UC Berkeley – Computer Science
CS61B: Data Structures

Midterm #2, Spring 2015

This test has 10 questions worth a total of 35 points. The exam is closed book, except that you are allowed to use two (front-and-back) handwritten pages of notes. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write the statement out below in the blank provided and sign. You may do this before the exam begins.**

"I have neither given nor received any assistance in the taking of this exam."

____
____

Signature: ______________________

	Points		Points
0	0.5	5	2.5
1	6	6	6
2	5	7	2
3	2	8	3.5
4	5	9	0
		10	2.5

Your Name: ______________________

Your Student ID: ________________
Three-letter Login ID: ___________
Login of Person to Left: __________
Login of Person to Right: ________
Exam Room: _____________________

Tips:

- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. Do not get overly captivated by interesting design issues or complex corner cases you're not sure about.
- Not all information provided in a problem may be useful.
- All code that we've providedon this exam should compile. All code has been compiled and executed before printing, but in the unlikely event that we do happen to catch any bugs in the exam, we'll announce a fix. The correct answer is not 'does not compile.'
- Don't panic! Recall that we shoot for around a 60% median. You should not expect to be able to do all of the problems on this exam.
- If you're feeling anxious and need to take a break, go for it. Just let a TA know, and leave your test and electronic devices behind.

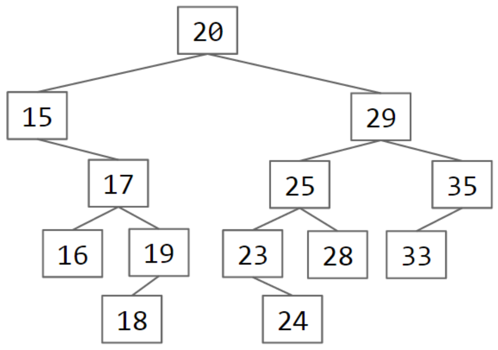
Optional. Mark along the line to show your feelings on the spectrum between ☹ and ☺.

Before exam: [☹__________________☺].
After exam: [☹__________________☺].

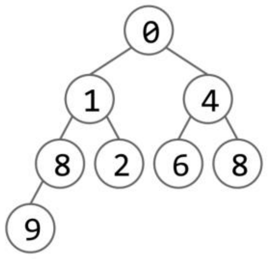
0.Another half point.(0.5 points). Write your name, login, and ID on the front page. Write your exam room. Write the IDs of your neighbors. Write the given statement and sign. Write your login in the corner of every page. Enjoy your free half point. ☺

1. **Basic Operations (6 Points).**

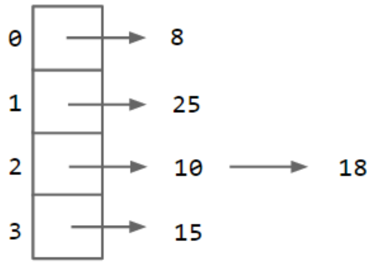
a. **To the right of the BST below**, draw a BST that results if we delete 20 from the BST. You should use the deletion procedure discussed in class (i.e. no more than 4 references should change).



b. **To the right of the minHeap below**, draw the minHeap that results if we delete the smallest item from the minHeap.



c. **To the right of the External Chaining Hash Set below**, draw the External Chaining Hash Set that results if we insert 5. As part of this insertion, you should also resize from 4 buckets to 8 (in other words, the implementer of this data structure seems to be resizing when the load factor reaches1.5). Assume that we're using the default `hashCode` for integers, which simply returns the integer itself.



d. Draw a valid Weighted Quick Union object that results after the following calls to connect: `connect(1, 4)`, `connect(2, 3)`, `connect(1, 3)`, `connect(5, 1)`. Don't worry about the order of the arguments to each connect call, we'll accept any reasonable convention.

