CS 455 MT 2 Sample Questions

This doesn’t cover everything. There’s likely to be at least one question about exceptions (e.g., writing some code with exceptions, or saying what some code does that uses exceptions), and possibly a question related to interfaces, e.g., you implement an interface. For exceptions you should know which general categories of exceptions are checked, and which are unchecked, and what that means (you do not have to memorize the whole exception hierarchy.) (I did include some textbook problems on some of these topics.)

Besides the textbook problems, these are just a bunch of old questions I have asked on previous exams. This problem set definitely longer than the real exam will be, and doesn’t necessarily represent the exact mix of questions you will get on your exam.

**Textbook problems (note: these are from the 4th edition of Big Java):**

R10.13
R11.7, R11.9, R11.13, R11.18, P11.16
R14.10 (also implement the alg described), R14.11, P14.7 (calling, not implementing a sort), P14.13
R15.11, P15.1, P15.2, P15.14, P15.15
P16.3, P16.7

**Problem 1.** Consider inserting the following keys that have the following hash values into a hash table that has 11 buckets (indices 0 through 10) and that uses chaining. Show the resulting data structure.

<table>
<thead>
<tr>
<th>Key</th>
<th>hash value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Ashley</td>
<td>5</td>
</tr>
<tr>
<td>b. Joe</td>
<td>3</td>
</tr>
<tr>
<td>c. Ted</td>
<td>4</td>
</tr>
<tr>
<td>d. Bob</td>
<td>3</td>
</tr>
<tr>
<td>e. Dana</td>
<td>10</td>
</tr>
<tr>
<td>f. Sue</td>
<td>7</td>
</tr>
</tbody>
</table>

For each of the following lookups from the table above, give the sequence of values that the target value would have to be compared with to do the lookup.

**Part C.** lookup *Bob*; Bob hashes to 3

**Part D.** lookup *Fred*; Fred hashes to 6

**Problem 2.** Consider a Map class to store a collection of key-value pairs with operations *insert*, *remove*, *lookup*, and *printInOrder* (the last is to print all entries in order by key).

A. If the Map used a hash table internally, how long would *printInOrder* take in big-O terms?

B. How long would *printInOrder* take if we used a balanced tree instead?
**Problem 3.** Write the function `print10PerLine`, which prints all the values in a linked list of words, formatting them so there are ten words per line.

For example, if the vector had the following 23 words in it, it would get printed as follows:

```
this is some text that is being printed out 10
words per line here is some more text to make
this example longer
```

It’s fine if there is an extra space to the right of the last word in every line of output.

```java
public static void print10PerLine(LinkedList<String> wordVec) {
```

**Problem 4.** Write a new member function to the `Poly` class you wrote as part of PA3. The new member function is a version of `addIn` that allows you to add just one `Term` to a polynomial (our program had a version that allowed us to add a `Poly` to another `Poly`). This new function would be useful for implementing the “create” command of our PolyProg.

Recall that the representation used by the `Poly` class is an ordered Java LinkedList object of non-zero terms. That is, `<coeff, expon>` pairs are stored in decreasing order by exponent. Also, recall that the type `Term` is a class to store a `<coeff, expon>` pair (see `Term` class def. on code handout). Also, for the purposes of this question, assume that the zero polynomial is represented as an empty list. Hint: remember that Terms are allowed to have a coefficient of zero (e.g., just like the ones we pass to the 1-arg Poly constructor).

Your function may not call the other Poly `addIn` function.

Here is the private data of the `Poly` class:

```java
LinkedList<Term> data;
```

Here’s the interface for the function

```java
void addIn(Term t)
```

**Problem 5.** Assume you have a vector of names with the following contents (numbers above the line are indices):

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Ann</td>
<td>Bob</td>
<td>Carly</td>
<td>Dave</td>
<td>Ed</td>
<td>Frida</td>
<td>Hal</td>
<td>Hank</td>
<td>Iris</td>
<td>Joe</td>
<td>Kat</td>
<td>Lou</td>
<td>Mary</td>
</tr>
</tbody>
</table>

For each of the targets given below, list the **names** that the target would be compared with in a **binary search** on the vector. List them in the order that the comparisons would be done.

1. target = **Ed**
2. target = **Joe**
3. target = **Sam**
Problem 6. Write a function to compute the intersection of two TreeSet’s of ints. To get full credit, take advantage of the fact that TreeSets are ordered containers. Hint: your solution will traverse each input set only once.

Example:

set1: 3 5 10 15
set2: 2 3 10 20 32

result of intersect(set1, set2):  3 10

// set1 and set2 are unchanged by this operation
public static TreeSet<Integer> intersect(TreeSet<Integer> set1, TreeSet<Integer> set2)

Problem 7. Implement the following function (see comments on function header).

Example:

<table>
<thead>
<tr>
<th>list before</th>
<th>amount</th>
<th>list after rotateLeft(list, amount)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b c d</td>
<td>1</td>
<td>b c d a</td>
</tr>
<tr>
<td>a b c d e</td>
<td>2</td>
<td>c d e a b</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>&lt;empty&gt;</td>
<td>3</td>
<td>empty</td>
</tr>
</tbody>
</table>

// rotate the linked list left by 'amount'. PRE: amount >= 0
public static void rotateLeft(LinkedList<Character> list, int amount)

Problem 8.

