CS 0445 Data Structures
Midterm-exam Practice Questions

Below are some examples of the **types** of questions that **may** appear on the midterm exam. I recommend trying all of the questions before looking at the answer key. Please remember that these are just examples, and that the actual number and type of questions on the exam will not match those here. To adequately prepare for the exam you must study everything indicated in the study guide.

Fill in the Blanks

1) Abstract Data Types (ADTs) can be thought of as an encapsulation of _____ and _____.

2) The two primary techniques for building a new class in Java are through _____ (use components from previously defined classes) and _____ (extend a previously defined class).

3) If a method in a superclass is redefined in a subclass using the identical method signature, we say that we are _____ the method.

4) The two ADT Bag implementations that we discussed in detail had _____ and _____ as their underlying data implementations.

5) A program that executes $4N^2 + 100N + 10000$ operations for a problem of size N has a Big-O runtime of _____.

6) In the ArrayBag class the add() method has a worst case run-time of _____ and in the LinkedBag class the toArray() method has a worst case run-time of _____.

7) Any recursive method needs one or more _____ and one or more _____.

8) Any recursive method that is _____ can easily be converted into a nonrecursive equivalent method.

True/False (explain why false answers are false)

1) Java primitive types can be stored in ArrayBags

2) Assuming the BagInterface<T> interface and LinkedBag<T> class as we discussed, the following statement is legal:
```
BagInterface<String> L = new LinkedBag<String>();
```

3) Assuming the BagInterface<T> interface as we discussed, the following statement is legal:
```
BagInterface<String> L = new BagInterface<String>();
```

4) Binary Search has a worst case run-time of O(N).

5) The method add(newEntry) is **more efficient** with an LinkedBag implementation than with an ArrayBag implementation.

6) Recursion is implemented utilizing **activation records** and the **run-time queue (RTQ).**

Short Answers

1) In class we discussed two variations for resizing a dynamic array list:
   a) Increase the size of the array by 1
   b) Double the size of the array

   Explain which of these is preferable and why. Be specific, using mathematical justification. You do not have to give all of the mathematical details for b) but you must give the general idea.

2) In implementing our LinkedBag class the text authors made the Node class an **inner class**. Explain what an inner class is and why the Node class was implemented in this way.

Traces

1) Give ALL output produced by the execution of the Java program below. To avoid ambiguity, clearly mark your output by **drawing a box around it.**

```
public class PracTrace1
{
      int data;

      public PracTrace1(int d)
        { data = d; }

      public void change(int newdata)
        { data = newdata;       }

      public String toString()
      {
           return new String("Data: " + data);
      }

      public static void main(String [] args)
      {
           PracTrace1 [] A1 = new PracTrace1[4];
           A1[0] = new PracTrace1(10);
           A1[1] = new PracTrace1(20);
           A1[2] = new PracTrace1(30);
           A1[3] = new PracTrace1(40);

           showData(A1);

           PracTrace1 [] A2 = A1;
          PracTrace1 [] A3 = new PracTrace1[A1.length];
          for (int i = 0; i < A1.length; i++)
                A3[i] = A1[i];

           A2[1].change(25);
           A2[2] = new PracTrace1(35);
           A1[2].change(77);

           A3[0].change(15);
           A3[2].change(88);
           A3[3] = new PracTrace1(45);

           showData(A1);
           showData(A2);
```

```
            showData(A3);
        }

        public static void showData(PracTrace1 [] A)
        {
          for (int i = 0; i < A.length; i++)
             System.out.println(A[i]);
           System.out.println();
        }

}
```

2) Give ALL output produced by the execution of the Java program below. To avoid ambiguity, clearly mark your output by **drawing a box around it.**

```
public class PracTrace
{
      static int [] A = {50, 40, 80, 60, 90};
      int workingSum = 0;

      public PracTrace()
      {
            workingSum = 0;
            int sum = recGetSum(A, 0);
            System.out.println("The final working sum is " + workingSum);
            System.out.println("The overall sum is " + sum);
      }

      public int recGetSum(int [] A, int loc)
      {
            if (loc < A.length)
            {
                  workingSum = workingSum + A[loc];
                  System.out.println("loc = " + loc + " working sum = " +
                                     workingSum);
                  int theSum = A[loc] + recGetSum(A, loc + 1);
                  workingSum = workingSum - A[loc];
                  System.out.println("loc = " + loc + " working sum = " +
                                     workingSum);
                  return theSum;
            }
            else
               return 0;
      }

      public static void main(String [] args)
      {
            PracTrace P = new PracTrace();
      }
}
```

Coding: Make sure you are familiar with both array-based implementations and linked implementations of the ADTs that we have discussed. Also, consider anything in your programming assignments to be material that could be on the exam. Below are a couple of example questions.

1) A method that is useful in the ArrayBag class is the method
   `private int getIndexOf(T anEntry)`

This method iterates through the array and returns the index of the first location whose data matches **`anEntry`**. If **anEntry** is not found the method should return -1. Complete this method below. Recall that the ArrayBag class has two instance variables:

```
private T [] bag;
private int numberOfEntries;
```

2) A method that is useful in the LinkedBag class is the method
`private Node getReferenceTo(T anEntry)`
This method iterates through the list and returns a reference to the first Node whose data matches **`anEntry`**. If **`anEntry`** is not found the method should return null. Complete this method below. Recall some important declarations in the LinkedBag class:

```
private Node firstNode;
private int numberOfEntries;

 private class Node
 {
   private T data;
   private Node next;
   // methods omitted
 }
```