Total Score: **0.0**

1. [/12] True or False

Determine whether each of the statements is true or false. Write "True" or "False" in the box.

For the first **two** questions, suppose $T_1(n)$ is $O(f(n))$ and $T_2(n)$ is $O(g(n))$ are positive functions.

(a) [ ] $T_1(n) - T_2(n)$ is always $O(f(n) - g(n))$

(b) [ ] $T_1(n)/T_2(n)$ is always $O(f(n)/g(n))$

(c) [ ] $(\log\log\log n)^2$ is $\omega(\log\log n)$.

(d) [ ] If a problem has a lower bound of $\Omega(n^2 \log\sqrt{n})$, then no algorithm exists that solves it in $o(n^2 \log\sqrt{n})$.

(e) [ ] Every max heap of n keys is a BST.

(f) [ ] An array sorted in increasing order is a min-heap.

(g) [ ] Suppose two algorithms A and B solve the same problem where A is

$O(n)$ and B is $\Omega(n^3)$. Then, for all inputs, A is faster than B.

(h) ☐ All compressed tries with n leaves, will have exactly $n-1$ internal nodes.

(i) ☐ In the worst case, the number of rotations required after inserting a key to an AVL tree is $O(1)$.

(j) ☐ When a key is deleted from a trie, its possible that the trie may remain the same size (i.e. no nodes are removed).

(k) ☐ Given array $A = [8342, 7564, 9342, 5394, 8934, 5555, 2843, 5394, 7342]$. CountingSort on A runs in $\Theta(n)$ time.

(l) ☐ The expected runtime for search in a skip list of n elements when all towers have height bounded by a given constant is $\Omega(n)$.

Q2abc Score: **0.0**

2. /14=2(4)+3(2) Short Answer

(a) **[2 Marks]** When quicksort is implemented by choosing the pivot randomly, the best case and worst case expected runtimes are the same. Briefly explain why this is the case.

(b) **[2 Marks]** Consider a max-heap that is implemented as an array A. Describe an algorithm for the function `deleteAtIndex` that given an array index i removes $A[i]$ from the max-heap.

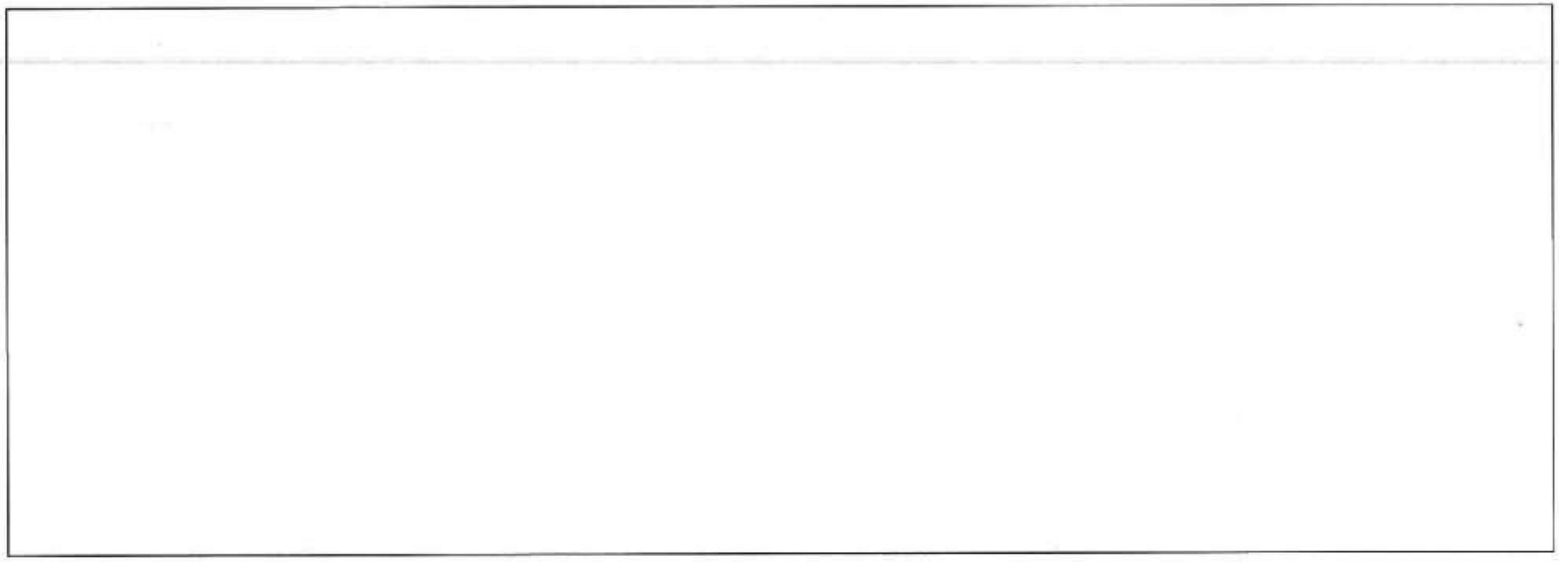
(c) **[2 Marks]** Given the following array of integers: [45981, 33333, 45979, 45368, 02531, 33001, 04598].

