Name: Sample Solution

UWNetID: ____________________

# CSE 332 Autumn 2018: Midterm Exam
**(closed book, closed notes, no calculators)**

**Instructions:** Read the directions for each question carefully before answering. We will give partial credit based on the work you **write down**, so show your work! Use only the data structures and algorithms we have discussed in class so far.

**Note**: For questions where you are drawing pictures, please circle your final answer.

**Good Luck!**

Total: 100 points. Time: 60 minutes.

| Question | Max Points | Score |
|---|---|---|
| 1 | 18 | |
| 2 | 16 | |
| 3 | 12 | |
| 4 | 8 | |
| 5 | 10 | |
| 6 | 9 | |
| 7 | 10 | |
| 8 | 8 | |
| 9 | 9 | |
| **Total** | 100 | |

**1. (18 pts) Big-Oh**

(2 pts each) For each of the operations/functions given below, indicate the tightest bound possible (in other words, giving $O(2^N)$ as the answer to every question is not likely to result in many points). Unless otherwise specified, all logs are base 2. **Your answer should be as "tight" and "simple" as possible.** For questions that ask about running time of operations, assume that the most efficient implementation is used. For array-based structures, assume that the underlying array is large enough.

You do not need to explain your answer.

a) *Finding and removing the largest item in a* ***binary search tree*** *containing N elements (worst case).* $O(N)$

b) $T(N) = 2\,T(N - 1) + 3$ $O(2^N)$

c) *Enqueue in a* (FIFO) ***queue*** *containing N elements implemented using an array as the underlying structure. (worst case)* $O(1)$

d) remove*(k) on a* ***binary min heap*** *containing N elements. Assume you have a reference to the key k that should be removed. (worst case)* $O(\log N)$

e) $f(N) = (\log N)^2 + N \log (N^2)$ $O(N \log N)$

f) Inserting the integers 1, 2, 3,.. N (in that order) into a ***binary min heap***. $O(N)$

g) $f(N) = \log \log N + \log^2 N$ $O(\log^2 N)$

h) *Finding the largest* *even* *value in an* ***AVL tree*** *containing N integers. (worst case)* $O(N)$

i) $T(N) = 2\,T(N/2) + ½\,(N)$ $O(N \log N)$

**2. (16 pts) Big-Oh and Run Time Analysis:** Describe the worst case running time of the following pseudocode functions in Big-Oh notation in terms of the variable `n`. Your answer should be as "tight" and "simple" as possible. ***Showing your work is not required.***

Runtime:

I.
```
void treat(int n, int apples) {
  for (int i = 0; i < n * n; i++) {
    if (i % 7  == 0) {
      for (int j = 0; j < i; j++) {
        apples++;
      }
    }
  }
}
```

$\boxed{O(n^4)}$

II.
```
int spider(int n) {
  if (n < 100) {
      for (int i = 0; i < n; i++) {
          print("WEB!");
      }
      return 27;
  } else if (n < 2000) {
      return spider(n / 2);
  }
  return spider(n / 2) + spider(n / 2);
}
```

$\boxed{O(n)}$

III.
```
int spooky(int n, int candy) {
  int ghost = n;
  while (ghost > 0) {
    for (int i = 0; i < n; i++) {
      candy += 4;
    }
    ghost = ghost / 2;
  }
  return candy;
}
```

$\boxed{O(n\log n)}$

IV.
```
void pumpkin(int n) {
  if (n <= 0) return;
  if (n % 2 == 0) {
    for (int i = 0; i < n; i++) {
      print("Jack O'");
    }
  } else (
      for (int i = 0; i < n * n; i++) {
        print("Lantern");
      }
  )
  pumpkin(n - 1);
}
```

$\boxed{O(n^3)}$

**3. (12 pts) Big-O, Big Ω, Big Θ**

(4 pts each) For parts (a) – (c) circle **ALL** of the items (if any) that are TRUE. You do not need to show any work or give an explanation.

a) $\log^2 N + N^2 \log N$ is:

$\Omega(N^2 \log^2 N)$ $O(N \log^2 N)$ $\Omega(N^2 \log N)$ $\Theta(N^2 \log N)$

b) $2^{(3/2) * N} + N^{3/2}$ is:

$O(N^3)$ $\Omega(2^{3*N})$ $\Theta(N^{3/2})$ $\Omega(N^{3/2})$

c) $\log(N^2) + \log\log N$ is:

