Name:_____

CSE373 Winter 2014, Midterm Examination
January 29, 2014

Please do not turn the page until the bell rings.

Rules:

- The exam is closed-book, closed-note, closed calculator, closed electronics.
- **Please stop promptly at 3:20.**
- There are **90 points** total, distributed **unevenly** among **7** questions (many with multiple parts):

Question	Max	Earned
1	14	
2	15	
3	12	
4	17	
5	10	
6	12	
7	10	

Advice:

- Read questions carefully. Understand a question before you start writing.
- **Write down thoughts and intermediate steps so you can get partial credit. But clearly circle your final answer.**
- The questions are not necessarily in order of difficulty. **Skip around.** Make sure you get to all the problems.
- If you have questions, ask.
- Relax. You are here to learn.

Name:_____

1. (**14** points) For each function $f(n)$ below, give an asymptotic upper bound using "big-Oh" notation. You should give the tightest bound possible (so giving $O(2^n)$ for every question is unlikely to result in many points).

   (a) $f(n) = n^4 + n^3 + 2n^2 + 3n + 5$ _____

   (b) $f(n) = \log_{10}(n^2 n^7 n)$ _____

   (c) $f(n) = 6n \log\log(n^2) + 2n \log^2 n + n \log n$ _____

   (d) $f(n) = \log n(n^4 - 6) + n^2(n^3 + 4n)$ _____

   (e) $f(n) = 45n^4 - 45n^4 + 17n^{1/2} + 13n$ _____

   (f) $f(n) = \frac{n^2+3n+2}{n+1}$ _____

   (g) $f(n) = (14 \log n)^2 + \log(3^n)$ _____

   Solution:

   (a) $O(n^4)$
   (b) $O(\log n)$
   (c) $O(n \log^2 n)$
   (d) $O(n^5)$
   (e) $O(n)$
   (f) $O(n)$
   (g) $O(n)$

Name:_____

2. (**15** points) Describe the worst case running time of the following code in "big-Oh" notation in terms of the appropriate variables. You should give the tightest bound possible.

(a)
```
void nero(int n) {
  for(int i=0; i < n; i++) {
    System.out.println("!");
  }
  for(int k=0; k < n*n*n; k++) {
    System.out.println("!");
  }
}
```

(b)
```
void claudius(int n) {
  for(int i=0; i < n; i++) {
    int j = 1;
    while(j < n) {
      System.out.println("j = " + j);
      j = j * 2;
  } }
}
```

(c)
```
int vespasian(int n, int m) {
  for(int i=m; i > 0; i--) {
    for(int j=0; j < 200; j++) {
      for(int k=0; k < n; k += 3) {
        System.out.println("$");
  } } }
}
```

(d)
```
int tonyblair(int n, int a) {
  if (a < 12) {
    for(int i=0; i < n; i++) {
      System.out.println("*");
    }
    tonyblair(n-1, a);
  } else {
    for(int k=0; k < 3000; k++) {
      for(int j=0; j < n * k; j++) {
        System.out.println("#");
  } } }
}
```

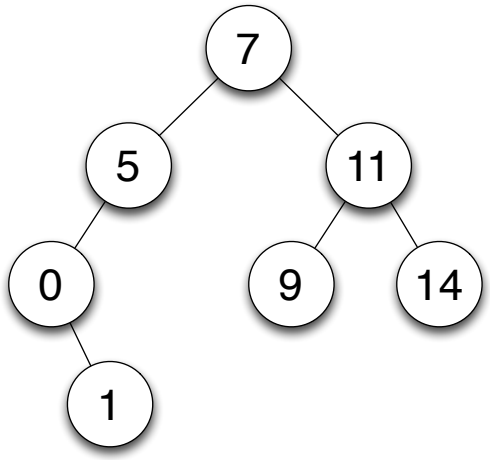
(e)
```
int augustus(int n) {
  int ans = 1;
  for(int i=0; i < n; i++) {
    ans += augustus(i);
  }
  return ans;
}
```

Solution:

(a) $O(n^3)$
(b) $O(n \log n)$
(c) $O(nm)$
(d) $O(n^2)$ or anything about non-terminating or infinity
(e) $O(2^n)$

Name:_____

3. (**12** points) The tree `T`, shown here, is a binary search tree where the keys are integers.



`T` has the following sequence of operations applied:

```
T.insert(8)
T.delete(0)
T.insert(17)
T.delete(11)
```

Please draw `T` after each operation using the space below (you can show just the keys, not the corresponding values). Also, for each step, say if `T` is an AVL tree. Note the operations are sequential, so each one is applied to the result of the previous operations.

