Name: Solutions

UW Net ID: ____________________

CSE373 Spring 2015: Midterm

(closed book, closed notes, NO calculators allowed)

Instructions: Read the directions for each question carefully before answering. We may give partial credit based on the work you write down, so if time permits, show your work! Use only the data structures and algorithms we have discussed in class or that were mentioned in the book so far.

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

Good luck!

Total: 60 points. Time: 50 minutes.

Question	Max Points	Score
1	10	
2	8	
3	8	
4	16	
5	8	
6	10	
Total	**60**	

Name: ________________________________

1) **[10 points total] Big-Oh and Run Time Analysis:**
This question has two parts. Determining the Big-Oh based on the code and on functions.

Describe the worst case running time of the following psuedocode and functions in Big-Oh notation in terms of the variable N. You should give 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). You must choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of a.-e.)):

$O(N^2)$, $O(N^{1/2})$, $O(N^3 \log N)$, $O(N \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^{12})$, $O(N^N)$, $O(N^6)$, $O(N^8)$, $O(N^9)$, $O(N^{10})$

a)
```
int f1(int N)
     for(int j = 0; j < N; j++) {
         for(int k = 0; k < N; k++) {
             for(int m = 0; m < k; m++) {
                 sum++;
     } } }
 }
```
______$O(N^3)$_____

b)
```
int f2(int N){
     int sum = 0;
     for(int j = 1; j < N; j *= 2){
         for(int k = 0; k < N; k++){
             sum++;
     } }
 }
```
_____$O(N \log N)$__

c)
$f(N) = 4N^2 + 2N \log N + 3N + 17$ __________ $O(N^2)$______

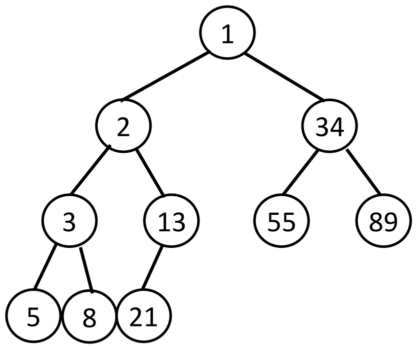
d)
$f(N) = (N + 4 \log N)^3 + N^2$ __________ $O(N^3)$________

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

Name: ________________________________

2) [8 points total] Trees. Please answer circle the correct answer for each tree.

a)



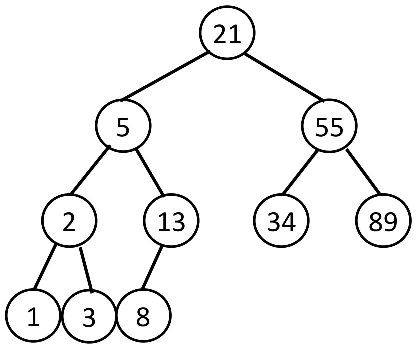
This diagram shows a(n):

AVL Tree

[✓ Heap]

Neither of the Above

b)



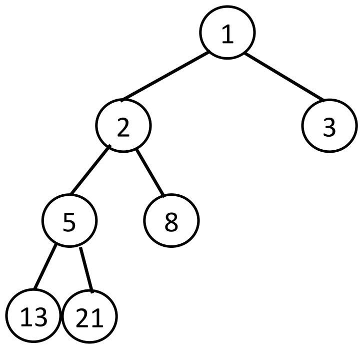
This diagram shows a(n):

[✓ AVL Tree]

Heap

Neither of the Above

c)



This diagram shows a(n):

