Homework M7: Recursion

Due Mar 21 at 11:59pm **Points** 100 **Questions** 33
Available until May 30 at 11:59pm **Time Limit** None

Instructions

Review the **Homework FAQ** page for details about submitting homework. In this homework, you will:

- trace recursion
- write recursive methods

Homework Files

Below is the driver/tester program. I strongly recommend using this to test your code before submitting. You can ignore the *test methods* at the end of the file.

HomeworkM7Files.zip (https://ccsf.instructure.com/courses/47904/files/7497792/download?download_frd=1)

Notes

For Full Credit:

- **Use recursion. Non-recursive methods will receive 0 points.**
- **Do not include static variables** to make the methods work. Use only passed parameters and local variables declared inside the method.
- Consider efficiency. When possible, use a helper method with extra parameters instead of destroying/rebuilding or copying the data.

Helper methods ARE allowed!

- When writing a recursive solution, you can **always** include additional private helper methods if you need extra parameters.
- When you use a helper method, the original method might not be recursive but the helper method is. That's okay because the *solution is recursive*.
- Make sure you submit the helper method **and** the original method that invokes the helper method (so we can see the initial values used to invoke the method).

Be careful with valued recursive methods!

- Make sure you link the recursive calls together with return statements or local variables!

This quiz was locked May 30 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	4,071 minutes	106 out of 100

Score for this quiz: **106** out of 100
Submitted Mar 17 at 8:36pm
This attempt took 4,071 minutes.

Coding Questions

Question 1 — 15 / 15 pts

Write a **recursive** void method that reverses the contents of an array. Do **not** create a new array- reverse the contents of the current array. (Note the void return type.)

The method header is:

```
public static void arrayReverse(int[] array)
```

Your Answer:

```
// Question 1 homework
public static void arrayReverse(int[] array) {
    int start = 0;
    int end = array.length - 1;

    // recursive helper call if list has content
    if (array.length > 1) {
        arrayReverseHelper(array, start, end);
    }
```

```
}

// helper for arrayReverse
private static void arrayReverseHelper(int[] array, int start, int end) {
    if (start <= end) {
        int temp = array[start];
        array[start] = array[end];
        array[end] = temp;

        // recursive call
        arrayReverseHelper(array, start + 1, end - 1);
    }
}
```

Question 2

20 / 20 pts

Write a recursive method to determine whether a String contains a 'q' **not immediately followed by** a 'u' (ignoring capitalization). In other words:

- the word **does** contain at least one 'q'
- and that q is followed by **anything except** a 'u'

Carefully review the provided driver program to see example test cases. The method header is:

```
public static boolean qNotFollowedByU(String word)
```

Your Answer:

```
// Question 2 homework
public static boolean qNotFollowedByU(String word) {
    if (word.length() == 1 && (word.contains("q") || word.contains("Q"))) {
        return true;
    } else if (word.length() > 1 && (word.contains("q") || word.contains("
Q"))) {
        return qNotFollowedByUHelper(word);
    }
    return false;
}

private static boolean qNotFollowedByUHelper(String word) {
    boolean qFirstNoU = false;
    int index = 0;

    // check for q // check for u
    if (word.toLowerCase().charAt(0) == 'q' && word.toLowerCase().charAt(1)
!= 'u') { // check for q after u
        qFirstNoU = true;
    } else if (word.toLowerCase().charAt(word.length() - 1) == 'q') {
```

```
        qFirstNoU = true;
    }

    if ((index < word.length() - 2) && qFirstNoU == false) {
        qFirstNoU = qFirstNoU || qNotFollowedByUHelper(word.substring(index
+ 1));
    }
    return qFirstNoU;
}
```

Question 3

10 / 10 pts

Rewrite the getFrequencyOf method in the LinkedBag class using recursion. The original, non-recursive version is shown below for reference.

For full credit, do **not** invoke the toArray method. Directly use the node chain.

