(iii) [8 pts] What set(s) of values will trigger the *best asymptotic* runtime?

Powers of 3 will trigger a log N runtime. All other values (including multiples-of-3-that-are-not-powers-of-3) will trigger a runtime that is upper bounded by $O(N^3)$

UWNetID: _____

Question 4:

Subquestions (i) and (ii) both refer to the following unbalanced Binary Search Tree



(i) [6 pts] If there were a node inserted at the "HERE" placeholder, what values could it contain? Stated in another way, what constraints must a node at "HERE" obey?

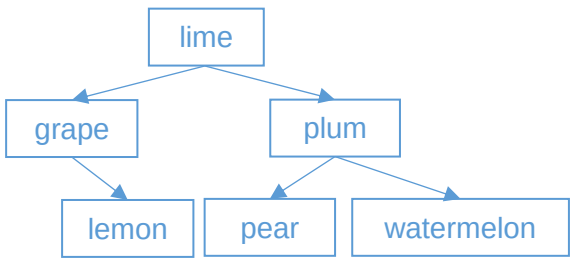
Any value stored at "HERE" needs to be greater than "grape" and less than "lime".

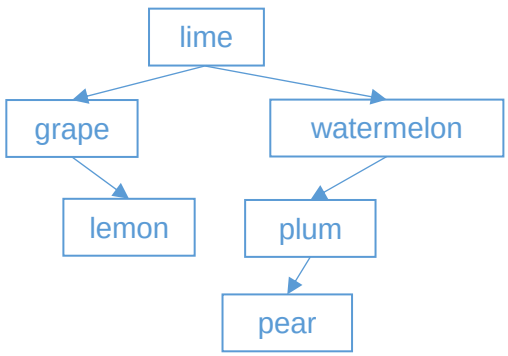
(ii) [6 pts] Assume that the "HERE" placeholder is not present in the pictured tree. If we do a pre-order traversal, what sentence is output?

lime, grape, tomato, plum, watermelon

(iii) [9 pts] Assume that the "HERE" placeholder is not present in the pictured tree. Draw the tree after the following three operations: (1) insert "pear", (2) insert "lemon", and (3) remove "tomato".

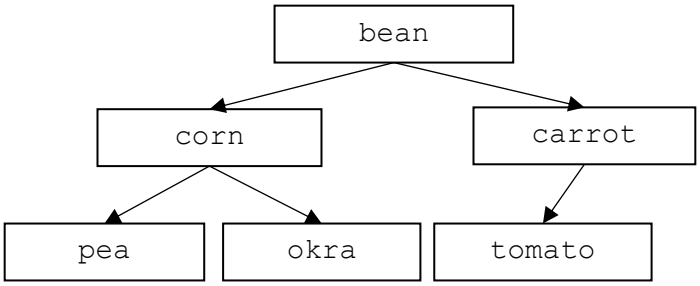
Either of the following two options were accepted; we didn't specify whether you should choose "tomato"s successor or predecessor.





Question 5:

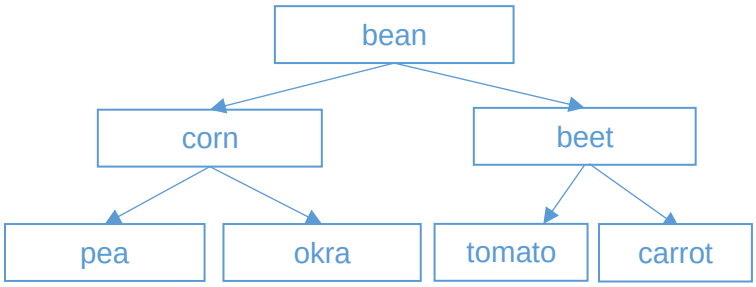
Subquestions (i), (ii) and (iii) refer to the following binary min heap:



(i) [6 pts] Recall that binary heaps can be stored as an array. Draw the array representation of the above heap.

0	1	2	3	4	5	6
(empty)	bean	corn	carrot	pea	okra	tomato

(ii) [7 pts] Draw the heap in its tree representation after adding "beet".



(iii) [5 pts] Imagine we are implementing removeMin() on the pictured heap (ie, do not include the value "beet" from part (ii)). The following operations need to be rearranged in the correct order to correctly re-establish the heap invariant.

Not all of these operations may be necessary. If you do not use an operation, mark it with an 'X'.

Operation	Step #
Replace "bean" with "tomato"	1
Swap "corn" and "pea"	X
Swap "corn" and "tomato"	X
Swap "carrot" and "tomato"	2
Swap "pea" and "tomato"	X
Swap "okra" and "tomato	X

Question 6:

Recall that a good hash function is uniform, efficient, and deterministic. For each of the following hash, indicate whether it demonstrates these characteristics; if it does not, explain why.

(i) [6 pts]

```
public int hash(String s) {
  return s.length();
}
```

Uniform?	☐ Yes	■ No. Explanation: Hash values will cluster at common word lengths; for example, hash("cat") == hash("and") == hash("rat").
Deterministic?	■ Yes	☐ No. Explanation:

(ii) [6 pts]

```
// codePointAt is a static member of Character.  Its arguments
// are a string and an index into it, and it returns an integer
// uniquely representing the character at that index.
public int hash(String s) {
  int result = 7;
  for (int i = 0; i < s.length(); i++) {
    for (int j = i; j < s.length(); j++) {
      result = result * j + Character.codePointAt(s, i);
    }
  }
  return result;
}
```

Efficient?	☐ Yes	■ No. Explanation: A reasonable runtime for a hash function is Θ (N), where N is the length of the input. This function runs in $\Theta(N^2)$. Hashing longer strings will take a disproportionately long time.
Deterministic?	■ Yes	☐ No. Explanation:

Question 7:

[1 pt; all non-empty answers receive this point] Is a tomato a fruit, or is it a vegetable? You may add your justification, sketch some tomatoes, compose a 12-part epic poem, share a favorite recipe, or anything.

¯_(ツ)_/¯

I guess idk, but apparently the Supreme Court does? https://en.wikipedia.org/wiki/Nix_v._Hedden

Congratulations, you've completed the midterm!
Have you written your UWNetID on every page?