CS146-Homework-2 Due: September 20 at 11:59PM Name:

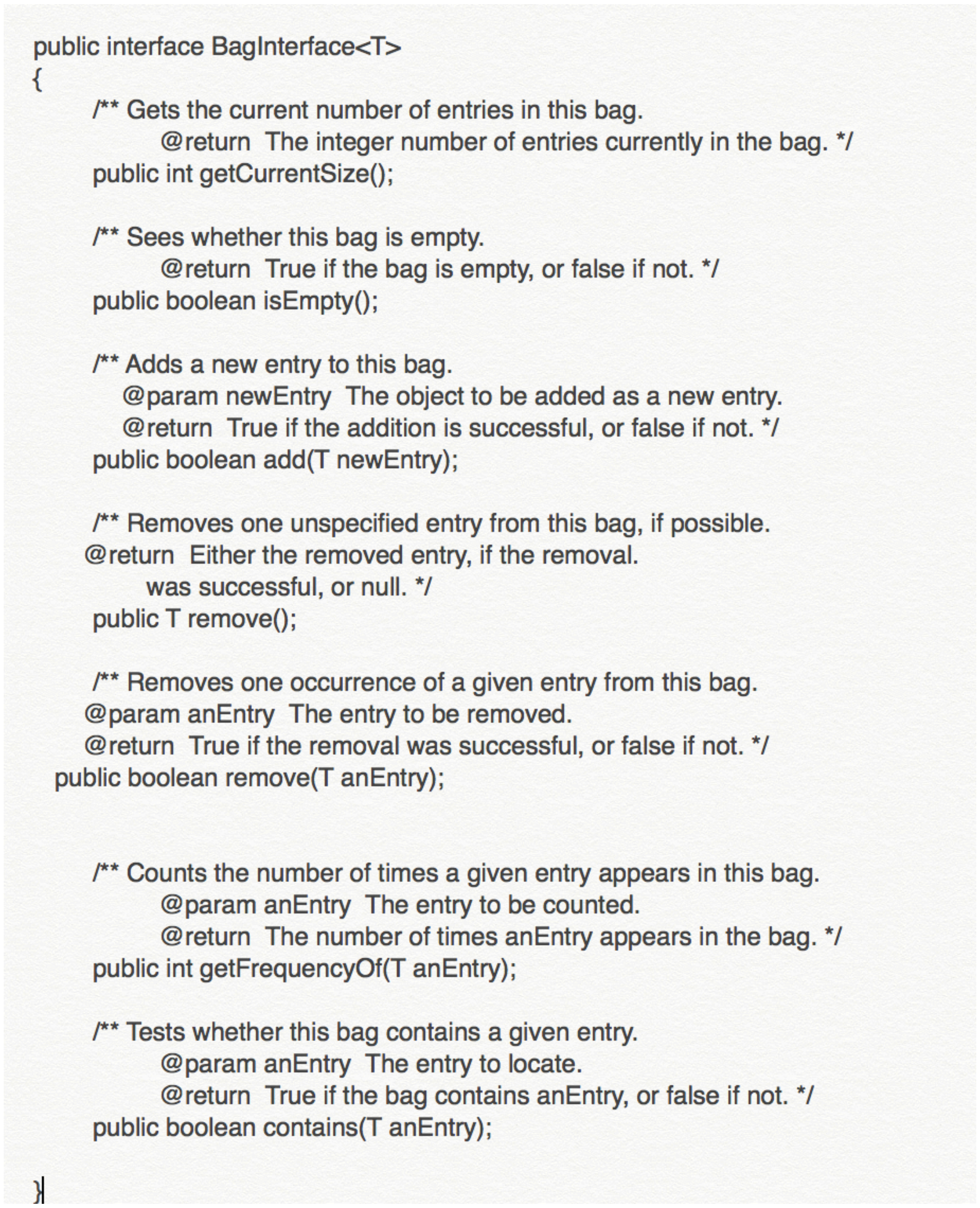
Reading assignments:

Read chapter-3 and class lecture notes.
Submit to Canvas!

If you have any questions about the lab please send the question to answerneededsoon@gmail.com

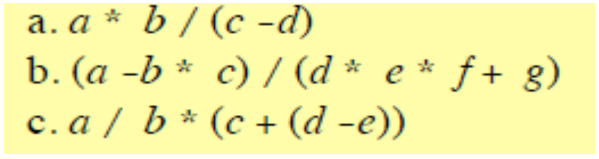
Note that you should have java programs for problems 1 to 6 and have a main to test your code. A copy and paste of all source codes and screen shot of results are needed for these problems. Last problem, you may write it on a paper but you include the picture or scan as part of the word document (you may use snipping tool to do that).

1- Given two sorted lists, L1 and L2, write a procedure to compute intersection of L1 and L2 using only the basic list operations. Test your code!
2- Given two sorted, L1 and L2, write a procedure to compute union of L1 and L2 using only the basic list operations. Test your code!
3- Write a program to evaluate a postfix expression. Assume the function evaluates a postfix expression, using + ,–, * and /ending in =. Test your code!
4- Write routines to implement two stacks using only one array. Your stack routines should not declare an overflow unless every slot in the array is used.
5- You are given a list, L and another list P, containing integers sorted in ascending order. The operation generate(L, P) will create the list Q containing the elements in L that are in positions specified by P. For instance, if P = 1, 3, 4, 6, the elements in positions 1, 3, 4, and 6 in L will be in the list Q. Write the procedure generate(L, P) that returns the list Q. You may use only the public Collection API container operations.
6- Modify the Bag class we did in class to implement the following interface.



```
public interface BagInterface<T>
{
      /** Gets the current number of entries in this bag.
            @return  The integer number of entries currently in the bag. */
      public int getCurrentSize();

      /** Sees whether this bag is empty.
            @return  True if the bag is empty, or false if not. */
      public boolean isEmpty();

      /** Adds a new entry to this bag.
         @param newEntry  The object to be added as a new entry.
         @return  True if the addition is successful, or false if not. */
      public boolean add(T newEntry);

      /** Removes one unspecified entry from this bag, if possible.
     @return  Either the removed entry, if the removal.
           was successful, or null. */
      public T remove();

      /** Removes one occurrence of a given entry from this bag.
     @param anEntry  The entry to be removed.
     @return  True if the removal was successful, or false if not. */
  public boolean remove(T anEntry);


      /** Counts the number of times a given entry appears in this bag.
            @param anEntry  The entry to be counted.
            @return  The number of times anEntry appears in the bag. */
      public int getFrequencyOf(T anEntry);

      /** Tests whether this bag contains a given entry.
            @param anEntry  The entry to locate.
            @return  True if the bag contains anEntry, or false if not. */
      public boolean contains(T anEntry);

}
```

7- Using the algorithm convertToPostfix given in lecture, convert each of the following infix expressions to postfix expressions. Use the stack technique but check your work using pencil and paper technique as well as stack technique.



a. $a * b / (c - d)$
b. $(a - b * c) / (d * e * f + g)$
c. $a / b * (c + (d - e))$