AVL Tree

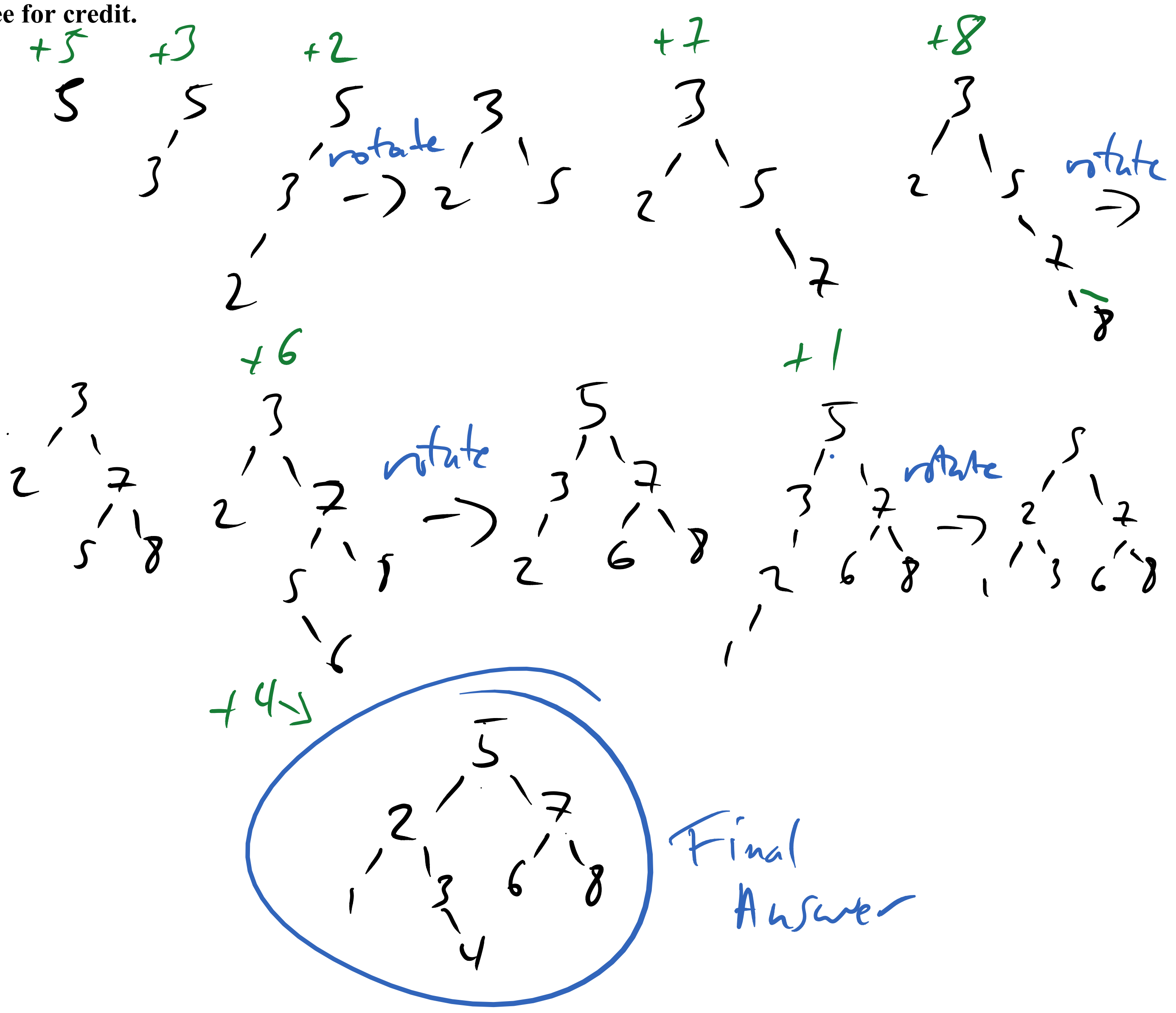
Heap

[✓ Neither of the Above]

d) The last diagram is: Balanced Complete [✓ Full] Perfect

Name: ________________________________

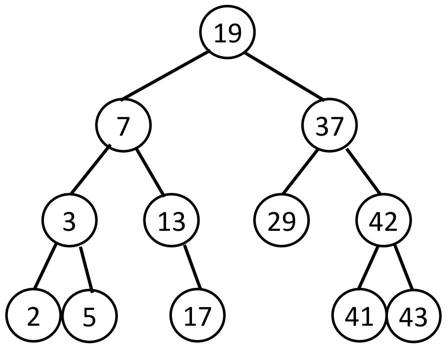
3) **[8 points total] AVL Trees:** Draw the AVL tree that results from inserting the keys: **5, 3, 2, 7, 8, 6, 1, 4** in that order. Start with an initially empty AVL tree. You are only required to draw the final tree, but you may get partial credit for drawing intermediate trees. **Circle your final tree for credit.**



Name: ______________________________

4) [16 points total] Binary Search Tree. This problem (over two pages) will ask about Binary Search Trees across 4 sub-questions. *Do not forget to answer e.*

a) Please write the keys of the tree in the order that results from a pre-order traversal.