- `T.insert(8)`
  Is `T` an AVL Tree? _____

Name:_____

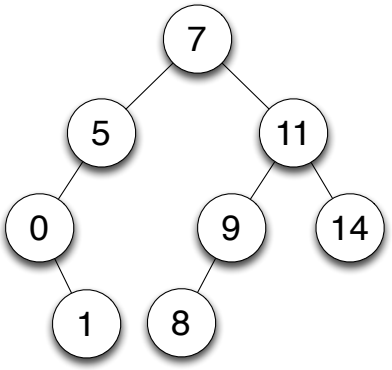
- `T.delete(0)`
  Is `T` an AVL Tree? _____

- `T.insert(17)`
  Is `T` an AVL Tree? _____

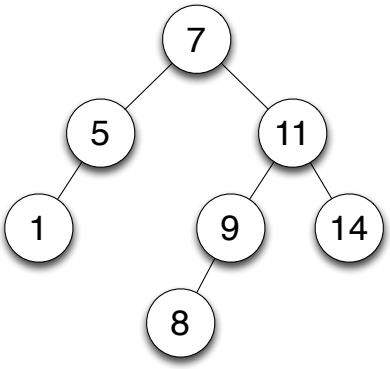
- `T.delete(11)`
  Is `T` an AVL Tree? _____

Solution:

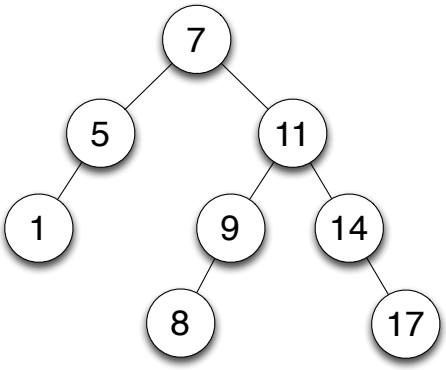
- NO,

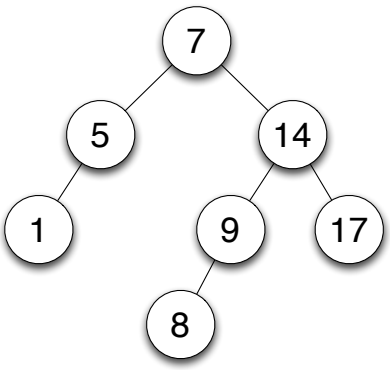


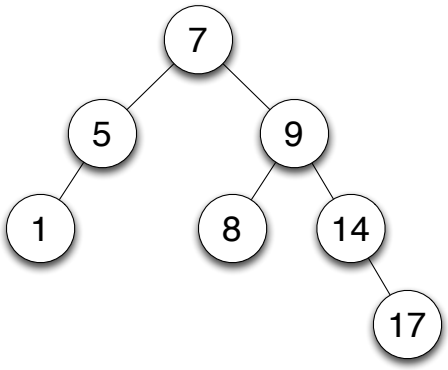
- YES,



- YES,



- YES, 

or 

Name:_____

4. (**17** points) In this problem, you will write some **Java code** for simple operations on **binary search trees** where keys are integers. Assume you already have the following code. Assume the method bodies, even though not shown, are correct and implement the operations as we defined them in class.

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

Don't overlook part (c).

(a) Add a method `public int keySum()` to the `BinarySearchTree` class that returns the sum of all the keys in the tree. You will need a helper method.

(b) Add method `public void deleteMin()` to the `BinarySearchTree` class that deletes the *minimum* element in the tree (or does nothing if the tree has no elements). You may want a helper method.

(c) For your answers in part (a) and part (b), give a worst-case big-Oh running time in terms of n, where n is the number of nodes in the tree. Do not assume the tree is balanced.

There is room below and on the next page for solutions.

Name:_____

More space for solutions.

Solution:

(a)
```
public int keySum() {
   return keySumHelper(root);
}
int keySumHelper(BinarySearchTreeNode node) {
   if (node != null) {
      int sum = node.key);
      sum += keySumHelper(node.left);
      sum += keySumHelper(node.right);
      return sum;
   }
   return 0;
}
```

(b)
```
public void deleteMin() {
   if(root == null)
      return;
   else if(root.left == null) // root is min
      root = root.right;
   else
      deleteMinNode(root);
}
// recursive version
void deleteMinNode(BinarySearchTreeNode prev) {
   if(prev.left.left == null)
     prev.left = prev.left.right; // partial credit: prev.left = null;
   else
     deleteMinNode(prev.left);
}
// iterative version [helper method not needed]
void deleteMinNode(BinarySearchTreeNode prev) {
   while(prev.left.left != null)
     prev = prev.left;
  prev.left = prev.left.right; // partial credit: prev.left = null;
}
// A third version that uses the provided delete method
public void deleteMin() {
   if(root == null)
     return;
   BinarySearchTreeNode n = root;
   while(n.left != null)
      n = n.left;
   delete(n.key);
}
```

(c) Both operations are $O(n)$ in the worst case.