2. **Asymptotics (5 Points)**

a. Suppose we run experiments to understand the runtime performance of the `add` method of the `PotatoSack` class. The runtime as a function of N (the number of inserts) is shown below. Using the technique from the asymptotics lab, ***approximate*** the empirical run time in ***tilde notation*** as a function of N. As a reminder, in that lab, we assumed that the runtime is $\sim aN^b$, and found$a$ and b.Do not leave your answer in terms of logarithms. Your $a$and $b$must be within 25% of our answers. Use only the data points that you expect to give the best approximation of the asymptotic behavior of the algorithm.Hint: To double check your answer, plug in N=1000 and see if the runtime prediction seems sensible.

```
N   Time (s)
         1     0.00
        2     0.01
        3     0.01
        6     0.03
        13     0.16
       25     0.63
       50     2.50
     100     9.97
```

Answer: $\boxed{}$

b. Suppose we measure the performance of a collectionX, and find that inserting N items takes $\Theta(N^2)$ time. For each of the following**, circle the collection type if it is possible** for that collection to take $\Theta(N^2)$ time to insert N items on a worst-case input, and **cross out the collection type if it is impossible**. Assume that each is correctly implemented. Either circle or cross out every answer.

LinkedList	2-3 Tree Set	HeapMinPQ	LLRBST Set	Your BSTMap (from HW6)	External Chaining Hash Map	ArrayList

c. If we have two correct algorithms for solving the same problem that use the exact same amount of memory, but have worst-case runtimes that are $\Theta(N)$ and $\Theta(N^2)$, is it always better to use the algorithm that is $\Theta(N)$? If so, why? If not, why not?

3. **Exceptions (2 Points).**

One common software engineering strategy is to create log files that can be manually examined if something goes wrong. In the code below, the writeToLog method writes the given argument to some log. What are the contents of the log file after the code below is executed? You may not need all of the lines provided.

____

____

____

____

____

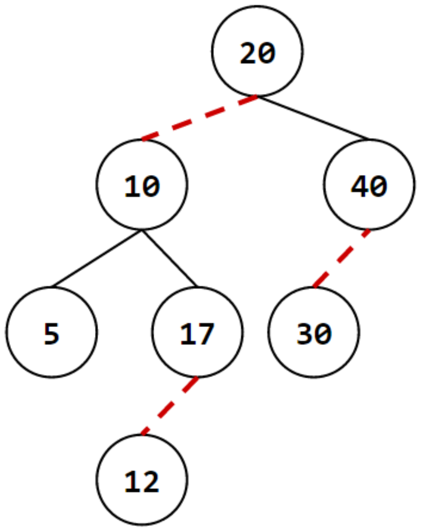
```
public class QuestionThree {
    public static void printTenth(int[] a) {
        try {
            writeToLog(a[10]);
        } catch(IndexOutOfBoundsException e) {
            writeToLog("No tenth item available!");
            throw(e);
        }
    }

    public static void main(String[] args) {
        printTenth(new int[]{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10});
        printTenth(new int[]{0, 1, 2, 3});
        printTenth(new int[]{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10});
    }
}
```

This area is a designated fun zone. Perhaps you would like to compose a poem in your native language about some topic of interest? Or perhaps you'd like to draw a dog with an overly ornate moustache?

4. **TreeTime (5 Points).**

a. True or false: If A and B are 2-3-4 trees with the same exact elements, they must be identical. If true, justify with a short **(less than 20 word)** explanation. If false, provide a counter-example.

b. Draw the red-black tree which results after calling `insert(25)` on the red-black tree shown below. We denote red links with dashed lines and black links with normal lines. Please use the same notation in your answer.**You should draw your tree in the empty space to the right of the given tree. Do not modify the given figure.**Hint: Every 2-3 tree corresponds to exactly one LLRB, and every LLRB corresponds to exactly one 2-3 tree.



c. Suppose that we want to write a method `sumDescendants`, which replaces the value of each node in a tree with the sum of all of its descendants' values (not including itself), and then returns the sum of its original value (before being changed) plus all of its descendants' values.