Your Answer:

```
public int getFrequencyOf(T anEntry) {
    Node currentNode = this.firstNode;

    return getFrequencyOfHelper(currentNode, anEntry);
}

private int getFrequencyOfHelper(Node node, T anEntry) {
    int anEntryCounter = 0;

    if (node != null) {

        // checks condition for a match
        if (node.data.equals(anEntry)) {
            anEntryCounter++;
        }
        // recursive call and store
        anEntryCounter = anEntryCounter + getFrequencyOfHelper(node.next, a
nEntry);
    }
    return anEntryCounter;
}
```

Non-recursive method in LinkedBag:

```
public int getFrequencyOf(T anEntry) {
   int frequency = 0;
   int counter = 0;
   Node currentNode = firstNode;

   while ((counter < numberOfEntries) && (currentNode !
= null)) {
      if (anEntry.equals(currentNode.data)) {
         frequency++;
      }
      counter++;
      currentNode = currentNode.next;
    }
   return frequency;
}
```

Multiple Choice Questions

Question 4

2 / 2 pts

A recursive solution can always be rewritten with iteration.

Correct!

- (●) True
- ( ) False

For the next set of questions, use the following method.

```
public int methodA(int n) {
```

```
    if(n==0) {
       return 0;
    } else if(n>0) {
       return 1 + methodA(n-1);
    } else {
       return 1 + methodA(n+1);
    }
}
```

Question 5

2 / 2 pts

What is the value returned from invoking methodA(4)?

Correct!

4

Correct Answers

4 (with margin: 0)

Question 6

2 / 2 pts

What is the value returned from invoking methodA(0)?

Correct!

0

Correct Answers

0 (with margin: 0)

Question 7

2 / 2 pts

What is the value returned from invoking methodA(-3)?

Correct!

3

Correct Answers

3 (with margin: 0)

For the next set of questions, use the following method.

```
public int methodB(String s, char c) {
   if(s.length()==0) {
      return 0;
   } else {
      return (s.charAt(0)==c ? 1 : 0) + methodB(s.subst
ring(1), c);
   }
}
```

Question 8

2 / 2 pts

What is the value returned from invoking methodB("eieio",'e')?

Correct!

2

Correct Answers

2 (with margin: 0)

Question 9

2 / 2 pts

What is the value returned from invoking methodB("hello there",'h')?

Correct!

2

orrect Answers

2 (with margin: 0)

Question 10

2 / 2 pts

The following method is designed to count the number of even numbers in an array.

```
public int countEvens(int[] array) {
   return countEvensHelper(array, 0, array.length);
}

private int countEvensHelper(int[] array, int start, in
t stop) {
   int count = 0;
   if(start==stop) {
      return count;
   } else {
      if(array[start] % 2 == 0) {
         count = 1 + countEvensHelper(array, start+1, s
top);
      }
      return count;
   }
}
```

Which of the following is true if the method is invoked with a non-empty array?

- The code is correct.
- There is a compiler (syntax) error.
- none of these is correct
- The code will run, but then will either crash or go into infinite recursion.

Correct!

- The code will run, but there is a logical error.

The next set of questions asks about the following recursive method, described in pseudocode.

I recommend you trace out this method and have the output on hand to answer the questions.

```
public void recMethod(int[] array, int start, int end)
{

   if(start < end) {
      print the array

      double the value in the array at position start

      recMethod(array, start+1, end)

      print the array

   } else {
      print "done"
   }
}
```

Assume the method is invoked with array = [3, 4, 6, 7, 8, 10, 4], start = 2, and end = 5.

Question 11

2 / 2 pts

How many times is the array printed **before** the word "done" is printed?

Correct!

3

orrect Answers

3 (with margin: 0)

Question 12

2 / 2 pts

How many times is the array printed **after** the word "done" is printed?

Correct!

3

orrect Answers

3 (with margin: 0)

Question 13

2 / 2 pts

Which of the following is true?

- The contents printed in the line directly before the "done" are the same as the contents printed in the line directly after the "done."

Correct!

- The contents printed in the line directly before the "done" are different from the contents printed in the line directly after the "done."
- There is nothing printed out after the "done."

Question 14

2 / 2 pts

Which of the following is true?

Correct!

- (selected) All array print outs after the "done" show the same contents as each other.
- The array print outs after the "done" are not all the same.
- There are no array print outs after the "done."

Question 15

0 / 2 pts

Assume that the method is invoked with array = [1, 4, 6, 3, 2, 7, 8], start = 3, and end = 2.

Which of the following is true?

- there will be 0 lines of output (and no exceptions)
- there will be 2 lines of output (and no exceptions)

orrect Answer

- there will be 1 line of output (and no exceptions)

ou Answered

- (selected) none of these is correct
- an exception or error is thrown

Question 16

2 / 2 pts

Assume that the method is invoked with array = [1, 4, 6, 3, 2, 7, 8], start = 3, and end = 8.