Show the resulting array after one round of LSD radix sort, and after two rounds of LSD radix sort.

(d) **[2 Marks]** What are the minimum and maximum number of non-empty keys, that can be stored in an uncompressed trie with n nodes. Justify your answer.

(e) **[3 Marks]** A dictionary D is stored as an unordered linked list L of size n. Give a sequence of k searches on L which is the worst case for Transpose but $\Theta(k+n)$ for Move-To-Front (MTF). Briefly justify the runtime of each heuristic.

(f) **[3 Marks]** Consider sorting an array of size 6 using a comparison based sorting method. Show how a lower bound on the number of comparisons can be derived. You do not need to simplify your expression.



Q3ab Score: **0.0**

3. /17=3(3)+4+2(2) Asymptotic Analysis

(a) **[3 Marks]** Prove from first principles that $3n^2 - n + 2$ is $\Theta(n^2)$.

(b) **[3 Marks]** Prove from first principles that $7n + 3 \in o(n \log n)$.

(c) [**3 Marks**] Analyze the pseudocode below and give a Θ-bound on the running time as a function of n. As a first step, give a closed form for j and k in terms of i.

```
mystery(n)
1.   i := 0
2.   j := 0
3.   k := 1
4.   while j <= n
5.       i := i + 1
6.       j := j + k
7.       k := k + 2
```

(d) [**4 Marks**] For each pair $f(n)$, $g(n)$, indicate in the table whether or not $f(n)$ is in o, O, Ω, or ω of $g(n)$. Use "Y" for yes, and "N" for no. Indicate all that apply.

$f(n)$	$g(n)$	O	Ω	o	ω
$n^2 + \sin(n)$	n				
$5^n + n^5 + 5^5$	5^{n^2}				
$\log(\frac{n^2}{n-1}) + 7$	$\log n$				
$\frac{2(n+\sqrt{n})}{3}$	n				

(e) **[2 Marks]** Consider the following algorithm.

```
luckydie(n)
1.  i := random outcome from rolls the die (see below)
2.  if i is even
3.      for j = 1 to n
4.          printf("*")
5.  else
6.      printf("*")
```

The die used in the function that "rolls the die" is a 6-sided die with the following numbers on each side: 1 3 3 5 5 6 (**not the typical 1 2 3 4 5 6**).
Given an input of size 12, state the expected number of times an asterisk (*) will be printed.

(f) **[2 Marks]** Consider the following algorithm.

```
luckydie2(n)
1.  i := random outcome from rolls the die (see below)
2.  if i is even
3.      for i = 1 to n
```

```
4.            printf("*")
5.    else
6.        printf("*")
7.        luckydie2(n)
```