Part A Assume this uses a Stack class of our own devising which can hold ints, and has the usual operations push, pop, top, and, isEmpty, as well as a default constructor. For the current version of the code below (main and foo) show the contents of stacks s, and tmp at each of the points marked (i, ii, iii). Show your stacks so that the stack top is next to the variable name, and the top-to-bottom of the stack is shown left-to-right. Also make sure empty stacks are labeled as such. E.g., a stack named blob with 4 elements, such that 32 is the top element, might have the following contents:

blob: 32 99 12 5

public static void int main(String[] args) {
    Stack s = new Stack();
    s.push(7);
    s.push(4);
    s.push(11);
    Stack t; // only used in part B
    t = foo(s);
    return 0;
}
public static Stack foo(Stack s) {
    // current version does not return anything interesting

    Stack tmp = new Stack();
    // i. s: tmp:
    while (!s.isEmpty()) {
        tmp.push(s.top());
        s.pop();
    }
    // ii. s: tmp:
    while (!tmp.isEmpty()) {
        s.push(tmp.top());
        tmp.pop();
    }
    // iii. s: tmp:
}

Part B. Make minimal changes to function foo so that it returns a copy of s without using clone. It must work for any Stack s, not just the one shown in this example. Write your changes right in the code above.

Problem 9. Consider the use of a stack for storing information about function activations as a program runs. As discussed in lecture this is called the run-time stack or call stack. Reminder: a stack has the operations push, pop, top, and isEmpty.

Part A. In this application calling a function corresponds to which stack operation?

Part B. In this application returning from a function corresponds to which stack operation?

Part C. Consider the following program and suppose we are currently executing at the point labeled **C**. Show two different possible contents of the run-time stack at this point (Note there are more than two correct answers). Make it clear which direction the stack is going by labeling which end is the top of each stack you draw.

int main() {
    X();
    Y();
    Z();
    return 0;
}

void X() {
    Y();

Problem 10.
Part A. The code in part B below works, but accomplishes its task in a less than ideal way. What is the worst case big-O time for \texttt{addCounts} in terms of \texttt{scores.size()} and \texttt{maxScore}?

Part B. Modify the \texttt{addCounts} function below to improve the big-O running time. You can make your modifications right in the code below (you don’t need to rewrite the whole function).

```java
// computes the frequency of each score in ‘scores’
// and returns the result in an array
// PRE: scores are all between [0,maxScore] inclusive
public static int[] addCounts(ArrayList<Integer> scores, int maxScore) {
    int[] counts = new int[maxScore+1];
    for (int i = 0; i < scores.size(); i++) {
        for (int score = 0; score <= maxScore; score++) {
            if (scores[i] == score) {
                counts[score]++;
            }
        }
    }
    return counts;
}
```

Part C. What is the worst case big-O time for your new version of \texttt{addCounts}?

Problem 11. Put the following big-O times in order from slowest growing to fastest growing: \(O(n), \ O(n^2), \ O(\log n), \ O(2^n), \ O(1), \ O(n\log n)\)
**Problem 12.** Show what's in the queue after the following sequence of calls. (Make sure to identify where the front of your queue is.) Also show what the code prints. (This class uses standard queue operation names, rather than Java ones)

```java
Queue<Integer> q = new Queue<Integer>();
q.enqueue(3);
q.enqueue(4);
q.dequeue();
q.enqueue(7);
q.enqueue(10);
System.out.println(q.front());
q.dequeue();
q.enqueue(5);
q.enqueue(9);
q.dequeue();
// what’s in the queue here?
```

**Problem 13.**

**Description of AssocList:** Consider a new class AssocList, which associates sequential ints with unique strings. If we add a new string to the AssocList it associates it with the next unused number, starting at 0 for the first one. The main other operations are to find the associated number, given a string, and find the associated string, given a number.

For example, for a Date class, that needs to accept and/or print months in name or number form, it might be useful to associate month numbers with month names, such as in the following code:

```java
AssocList monthList = new AssocList();
monthList.addName("dummy"); // is associated with 0:
    // (a trick so real month number sequence starts at 1)
monthList.addName("January"); // is associated with 1
monthList.addName("February"); // is associated with 2
... 
monthList.addName("December"); // is associated with 12
... 
cout << "November is month number " << monthList.getNumber("November");
cout << endl << "Month 3 is called " << monthList.getName(3);
```

**Your task:** In this problem you are going to implement the complete AssocList class. We have given you the interface below, which documents what each function does. For full credit, your implementation must meet the requirement that each of the three member functions addName, getName and getNumber can be done in $O(\log n)$ time or better, where n is the number of associations (i.e., name-number pairs) in the AssocList. You are encouraged to use the Java library however you wish to help you – several Java containers and algorithms are described on the code handout.
Hint: to meet the time requirement above, you are allowed to use more space (i.e., memory) than you might otherwise need.

The interface (you need to add the instance variable(s) and method implementations):

```java
class AssocList {
    . . .
    // creates an empty AssocList
    public AssocList() . . .

    // number of associations
    public int size() . . .

    // make a new association,
    // and return the number to be associated with name
    // if this name is already in AssocList, returns its old association
    // (i.e., does not change AssocList in that case)
    public int addName(String name) . . .

    // find number that goes with name
    // or -1 if name is not in AssocList
    public int getNumber(String name) . . .

    // find name that goes with number
    // PRE: 0 <= number < size()
    public String getName(int number) . . .
}
```

**Problem 14.** Each of the following hashing schemes is bad. For each, explain what is wrong with it; there are at least two problems with each of them. Assume that your hash table is an array with indices 0 through HASHSIZE-1. Note: all of the functions return a value in the correct range.

**Hashing scheme 1.** Assume we're using chaining. Finds the hash bucket with the shortest chain, and returns its index

**Hashing scheme 2.** Assume the key is a 10-digit phone number (e.g., 213-740-2279). Uses the first three digits as the hash index then takes modulus to convert it to the range of HASHSIZE.