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.

<table>
<thead>
<tr>
<th>Question</th>
<th>Max Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
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). $O(N)$
  - $O(2^N)$

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

- **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)** $T(N) = 2T(N/2) + \frac{\sqrt{N}}{2}$ (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.*

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++;`  
         `}`  
      `}`  
   `}`

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);`  
   `} else {`  
      `return spider(n / 2) + spider(n / 2);`  
   `}`

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;`  

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);`  

**Runtime:**

I. \( O(n^4) \)

II. \( O(n) \)

III. \( O(n \log n) \)

IV. \( O(n^3) \)
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:
\[ \Omega (N^2 \log^2 N) \quad O (N \log^2 N) \quad \Omega (N^2 \log N) \quad \Theta (N^2 \log N) \]

b) \( 2^{(3/2)N} + N^{3/2} \) is:
\[ O (N^3) \quad \Omega (2^{3N}) \quad \Theta (N^{3/2}) \quad \Omega (N^{3/2}) \]

c) \( \log (N^2) + \log \log N \) is:
\[ \Omega (N) \quad O (\log \log N) \quad \Theta (\log N) \quad \Omega (\log^2 N) \]
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:

$$\frac{3^h - 1}{2}$$
5. **(10 pts) Binary Max Heaps**  
Use Floyd's build heap to create a **Max** 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 tree one step at a time.** You do not need to show the array. **THIS IS A BINARY MAX HEAP!**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>13</td>
<td>1</td>
<td>8</td>
<td>21</td>
</tr>
</tbody>
</table>

![Binary Max Heap Diagram]
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). **YOU DO NOT NEED TO SOLVE** this recurrence.

```c
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);
    }
}
```

\[
T(n) = \begin{cases} 
  c_1 & \text{For } n < 10 \\
  c_2 + c_3 \cdot n + 2 \cdot T\left(\frac{n}{3}\right) & \text{For } n \geq 10 
\end{cases}
\]

Yipee!!!! **YOU DO NOT NEED TO SOLVE** *this* recurrence…
7. (10 pts) Solving Recurrences

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

\[ T(1) = 7. \]

and

\[ T(N) = T\left(\frac{N}{3}\right) + 5 \quad \text{for integers } N > 1 \]

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.

\[
T(N) = T\left(\frac{N}{3}\right) + 5
\]

\[
= T\left(\frac{N}{9}\right) + 5 + 5
\]

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

\[
= T\left(\frac{N}{3^k}\right) + 5 \cdot k
\]

when

\[
\frac{N}{3^k} = 1
\]

\[
\log_3 (3^k) = \log_3 (N)
\]

\[
k = \log_3 N
\]

\[
= 7 + 5 \cdot \log_3 N
\]
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.

\[
4 \cdot (M-1) + 8 \cdot M \leq \text{page size}
\]
\[
4 \cdot 20 + 8 \cdot 21
\]
\[
80 + 168
\]
\[
\text{At least } 248 \text{ bytes needed for an interior node}
\]

\[
20 \cdot L \leq \text{page size}
\]
\[
20 \cdot 12
\]
\[
240
\]
\[
\text{Also: } 240 \text{ bytes needed for a leaf node}
\]

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:**

```
2 14
7 20
7
```

After inserting 50:

```
2 14
7 20
50
```

After deleting 14:

```
2 14
7 20
```

```
38 58 86
49 58 86
38 49 58 65 86 95
2 7 40 49 65 95
```