For example, given the tree on the left, `sumDescendants` on node 6 would return 42 and change the tree to look like the one on the right (since 36 + 6 = 42).

```
      6                    36
    /   \                /    \
   4     8              8     16
  / \   / \            / \   /  \
 3   5 7   9          0   0 0    0
```

Fill in the `sumDescendants` method. You may not need all lines. Do not use more lines.

```
    public class TreeNode {
        public TreeNode left, right;
public int value;

        public TreeNode(int n) {
            value = n;
        }

        /* Replaces value with sum of all of its descendants' values. */
        public int sumDescendants() {
            if (left == null && right == null) {
                int oldVal = value;
                ___________________________;
                return oldVal;
            }
            else {
_______________________________
_______________________________
                _______________________________
                _______________________________
                _______________________________
                _______________________________
                _______________________________
                _______________________________
                int oldVal = value;
                ___________________________;
                return oldVal + value;
            }
        }
}
```

5. Code Analysis (2.5 points).

For each of the pieces of code below, give the runtime in Θ(·) notation as a function of the given parameters. Your answer should be simple, with no unnecessary leading constants or unnecessary summations.

____

```
public static void f1(int n) {
     for (int i = 0; i < 2*n; i += 1) {
          System.out.println("hello");
     }
}
```

____

```
public static void f2(int n) {
     if (n == 0) { return; }
     f2(n/2);
     f1(n);
     f2(n/2);
}
```

____

```
public static void f3(int n) {
     if (n == 0) { return; }
     f3(n/3);
     f1(n);
     f3(n/3);
     f1(n);
     f3(n/3);
}
```

____

```
public static void f4(int n) {
     if (n == 0) { return; }
     f4(n-1);
     f1(17);
     f4(n-1);
}
```

____

```
public static void f5(int n, int m) {
     if (m <= 0) {
          return;
     } else {
          for (int i = 0; i < n; i += 1) {
               f5(n, m-1);
          }
     }
}
```

6. **The Right Tool for the Job (6 points).**

For each of the five tasks below, pick <u>one or two</u> data structures and describe very briefly (in 20 words or less) how you'd usethose data structures to solve the problem. You should select from the following Java Collections: TreeMap, TreeSet, HashMap, HashSet, LinkedList, ArrayList, HeapMinPQ, HeapMaxPQ, WeightedQuickUnion. TreeMap and TreeSet utilize red-black trees. HashMap and HashSet utilize external chaining.

You should pick the data structure (or structures) that are best suited to the task in terms of performance and ease-of-use, taking into account the specific types of inputs listed in each problem.For most problems, you should need only one data structure. If some part of the problem seems ambiguous, state the assumptions that you're making.

For each task, also give the runtime in $\boldsymbol{O}(\cdot)$ notation. Give the tightest bound you can (e.g. don't just write $O\left(n^{n^{n^n}}\right)$, which while technically correct, isn't very informative).

Task 1) Read in a text file and print all of the unique words that appear. The words should be printed in alphabetical order.Give the $\text{O}(\cdot)$ runtime in terms of N, the number of words in the file. Remember, you must provide either a choice of either <u>one or two</u> data structures, as well as a short (< 20 word) description of how you'd use those data structures to solve the problem.

Task 2) Given two Collections of very long Strings (e.g. DNA Sequences of tens of thousands of characters or more), `observed` and `known`, check each String in `observed` to see if it exactly matches any String in `known`. `observed` is an `ArrayList<String>` of size N_O. For this task, choose a data structure for `known`. Give the $\text{O}(\cdot)$ runtime in terms of N_O and N_K, where N_K is the size of `known`. Assume that the Strings in `known` are highly dissimilar from each other. Assume also that known has already been constructed.

Task 3) Read in a large number of grayscale images, and display all of the unique images. The images may be displayed in any order. Each image is stored as a Picture object that contains an`int[][]`variable,all of which are512 x 512 (i.e. have a width and height of 512).Give the O(·) runtime in terms of N, the number of images.

Task 4) Store the email address for each username on our website, which is devoted to publishingarticles about computer hackers being terrible people. Usernames and email addresses are both Strings(and thus use the default `.hashCode` and `.compareTo` method). Our website allows anybody to register any number of accounts.Give the O(·) runtime needed to add each user in terms of N, the current number of users.

Task 5)*Erweitetern Netzwerk*is a new German minimalist social network that provides three functions. Its user base is capped at a maximum of K users, where K is some large constant known at runtime:

- Neu: Enter a username and click the Neu button. Create a new user with this username if space is still available.
- Befreunden: Enter two usernames and click the befreunden button. The users are now friends.If one of the users does not exist, ignore the command.
- Erweiterten Netzwerk: Enter two usernames and click the Erweitertern Netzwerk button. The website prints true if there is a chain of user friendships that connect the users.