Name:_____

5. (**10** points) Below are five data sets or situations that you have been tasked with representing in software. For each one, state which of the abstract data types we have studied you feel is best suited to represent the data set or situation. In each case, please briefly (in no more than two sentences) describe how the ADT will be used and justify your choice. For reference, we have covered the following ADTs so far: Stack, Queue, Set, Dictionary, and Priority Queue.

   (a) A line of people in a cafeteria.

   (b) The names of candidates for political office and their party affiliation (e.g. Democrat, Republican).

   (c) Active tech support requests where the most urgent will be resolved first.

   (d) A catalog of items and their prices. The items have alphanumeric id codes.

   (e) The species of animals living at a zoo (just the species, not how many there are).

Solution:

(a) Use a queue since a cafeteria line is FIFO.

(b) Use a dictionary with the candidates names as keys and the party affiliation as values to associate the information and facilitate party lookup via name.

(c) Use a priority queue with the urgency of the requests as the priority, so `deleteMin` will return the next request to be resolved.

(d) A dictionary with id codes as keys and prices as values to facilitate price lookup via id code.

(e) A set since there shouldn't be duplicate species and there's no ordering.

Name:_____

6. (**12** points) An empty binary min-heap `H` in which the priorities are integers has the following sequence of operations applied to it:

```
H.insert(10)
H.insert(5)
H.insert(7)
H.insert(8)
H.deleteMin()
H.insert(3)
```

For each operation, please draw the appropriate tree *and* the corresponding array representation using the space below (you can show just the priorities, not the corresponding items). Note the operations are sequential, so each one is applied to the result of the previous operations.

- `H.insert(10)`

- `H.insert(5)`

- `H.insert(7)`

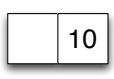
Name:_____

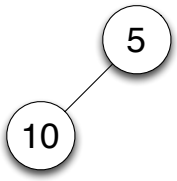
- `H.insert(8)`

- `H.deleteMin()`

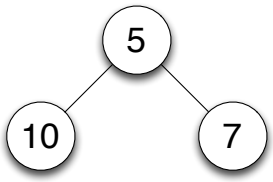
- `H.insert(3)`

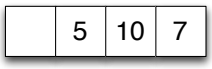
Solution:

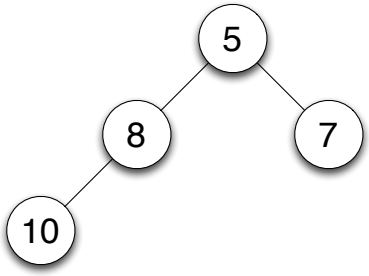
- 

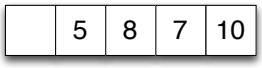

5
10

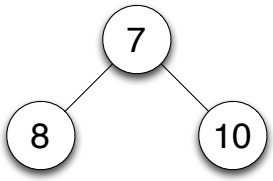
- 
5
10


5
10
7

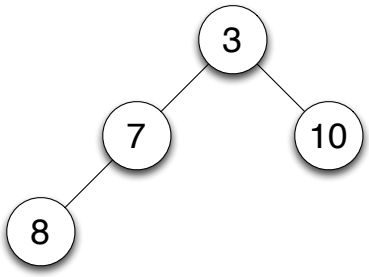
- 
5
10
7


5
8
7
10

- 
5
8
7
10


7
8
10

- 
7
8
10


3
7
10
8

- 
3
7
10
8

Name:_____

7. (**10** points) Answer each of the following; no explanations required.

   (a) What is the worst-case asymptotic running time for the best algorithm for finding the mode (i.e. the most frequently occurring value) of the keys in a binary search tree?

   (b) What is the worst-case asymptotic running time for the best algorithm for finding and deleting the maximum element in an AVL tree?

   (c) What is the worst-case asymptotic running time of sorting a sequence of integers using a priority queue implemented with a binary min-heap?

   (d) What is the worst-case asymptotic running time for the best algorithm for finding something in a sorted array?

   (e) What is the worst-case asymptotic running time of finding and removing all values greater than 12 from a stack implemented with a linked-list (leaving the rest of the stack in its original order)?

Solution:
Note: We give answers here with O, but Θ would technically be more precise.

(a) $O(n)$
(b) $O(\log n)$
(c) $O(n \log n)$
(d) $O(\log n)$
(e) $O(n)$