$\Omega(N)$ $O(\log\log N)$ $\Theta(\log N)$ $\Omega(\log^2 N)$

**4. (8 pts) 3 Heaps**

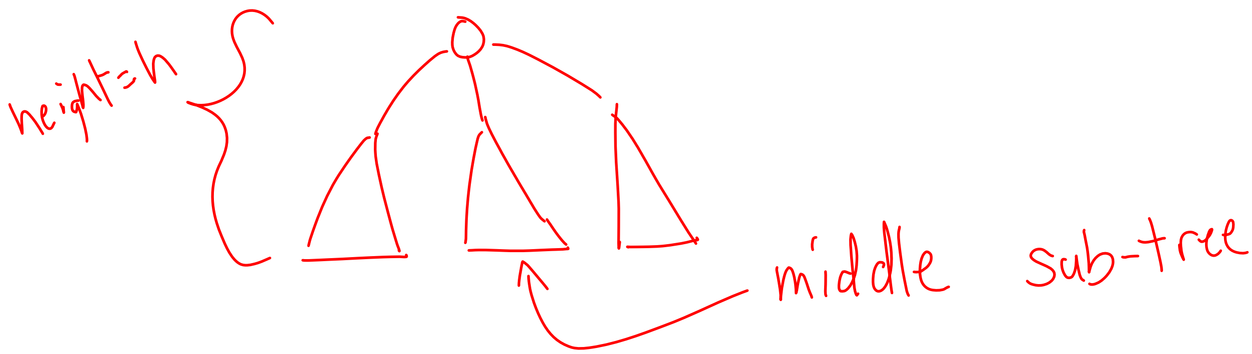
Given a 3-heap of height h, what are the minimum and maximum number of nodes in the **middle sub-tree** of the root? Give your answer in closed form (there should not be any summation symbols).

Min nodes in middle sub-tree:

$$\frac{3^{h-1}-1}{2}$$

Max nodes in middle sub-tree:

$$\frac{3^{h}-1}{2}$$




Max Nodes: (max nodes in 3-heap of height h-1)

$$\sum_{i=0}^{h-1} 3^i = \frac{3^h-1}{2}$$

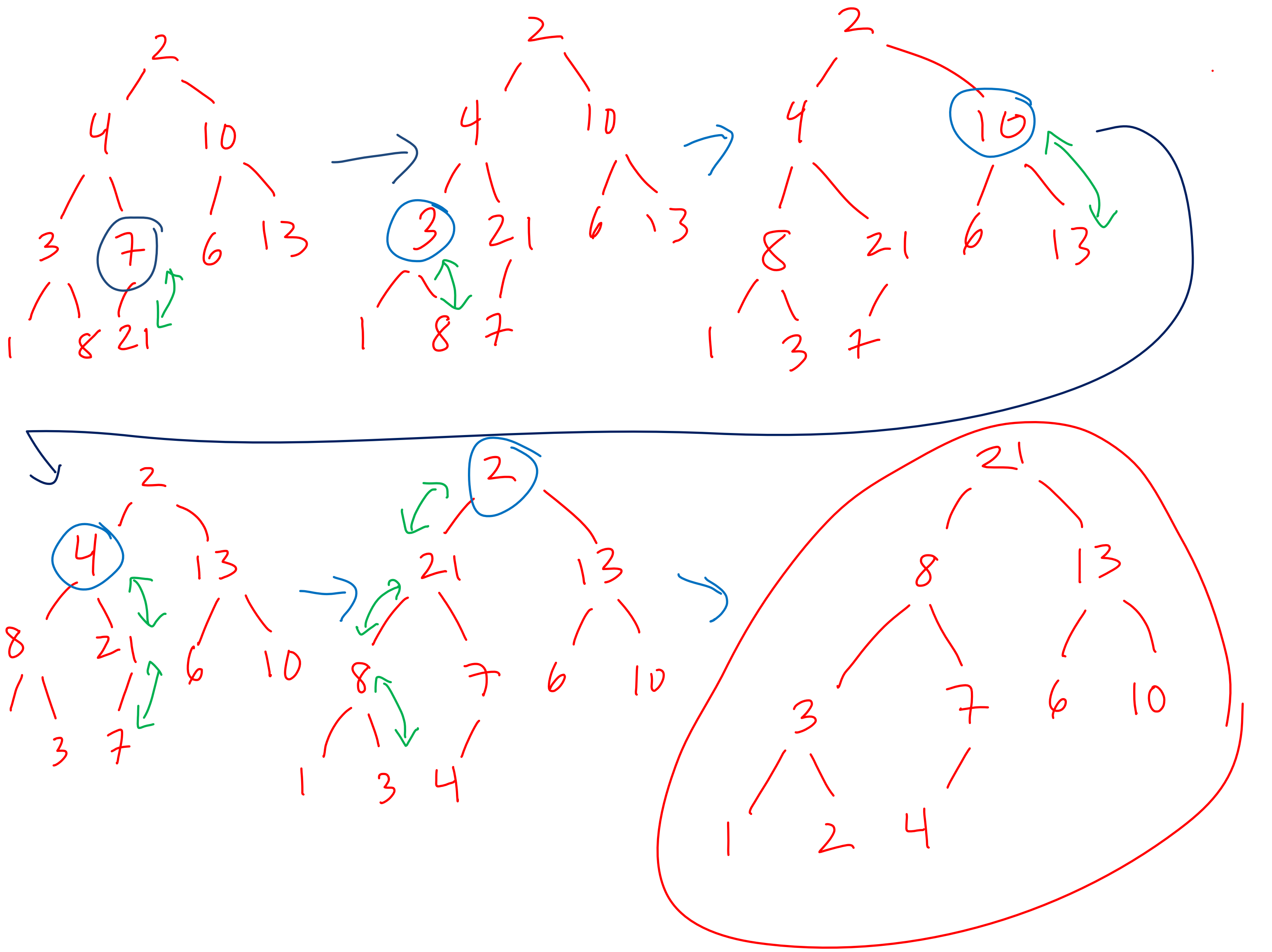
Min Nodes: (max nodes in a 3-heap of height h-2)