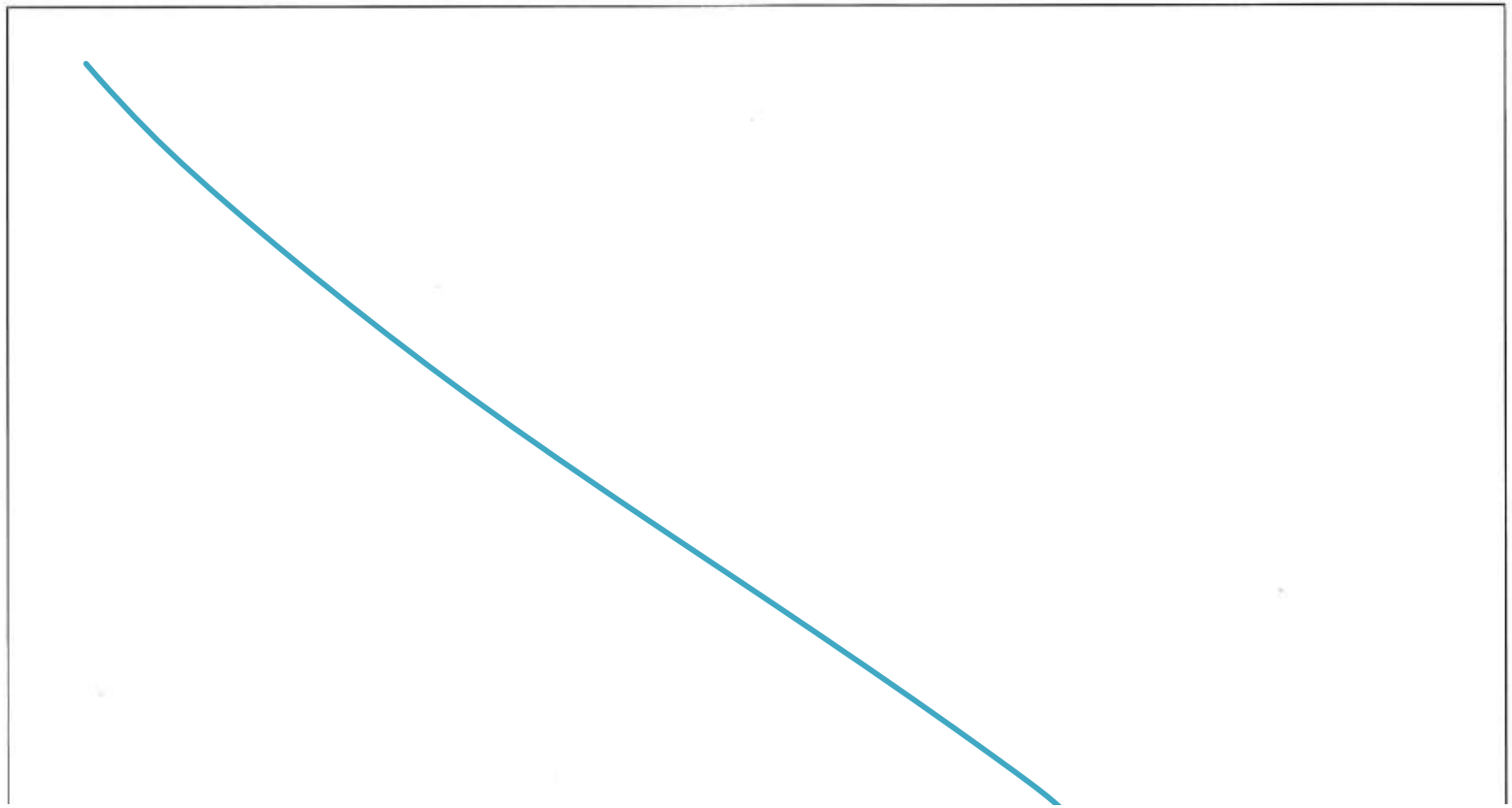
The same die from the previous part is also used. Given an input of size n, give a recurrence relation for the expected number of times an asterisk (*) will be printed and simplify $T(n)$ to be only in terms of n.

