CS 151 Final Exam Spring 2022

Name _______________

Unless you qualify for extended time on exams, **you have 2 hours to complete this exam** once you start it.

The 8 numbered questions are equally weighted.

If you forget the name of something (oh, what does Java call the length of a String??) just write a note saying "I'm going to call this X" and then use that.

Please do not write on the backs of the pages. If you need more space for a question there is a blank page at the end.

After you have finished the exam please indicate whether you followed the Honor Code on the exam.

I ☐ did ☐ did not

adhere to the Honor Code while taking this exam.

____

Signature

If you did not take the exam with the rest of the class please give your starting and finishing times:

I started the exam at time ________ and finished at time _________

1. Here is a list of data: 7 12 4 9 5 3 2 8 10
For each of the following structures I will walk through the data list in order, add each item to the structure and then go into a loop in which I remove elements one at a time from the structure and print them as I remove them. **In what order do I print the items for**

a) **A stack**. Using push( ) to add to the structure and pop( ) to remove.

This reverses the order: 10 8 2 3 5 9 4 12 7

b) **A queue**. Using offer( ) to add to the structure and poll( ) to remove.

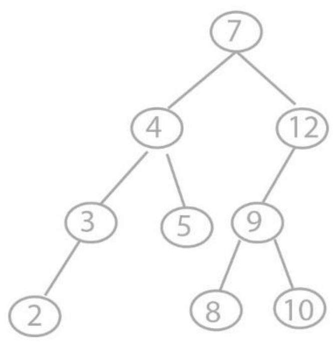
This preserves the order: 7 12 4 9 5 3 2 8 10

c) **A priority queue**. Using offer( ) to add to the structure and poll( ) to remove.

This puts them in order from smallest to largest: 2 3 4 5 7 8 9 10 12

d) In the add stage, I insert the values into a **BinarySearchTree** that starts off empty. So 7 becomes the root and I insert the other values around it. Skip the remove stage and instead give **preorder traversal** of the tree.

This builds the tree:



A preorder traversal is 7 4 3 2 5 12 9 8 10

.

2. In each part **give a Big-Oh estimate of the worst-case time it takes to complete the operation in the given structure**. If there are multiple ways to do something use the fastest way.

   a) Inserting into an AVL tree with n nodes

      O(log(n))

   b) Finding, then removing, a node from a Binary Search tree

      O(n) The "tree" could be very unbalanced.

   c) Finding an element in a sorted ArrayList with n elements. Here "finding" means returning the index of the first occurrence of the element in the list if it is in the list, and returning -1 if the element is not in the list.

      O(log(n)) Binary Search

   d) The same as (c), only with a sorted LinkedList.

      O(n) BinarySearch won't work in a linked list

   e) Polling a Priority Queue with n values.

      O(log(n))

   f) The addEdge(String source, String dest) from our Graph class of Lab 9, if the graph has n vertices.

      O(n) The HashMaps that let us look up a node from its name
      have O(n) as their worst-case times

3. Here is a function that computes the average value of a linked list of integers:

```
float avg(LinkedList<Integer> L) {
     If (L.size() == 0)
          return 0.0;
    float sum = 0.0;
    for( int i = 0; i < L.size( ); i++ )
              sum += L.get(i):
    return sum/L.size( );
}
```

a) Why is this not a good way to compute the average of a very long list?

Indexing through a linked list is O(n^2)

b) Give code for a better way to compute the average of a LinkedList.

```
float avg(LinkedList<Integer> L) {

     If (L.size() == 0)
          return 0.0;
    float sum = 0.0;
    Iterator<Integer> it = L.iterator( );
    while( it.hasNext() )
              sum += it.next( ):
    return sum/L.size( );
}
```

4. In Lab 8 we implemented PriorityQueue<T> in an ArrayList<T>; let's call that ArrayList *data*. The priority queue uses a Comparator<T> *cmp* to order the values. The PriorityQueue class has a method

   T peek( )

   that returns the smallest element in the queue, or null if the queue is empty. **Give code for the new method**

   T peek2( )

   which returns the second-smallest value in the queue, or null if the queue has fewer than 2 values. As with peek( ), calling peek2( ) should not change the queue. For example, if the priority queue has data 6 4 8 2 9 3 5 7 then peek2( ) should return 3. Your peek2( ) method should run in time O(1).

   The second-smallest element will be the smaller of the children of the root, if it has two children .

```
T peek2( ) {
     if ( size() < 2)
          return null;
     else if (size( ) == 2)
          return data.get(2);
     else {
        if (cmp.compare(data.get(2), data.get(3)) < 0)
               return  data.get(2);
       else return data.get(3);
}
```

5. In Lab 5 we had an abstract BinaryTree<T> class that united classes EmptyTree<T> and ConsTree<T>. Give code for two methods:

   int height( ), which returns the height of the tree (You wrote this in Lab 5.)
   int heightBalanceFactor( ), which returns the maximum over all of the nodes in the tree of the absolute value of the difference in the height of the left child and the height of the right child. (This is new.)