$$\sum_{i=0}^{h-2} 3^i = \frac{3^{h-1}-1}{2}$$

**5. (10 pts) Binary <u>Max</u> Heaps**

Use Floyd's build heap to create a **<u>Max</u>** heap out of the following array. (Hint: a binary max heap would have the largest value at the root of the tree.) **For any credit, show your <u>tree</u> one step at a time.** You do not need to show the array. THIS IS A BINARY **<u>MAX</u>** HEAP!

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 10 | 3 | 7 | 6 | 13 | 1 | 8 | 21 |



**6. (9 pts) Recurrences**

Give a base case and a recurrence for the runtime of the following function. Use variables appropriately for constants (e.g. $c_1$, $c_2$, etc.) in your recurrence (you do not need to attempt to count the exact number of operations). **<u>YOU DO NOT NEED TO SOLVE</u>** this recurrence.

```
int onion(int n) {
  if (n < 10) {
    return n * n;
  }
  else {
    for (int i = 0; i < n; i++) {
      print "Keep trie-ing!";
      print "Onions rule!"
    }
    return n * onion(n / 3) + 10 * onion(n / 3);
  }
}
```

T(n) = $c_1$ For n < 10

T(n) = $c_2 + c_3 \cdot n + 2 \cdot T\left(\frac{n}{3}\right)$ For n >= 10

**Yipee!!!! YOU DO NOT NEED TO SOLVE** *<u>this</u>* recurrence…

**7. (10 pts) Solving Recurrences**

Suppose that the running time of an algorithm satisfies the recurrence relationship:

$$T(1) = 7.$$

and

$$T(N) = T(N/3) + 5 \qquad \text{for integers } N > 1$$

Find the closed form for T(N). **You may assume that N is a power of 3.** Your answer should *not* be in Big-Oh notation – show the relevant *exact* constants and bases of logarithms in your answer (e.g. do NOT use "$c_1$, $c_2$" in your answer). You should not have any summation symbols in your answer. The list of summations on the last page of the exam may be useful. ***You must show your work to receive any credit***.

$$T(N) = T(N/3) + 5$$

$$= T(N/9) + 5 + 5$$

$$= T\left(\frac{N}{27}\right) + 5 + 5 + 5$$

$$= T\left(\frac{N}{3^K}\right) + 5 \cdot K$$

$$= T(1) + 5 \cdot \log_3 N$$

$$= 7 + 5 \cdot \log_3 N$$

when

$$\frac{N}{3^K} = 1$$

$$\log_3(3^K) = \log_3(N)$$

$$K = \log_3 N$$

**8. (8 pts) B-Trees**

Given the following parameters for a B-tree with M = 21 and L = 12:

Key Size = 4 bytes

Pointer Size = 8 bytes

Data Size = 20 bytes per record (***includes*** the key)

Assuming that M and L were chosen appropriately, **what is the likely size of a page (also known as a disk block)** on the machine where this implementation will be deployed? Give a numeric answer and **a short justification *based on two equations*** using the parameter values above.