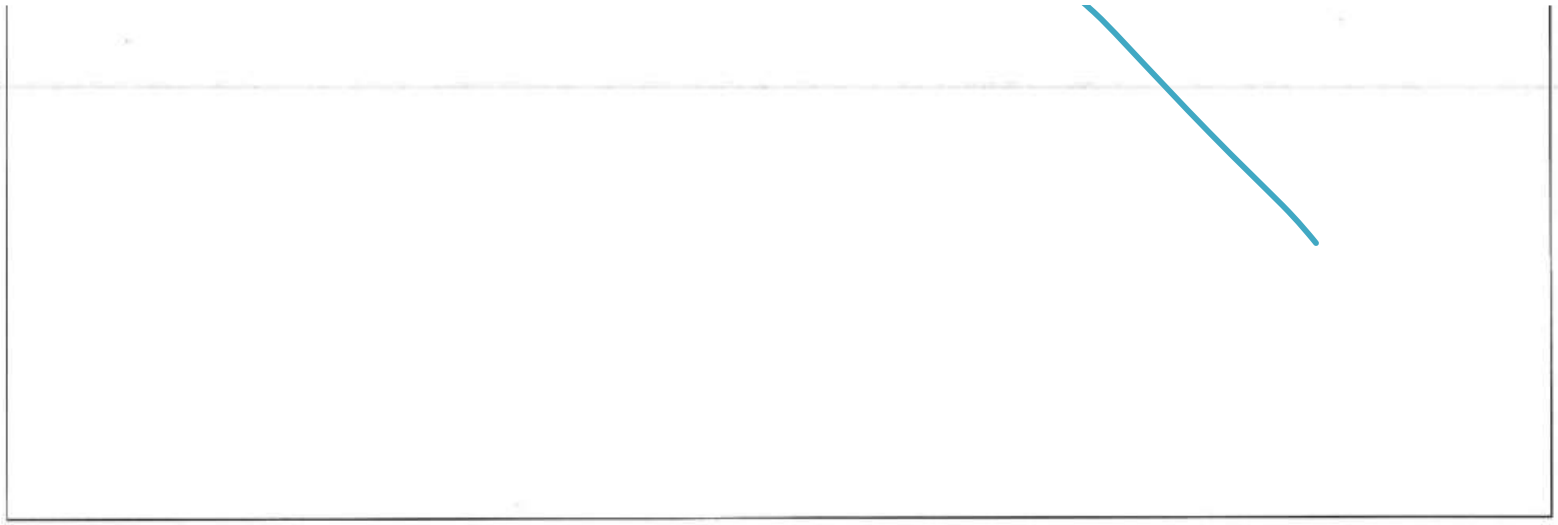
Q4 Score: **0.0**

4. /5 Algorithm Design

Given an array A of n distinct integers, suppose you want to find the $\sqrt{n}$ largest elements. However, the largest heap you may use is of size $\sqrt{n}$ but you may use as many as needed. Give a high-level description (or pseudocode) that efficiently solves this problem and briefly justify the runtime of your algorithm.

Marks will be deducted if your algorithm is a different asymptotic order from the optimal solution.