Your Neu, Befreunden, and Erweiterten Netzwerk commands must run in O(log N) time in the worst case. Give the O(·) runtime in terms of N, the number of users at the time the command was executed. Assume that $K > N$.

7. GorpyCorp (2 points).

Gorpy McGorpGorp is the founder ofGorpyCorp. GorpyCorp is organized into several independent teams known as ”Circles”, where every Circle has a leader known as a “lead link”. Gorpy uses the following data structure to record the `members`and `teamName` of each Circle:

```
public class Circle {
      HashSet<Member> members;
      String teamName;

      public int hashCode() {
            int hashCode = 0;
            for (Member m : members) {
                  hashCode = hashCode * 31 + m.hashCode();
            }
            hashCode = hashCode + teamName.hashCode();
            return hashCode;
      }

      public int compareTo(Circle other) {
            if (this.members.size() == other.members.size())
                  return this.teamName.compareTo(other.teamName);
            return this.members.size() – other.members.size();
      }

      public void addMember(Member newMember) {
            members.add(newMember);
      }

      ...
}
```

Rather than storing the leader of each Circle inside of the Circle object, Gorpy instead decides to create a separate HashMap defined below, which allows a programmer to lookup the lead linkof each circle.

```
HashMap<Circle, Member>leadLinks;
```

What is the most significant problem with Gorpy’s usage of a HashMap? If he uses a TreeMap instead of a HashMap, will this problem be fixed? Why or why not?

8. By the Numbers (3.5 Points).

For each of the scenarios below, give the correct numbers. On each line**, in the first blank write the minimum, and in the second blank write the maximum**. We define the height as the maximum number of links from the root to a leaf (so the tree in problem 1b has a height of 3, not 4). Each blank is worth 0.25 points (so don't burn all your time trying out bajillions of examples).

____ ____ The minimum and maximum height of a BST with 15 nodes.

____ ____ The minimum and maximum height of a Quick Union object with 15 elements where `isConnected(a, b)` returns true for every pair of items.

____ ____ The minimum and maximum height of a Weighted Quick Union object with 15 elements where`isConnected(a, b)` returns true for every pair of items.

____ ____ The minimum and maximum height of a 2-3-4 Tree containing 15 items. Recall that a node in a 2-3-4 tree may have 1, 2, or 3 items inside.

____ ____ The minimum and maximum height of an LLRB set containing 15 items.

____ ____ The minimum and maximum height of a binary heap containing 15 items.

____ ____ The minimum and maximum number of items in a single bucket for a chaining hash table with 30 items and 15 buckets (i.e. with a load factor of 2).

9. PNH (0 Points). Who was the *agoyatis* of Mr. Conchis?

10. Quartiler (2.5 points)

Warning: This problem is particularly challenging. **Do not start until you feel like you've done everything else you can.** We will be award very little partial credit for this problem. Solutions which are correct but do not meet our time and space requirements (below) will be not be awarded credit.

The interface for the `Quartiler` interface is shown below.

```
public interface Quartiler<Item> {
      /* Adds an item to the Quartiler. */
      public add(Item x);
      /* Gets the item that is closest to the 75th percentile. */
      public getTopQuartile();
      /* Deletes the item that is closest to the 75th percentile. */
      public deleteTopQuartile();
}
```

For example, if we add the integers 1 through 100, then `getTopQuartile`will return 75. If we instead add the integers 1 through 4 to an empty `Quartiler`, then `getTopQuartile` will return 3. If there is a tie (e.g. if the `Quartlier` contains the integers 1 through 6), then ties may be broken arbitrarily. Design a data structure that supports these operations in amortized $O(\log N)$time and $O(N)$ space.

a. Describe your data structure as concisely as possible while still including all relevant details. If you use existing data structure (for example, those listed in problem 6) as components, give them by name.

b. Draw your data structure after the following numbers have been added (in this order): 5, 1, 3, 2, 4, 6, 7, 8. You only need to draw the data structure after all 8 insertions have been completed.

c. Draw your data structure after a subsequent call to `deleteTopQuartile` (which will remove the 6).