$4 \cdot (M-1) + 8 \cdot M \le$ page size

$4 \cdot 20 + 8 \cdot 21$

$80 + 168$

At least 248 bytes needed for an interior node

Also: $20 \cdot L \le$ page size

$20 \cdot 12$

$240$

So at least 240 bytes needed for a leaf node

Page sizes are typically powers of 2. Whether we are trying to fit an interior node or a leaf node on a page, you want to be sure the node's total size does not exceed the size of a page. So the page would need to be at least 248 bytes (since that is the larger of the two node types. The next highest power of 2 is 256. So 256 bytes is the likely page size. (Also if we tried to increase M or L we would go over 256 bytes.)

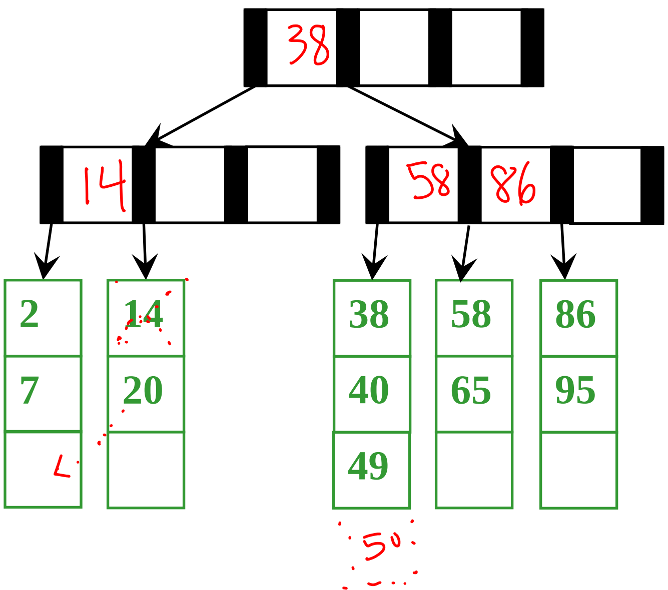
**9. (9 pts) B-trees**

a) (1 pt) In the **ORIGINAL** B-Tree shown below, **add values for the interior nodes**.

b) (4 pts) Starting with the **ORIGINAL** B-tree shown below, in upper box, draw the tree resulting after inserting the value 50 (*including values for interior nodes*). Use the method for insertion described in lecture and in the book.

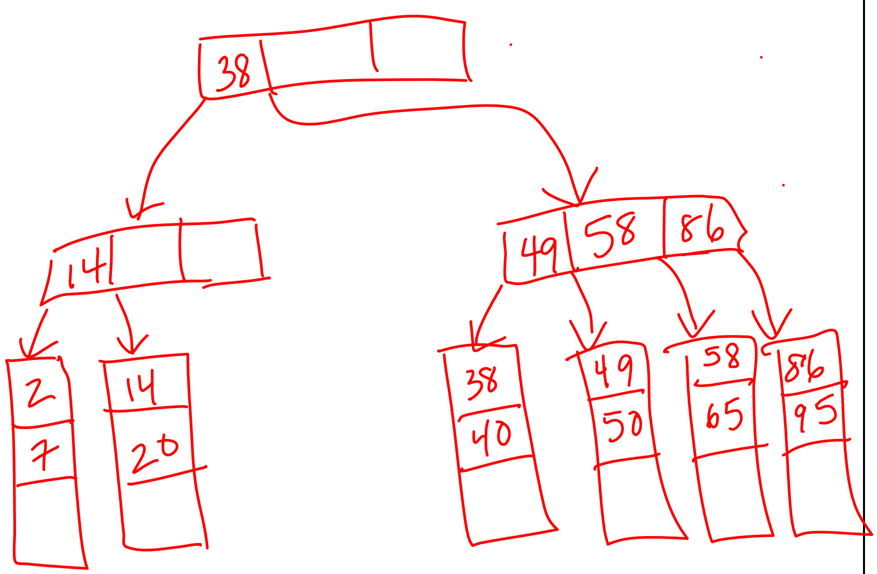
c) (4 pts) Starting with the **ORIGINAL** B-tree shown below, in the lower box, draw the tree resulting after deleting the value 14 (*including values for interior nodes*). Use the method for deletion described in lecture and in the book.

**ORIGINAL**:




After inserting 50:

- split leaf




After deleting 14:

- merge leaf
- adopt pointer