19	**7**	**3**	**2**	**5**	**13**	**17**	**37**	**29**	**42**	**41**	**43**

b) Now, write the keys of this tree in sorted order.

2	**3**	**5**	**7**	**13**	**17**	**19**	**29**	**37**	**41**	**42**	**43**

c) The traversal you did in part b is called a(n) ____**in-order**____________ traversal.

d) In order to delete a node with two children from a Binary Search Tree, you need to find the successor of the node. On the next page, write down the method `successor(BSTNode)` which will find the successor of that node (the node with the smallest key that is greater than the parameter node's key) and return it. You must do this by taking advantage of the properties of a Binary Search Tree. You can assume that the parameter node has both children. Presume that you are already given the following methods:

```
public class BSTNode {
      public int key;
      public String value;
      public BSTNode left, right;
}
public class BinarySearchTree {
      private BTSNode root;
      public String find(int key) { ... }
      public void insert(int key, String value) { ... }
      public void delete(int key) { ... }
```

}
4d, cont.) Write your Java code here, you may want to use a helper method:

```
public BSTNode successor(BSTNode node){

      return findMin(node.right);

}


public BSTNode findMin(BSTNode root){

      while(root.left != null){
            root = root.left;
      }
      return root;
}
```

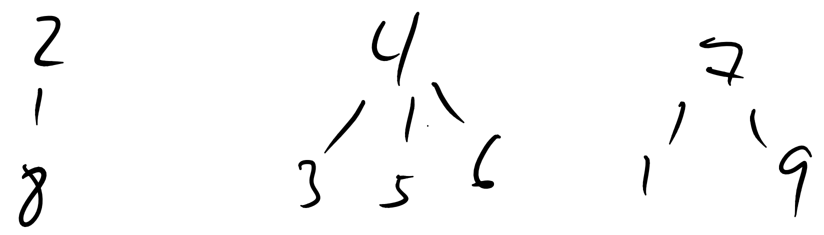
e) What is the worst-case runtime of your algorithm? ______**O(N)**_______________

Name: ________________________________

5) **[8 points total] Union Find.** Below is the array which represents an up-tree as an underlying data structure for the Union Find ADT.

7	**-2**	**4**	**-4**	**4**	**4**	**-3**	**2**	**7**

a) Draw the up tree(s) represented by this array. Each node can be numbered by its corresponding index and assume the indexing begins at 1 (not 0).



b) Write the corresponding array after performing **Union(4, 7)** with **union-by-size.**

7	**-2**	**4**	**-7**	**4**	**4**	**4**	**2**	**7**

will not compress

c) Based on your answer to part b, write the corresponding array after performing **Find(9)** with **path-compression**.

7	**-2**	**4**	**-7**	**4**	**4**	**4**	**2**	**4**

d) Based on your answer to part c, what does **Find(9)** return?

4

Name: ________________________________

6) **[10 points total] Hash Tables**
The following questions are short answer questions regarding hash tables. You should not need more than 2 or 3 sentences at most to answer the questions.

a. Why do you need to implement `hashcode` if you implement `equals`?

Hash tables require that if a and b are equal, then their hashcode needs to be equal. Therefore, if you change what determines if objects are equal, then you need to change hashcode. Otherwise, two elements that are equal under your new definition of equals may not hash to the same value. This will create problems with insert and find where the same item could be in the table twice or you may not find an item that is really in the table.

b. Name 2 things you can do to increase the likelihood that insert will be constant time in your implementation of hash table.

Pick 2 {

- **Make the table size a prime number**
- **Use Separate chaining, quadratic probing, or double hashing over linear probing**
- **Use a good hash function**
- **Keep the load factor low (same thing as increase the table size)**

c. Explain the difference between open addressing and separate chaining.

When an entry is added to a bucket that already has an entry, open addressing finds a new bucket, separate chaining overloads the bucket by making it a list of entries

d. Why should we keep the load factor under 1/2 for quadratic probing?

When the table size is prime and we keep the load factor under ½, we are guaranteed to find a spot in the table in (table size)/2 probes. If the load factor is above ½, we are not guaranteed to find a spot and might have an infinite loop

e. Explain one benefit quadratic probing has over linear probing.

Linear probing suffers from primary clustering. Quadratic probing helps avoid that problem as probes will quickly leave the neighborhood.