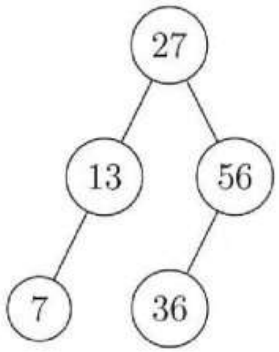


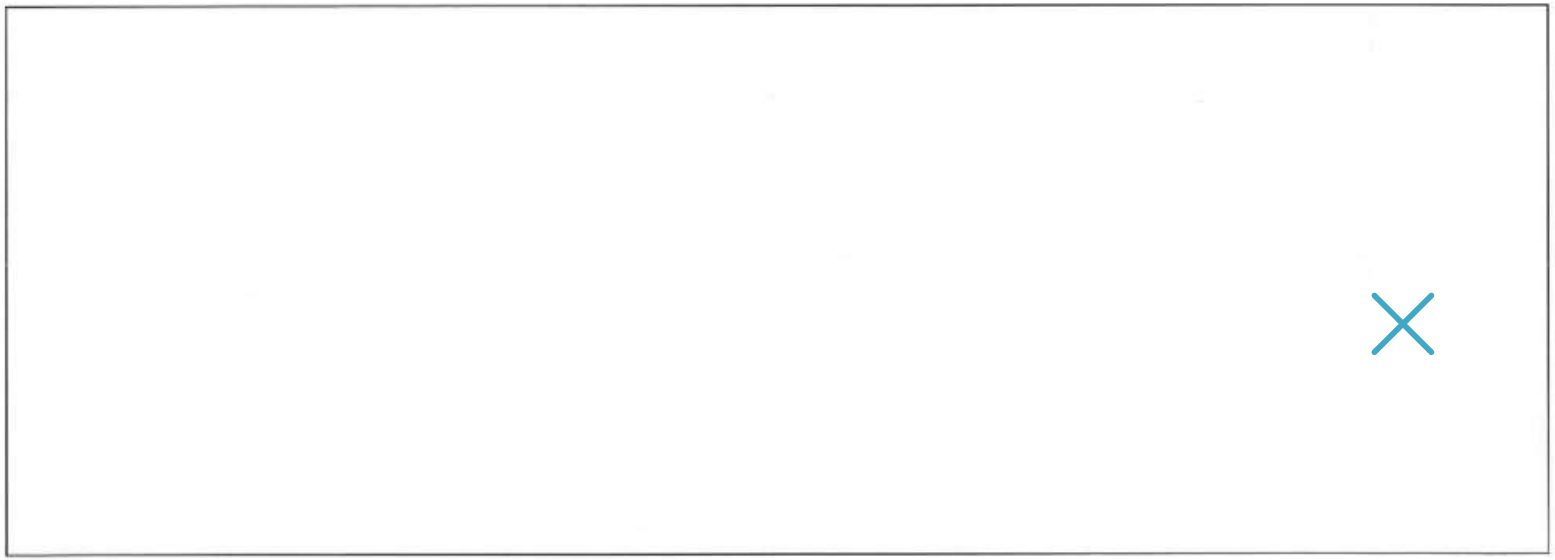
Q5ab Score: **0.0**

5. /8=2+2+4 AVL Trees

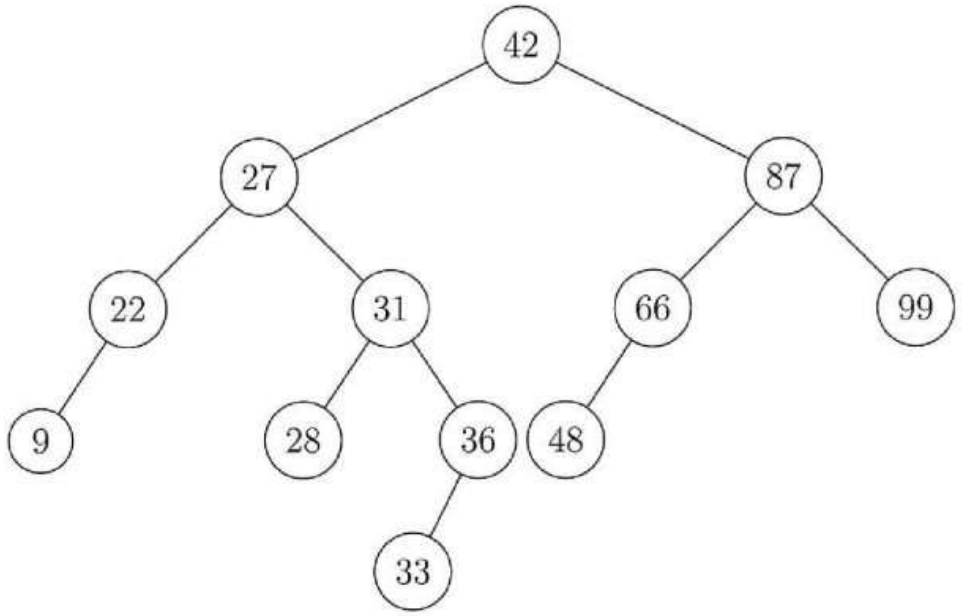
(a) **[2 Marks]** Construct an AVL tree with the following balance factors: +1, +1, 0, 0, 0, -1, -1. Each of the balance factors must be used exactly once. Provide a key along with the balance factor for each node.

(b) **[2 Marks]** Insert the balance factors in the tree below, then perform an AVL insertion of key 42. Show the resulting tree **and balance factors** after each single rotation (i.e. show both steps for a double rotation).





(c) [4 **Marks**] The following is an AVL tree. Perform an AVL deletion on 99. Show the resulting tree **and balance factors** after each single rotation (i.e. show both steps for a double rotation).



Q6a Score: **0.0**

6. /6=4+1+1 Skip Lists

(a) **[4 Marks]** Draw the skip list that results from inserting $[2, 5, 7, 13, 27, 42]$, in the order given, into an initially empty skip list.

To determine the height of a tower we will use a sequence of numbers generated randomly by a random number generator RNG that only outputs integers between 1 and 5. The height of a tower is determined as follows:

- Generate a number from RNG and **remember it**.
- **Count** the number of times RNG generates the same number from the previous step.
- **Stop** counting when RNG generates a different number. The number of repeated occurrences is the height of the tower.

Repeat this process to determine the height of each tower.
Note: The initial number from the first step is not counted towards the height and the last number (different from the one being counted) is only used to mark the end of the sequence.
For example, in the sequence: 3 3 3 4 5 5 6 the first tower has height 2 (counting 3s), the second tower has height 1 (counting 5s) and all numbers in the sequence (including 4 and 6) have been used.

Use the following sequence of numbers generated by RNG to determine tower heights:

3 3 3 4 5 5 6 2 1 3 1 1 1 1 1 3 2 3 4 4 4 5 5 5 4 3 3 2

Each number will only be used once, in order and there may be some unused numbers.

(b) [**1 Mark**] What is the probability that a key will have a tower of height at least 2? Give a brief justification.

(c) [**1 Mark**] What is the expected height of the skip list if it contains n keys? You do not need to justify your answer.

Q7 Score: **0.0**

7. /6=4+2 Tries

(a) **[4 Marks]** Construct a Patricia Trie on the following keys:

11, 000, 0010, 1110, 10000, 10110, 10010

(b) **[2 Marks]** Describe the steps required to delete key 10010 into the tree from the previous part.

Extra page A.

This page is intentionally left blank for your use. Do not remove it from your booklet. If you require more space to answer a question, you may use this page, but you must **clearly indicate** in the provided answer space that you have done so.

Extra page B

This page is intentionally left blank for your use. Do not remove it from your booklet. If you require more space to answer a question, you may use this page, but you must **clearly indicate** in the provided answer space that you have done so.

Extra Not graded

FC663BA6-ADBE-4B80-856E-AC1C264B47BE

Winter 2017 CS 240 Midterm

#150 16 of 16