   a) **Give height( ) and heightBalanceFactor( ) for an EmptyTree<T>**

```
int height( ) {                         int heightBalanceFactor( ) {
        return -1;                                      return 0;
}                                       }
```

   b) **Give height( ) and heightBalanceFactor( ) for a ConsTree<T>.** The class variables for ConsTree<T> are *data* (an object of type T), *leftChild* and *rightChild* (BinaryTree<T> objects).

```
int height( ) {
        return 1 + Math.max( leftChild.height( ), rightChild.height( ) );
}

int heightBalanceFactor( ) {
        int v1 = Math.abs(leftChild.height() – rightChild.height() );
        int v2 = leftChild.heightBalanceFactor( );
        int v3 = rightChild.heightBalanceFactor( );
        return Math.max( v1, Math.max(v2, v3) );
}
```

6. An application requires you to read a text file and keep track of every word, what words it was followed by, and how often. For example, if the text file is “a rose is a rose is a rose is an onion” the words are

   “a” followed by “rose” 3 times
   “rose” followed by “is” 3 times
   “is” followed by “a” 2 times, followed by “an” 1 time
   “an” followed by “onion” 1 time
   “onion” followed by the end of the input 1 time

   What data structures will you use to store this information? Give an algorithm in English for reading a text file word by word and building up this information. I know you know how to use Scanners to read a file word by word, so you can just say “get the next word” and represent the end of the input by “EOF” (EOF stands for “End of File”).

   - Class Word will have variables String word (which holds the word this class represents) and TreeMap<String, Integer> follows which holds the followup words and their counts.
   - Class Text has a HashMap<String, Word> that holds an instance of Word for each word in the text file.

   To read the file, keep two String variables w and f (for “word” and “following word”). Start with w the first word in the file. Then go into a loop until you reach the end of the file:

   - Let variable f be the next word in the file
   - If the HashMap has an entry for variable w, call that entry myWord; otherwise let myWord be a new Word object.
   - Add f to the TreeMap in myWord. If necessary put myWord into the HashMap
   - Prepare to read the next word from the file by setting w=f.

7. Hackers have broken into Oberlin's database and destroyed all grade information for last semester except for one backup file. Each line of the file has the format

   <student-name>|<class-name>#<grade>

   The lines of the file are in no particular order. You need to read this file and produce a list of every student and their gpa for the term, and also a list of all of the classes that were offered, the students in the class, and the grade for each student for the class. The first list might look like:

   ```
   Barney Rubble  2.1
   Fred Flintstone 3.2
   …………
   ```

   and the second list might be

   ```
   CINE 101
       Rocket J. Squirrel A
       Bullwinkle Moose C-
       Boris Badenov  D
       ….
   CINE 102
       Yogi Bear   B
       Booboo  Bear  B+
       Mr. Ranger  A-
       ………
   ```

   How will you use data structures to solve this problem? **Your answers should be in English, not code**. In particular:

   a. What will you do with each line of the file?
      Build a graph with nodes for classes and nodes for students. Also maintain a list of student and a list of classes. Each time you see a line student|class#grade get the student and class nodes; if you have to create a new node add it to the student list or class list. Then add a weighted edge with the grade as a weight from the student to the class and another weighted edge from the class to the student.

   b. How will you print the first list of students and their gpas
      Run through the student list and print each student's name. Go to the graph node for the student; each outgoing edge from this node represents a class the student has taken. Average the grades on these edges to get the student's gpa.

   c. How will you print the second list of classes, their students and grades?

      Run through the list of classes and print each class's name. Go to the graph node for the class. There is an outgoing edge for each student in the class, so you can print the name and grade for each student in the class.

8. Oh, no! You fell asleep during the CS 151 final and now you find yourself inside a video game with just your laptop. You seem to be in a gigantic house with many rooms and doors connecting the rooms. The name of each room is painted on the ceiling so you know where you are; you just have to find your way out, by getting to a room called Exit. Instead of a map you have a file that lists all of the doors of the house; a line in the file such as

Dining Room:Kitchen

means there is a door connecting the Dining Room and the Kitchen. You can go through the door in either direction. To make matters worse, all of the doors in the house are guarded by trolls. To go through a door you have to distract its troll by giving it a cookie. You have a small bag of cookies and no way to get any more. If you can find your way out before you run out of cookies, you will wake up in the wonderful Land of Internships. If you don't find your way out, those trolls look very hungry ….

Give an algorithm, in English, for how to find your way out.

Build an unweighted graph with a node for every room. Give every node a *predecessor* node, which should initially be null. For each line A:B of the file add a graph edge from node A to node B, and an edge from B to A. Then run the shortest path algorithm, using the room you are currently in as the source. For this algorithm you need a queue of nodes. Start by putting the node for the current room into the queue. Then do the following until the queue is empty:

- Let X be the result of polling the queue.
- For every edge going out from X, let Y be the edge's destination.
  If Y's predecessor is null, then make Y's predecessor be X and offer Y to the queue.

This loop can end either when the queue becomes empty or when the node for the Exit room has its predecessor set. When that happens make a Stack of nodes, which initially has the Exit node in it. If top is the node at the top of the stack we push top.predecessor onto the stack. This continues until the node at the top of the stack doesn't have A predecessor. Until the stack is empty pop and write down the name of the popped room. That should give you the shortest path from your current room to the exit.

;;;;

You can use this page as extra space for any question. Be clear about which question you are answering.