Midterm On-Your-Own Practice Questions

These questions are presented for practice. You are not required to post an answer.

Question 1. Bag Trace

Trace the contents of the bag (implements BagInterface) after each statement:

```
System.out.println(nameBag.isEmpty());
nameBag.add("adam");
nameBag.add("brian");
nameBag.add("carl");
nameBag.add("adam");
nameBag.add("fred");
nameBag.add("carl");
nameBag.add("harry");
nameBag.add("hank");
System.out.println(nameBag.remove("adam"));
System.out.println(nameBag.getCurrentSize());
System.out.println(nameBag.remove("adam"));
System.out.println(nameBag.remove("adam"));
System.out.println(nameBag.remove("ivan"));
System.out.println(nameBag.getCurrentSize());
System.out.println(nameBag.getFrequencyOf("carl"));
System.out.println(nameBag.contains("ivan"));
System.out.println(nameBag.getFrequencyOf("ivan"));
nameBag.clear();
System.out.println(nameBag.getCurrentSize());
```

Question 2. ListInterface Trace

Trace the contents of the list (implements ListInterface) after each statement:

```
System.out.println(nameList.isEmpty());
nameList.add("adam");
nameList.add("brian");
nameList.add("carl");
nameList.add("edgar");
nameList.add(3, "hank");
nameList.add("lenny");
nameList.add(1, "mark");
System.out.println(nameList.getLength());
System.out.println(nameList.getEntry(3));
```

```
System.out.println(nameList.remove(2));
System.out.println(nameList.getEntry(2));
System.out.println(nameList.remove(1));
System.out.println(nameList.remove(2));
System.out.println(nameList.getLength());
System.out.println(nameList.replace(2, "peter"));
System.out.println(nameList.getEntry(2));
System.out.println(nameList.getEntry(1));
System.out.println(nameList.getLength());
```

Question 3. List Trace

Trace the contents of the list (implements the Java List interface) after each statement:

```
System.out.println(nameList.isEmpty());
nameList.add("adam");
nameList.add("brian");
nameList.add("carl");
nameList.add("edgar");
nameList.add(3, "hank");
nameList.add("lenny");
nameList.add(1, "mark");
System.out.println(nameList.size());
System.out.println(nameList.get(3));
System.out.println(nameList.remove(2));
System.out.println(nameList.get(2));
System.out.println(nameList.remove(1));
System.out.println(nameList.remove(2));
System.out.println(nameList.size());
System.out.println(nameList.set(2, "peter"));
System.out.println(nameList.get(2));
System.out.println(nameList.get(1));
```

Question 4. Client Perspective on Bags and Lists

1. What does it mean to use a bag or list as a client (to use an interface)? What can you access?
2. Complete the following without using the toArray method. Write code that is as efficient as possible.

   a. Write code at the client level that merges two lists by alternating the elements together. The method takes two ListInterface objects as parameters and returns the new ListInterface. For example: List1 = {1, 2, 3}, List2 = {4, 6, 8, 10}, return a new list: {1, 4, 2, 6, 3, 8, 10}. The original two lists should not be modified when your program ends.

   b. Write code at the client level that merges two bags. The original two bags should not be modified when your program ends.

c. Write a containsDuplicates method for lists from the client perspective. The method takes a ListInterface as a parameter. The list should not be modified when the method ends.

d. Assume there is no getCurrentSize() method in BagInterface. Write a getSize method for bags from the client perspective. The method takes a bag as a parameter. The bag should not be modified when the method ends.

Question 5. Implementation Perspective on Bags and Lists

Complete the following without using the toArray method. Write code that is as efficient as possible.

a. Write a getFrequencyOf method for AList or LList. The method should be efficient because you can access the underlying data structure. (Do not just use the same method as you would use from the client perspective.)

b. Assume there is no getCurrentSize() method and no direct access to numberOfEntries. Write a getSize method for ArrayBag or LinkedBag. The method should be efficient because you can access the underlying data structure. (Do not just use the same method as you would use from the client perspective.)

Question 6: Tracing Nodes A

1. What is printed by the code below?
2. Explain what the mystery method does. (Do not repeat what the code does, but explain in words what it returns.)
3. Is there anything wrong with how this method is implemented? Will it ever crash?

```
list: 4 -> 6 -> 10 -> 12

Node nodeA = list.firstNode.next.next;
Node nodeB = list.firstNode.next;
Node nodeC = list.firstNode.next.next.next;
Node nodeD = nodeC.next;

System.out.println(nodeA.data);
System.out.println(nodeB.data);
System.out.println(nodeC.data);
System.out.println(nodeD.data);
System.out.println(mystery(list.firstNode, 9));
System.out.println(mystery(list.firstNode, 10));

public boolean mystery(Node firstNode, int target) {
   Node currentNode = firstNode;
   if(currentNode.next==null) {
      return false;
   } else {
```

```
            Node tmpNode = currentNode;
            currentNode = currentNode.next;
            while(currentNode!=null) {
               if(currentNode.data==target) {
                  System.out.println(tmpNode.data);
                  return true;
               } else {
                  tmpNode = currentNode;
                  currentNode = currentNode.next;
               }
            }
            return false;
         }
      }
```

Question 7: Tracing Nodes B

What is printed by the pseudocode below?

```
Node firstNode = new Node(3);
firstNode.next = new Node(4);
firstNode.next.next = new Node(6);
firstNode.next.next.next new Node(8);
Node currentNode = firstNode;

print currentNode.data
print the chain headed by firstNode

currentNode = currentNode.next
print currentNode.data
print the chain headed by firstNode

currentNode.data = 7
print currentNode.data
print the chain headed by firstNode

currentNode.next = currentNode.next.next;
print currentNode.data
print the chain headed by firstNode

firstNode = firstNode.next;
print currentNode.data
print the chain headed by firstNode
```

Question 8: Other Questions about Bags and Lists

a. What are the underlying data structure and other instance data variables for each of the four classes (ArrayBag, LinkedBag, AList, and LList)? In general, how are these instance data

variables affected by adds/removes (meaning where do adds/removes happen)?

b. What are the efficiencies of adding and removing for the two bag classes?

c. What are the efficiencies of adding to the front, adding to the back, removing from the front, and removing from the back for the two list classes? Is there anything you can do to improve the complexity of any of these methods for LList?

d. What are the advantages and disadvantages of implementing lists and bags with arrays vs linked nodes?

e. What special considerations must be taken into account for empty bags and lists? Are these the same considerations for array-based and node-based implementations? Are there things we need to think about for "full" data structures?

Question 9: Coding with Nodes

a. Write a complete method to traverse a linked list of Strings, headed by firstNode, and determine if each String contains a specific character.

b. Write a complete method to determine if a linked list of Integers, headed by firstNode, contains any two values repeated. For example, 4->2->2->3 would return true and 3->2->4->2 would return false.

c. Write a complete method that takes in a linked list of Integers (headed by firstNode) and an int array and merges them together into an array using an alternating merge (see Question 4 above).

Question 10: Comparison

a. When should you use == vs. equals?

b. Are there restrictions on what kind of objects can use equals?

c. When should you use compareTo vs < or >?

d. Are there restrictions on what kind of objects can use compareTo?

e. What values are returned from compareTo?

Question 11: Calculating Complexity

a. What is the relationship between an algorithm's running time and its big-oh order of growth?

b. What are considered some "good" Big-Os and some "bad" Big-Os?

c. What are the Big-Os of the following algorithms?

Algorithm 1:

```
statement1;

if(condition1) {
   statement2;
} else {
   for(int i=0; i<n; i++) {
      statement3;
   }
   statement4;
}
```

Algorithm 2:

```
i=0;
while(i<n) {
   for(int j=i; j<n; j++) {
      statement1;
   }
   statement2;
   i++;
}
```

Algorithm 3:

```
for(int i=0; i<n; i++) {
   for(int j=0; j<n; j++) {
      if(condition1) {
         for(int k=0; k<10; k++) {
           statement1;
         }
      } else {
         statement2;
      }
   }
}
```

Algorithm 4:

```
for(int i=0; i<=n; i++) {
   for(int j=0; j<=n; j++) {
      if(j%2==0) {
         statement1
      }
   }
}
```

d. What are the Big-Os of the following algorithms?

Algortihm A:

```
for(int i=0; i<n; i++)
   add i to the beginning of an array-based list
```

Algortihm B:

```
for(int i=0; i<n; i++)
   add i to the end of an array-based list
```

Algortihm C:

```
for(int i=0; i<n; i++)
   add i to the beginning of a linked list with only a head point
er
```

Algortihm D

```
for(int i=0; i<n; i++)
   add i to the end of a linked list with only a head pointer
```

Click for a Selection of Answers and Coding Examples (https://ccsf.instructure.com/courses/47904/pages/midterm-on-your-own-practice-questions-selected-answers-and-examples)