Which of the following is true?

- there will be 1 line of output (and no exceptions)
- there will be 0 lines of output (and no exceptions)
- there will be 2 lines of output (and no exceptions)
- none of these is correct

Correct!

- an exception or error is thrown

The next set of questions asks about the following recursive method, described in pseudocode.

I recommend you trace out this method and have the output on hand to answer the questions.

```
public void recMethod(Node firstNode) {
   print the chain of nodes headed by firstNode
   if(firstNode.next!=null) {
      double the data in firstNode
      recMethod(firstNode.next.next);
   } else {
      print "done"
   }
   print the chain of nodes headed by firstNode
}
```

Assume the method is invoked with the firstNode of this chain: 4 -> 6 -> 5 -> 3 -> 2

Question 17

2 / 2 pts

How many times is the chain of nodes printed **before** the word "done" is printed?

Correct!

3

orrect Answers

3 (with margin: 0)

Question 18

2 / 2 pts

How many times is the chain of nodes printed **after** the word "done" is printed?

Correct!

3

orrect Answers

3 (with margin: 0)

Question 19

2 / 2 pts

Which of the following is true?

Correct!

- (selected) The print outs of the chain before the "done" each have a different number of values displayed.
- The print outs of the chain before the "done" all have the same number of values displayed.
- There are no print outs of the chain before the "done."

Question 20

2 / 2 pts

Which of the following is true?

- The print outs of the chain after the "done" all have the same number of values displayed.

Correct!

- (selected) The print outs of the chain after the "done" each have a different number of values displayed.
- There are no print outs of the chain after the "done."

Question 21

2 / 2 pts

The first value of the first print is the same as the first value of the last print.

- True

Correct!

- (●) False

Question 22

2 / 2 pts

Assume that the method is invoked with the firstNode of this chain: 5 (a singleton chain).

Which of the following is true?

- ( ) an exception or error is thrown
- ( ) there will be more than 3 lines of output (and no exceptions)

Correct!

- (●) there will be 3 lines of output (and no exceptions)
- ( ) there will be less than 3 lines of output (and no exceptions)

Question 23

2 / 2 pts

Assume that the method is invoked with an empty chain.

Which of the following is true?

- there will be less than 2 lines of output (and no exceptions)
- there will be 2 lines of output (and no exceptions)

Correct!

- (selected) an exception or error is thrown
- there will be more than 2 lines of output (and no exceptions)

Question 24

1 / 1 pts

Code that contains an infinite recursion will result in

- a problem at compile time.

Correct!

- (selected) a problem at runtime.

Unfortunately, the compiler does not detect infinite recursion!

Question 25

2 / 2 pts

Why might the following method have infinite recursion?

```
public int infiniteRecursion(int n) {
   if (n > 0) {
      return infiniteRecursion(n)   -  2;
```

```
    } else {
       return 0;
    }
}
```

- Because there is no base case
- Because the base case will never be true

Correct!

- (selected) Because the recursive call does not move the parameter closer to the base case
- None of these are correct, there is no infinite recursion in this method

Question 26

2 / 2 pts

Why might the following method have infinite recursion?

```
public void infiniteRecursion(int n) {
   if (n > 0) {
      System.out.println("here");
      infiniteRecursion(n-1);
   } else if(n < 0){
      System.out.println("here");
      infiniteRecursion(n+1);
   } else {
      System.out.println("here");
   }
}
```

- Because the base case will never be true

Correct!

- (selected) None of these is correct, there is no infinite recursion in this method
-

Because the recursive call does not move the parameter closer to the base case

Because there is no base case

Question 27

0 / 2 pts

What condition of the parameters will cause the following method to have infinite recursion?

```
public int secretMethod(int x, int y) {
   if (x == y) {
      return 1;
   } else {
      return secretMethod(x-1, y) + 1;
   }
}
```

x == y

orrect Answer x < y

none of these is correct, there will be no infinite recursion in this method

ou Answered x > y

Question 28

1 / 1 pts

The following method is missing a base case.

```
public int noBaseCasePerhaps(int x) {
   if (x > 0) {
```

```
      return noBaseCasePerhaps(x-1) + 1;
   } else {
      return noBaseCasePerhaps(x-2) + 2;
   }
}
```

Correct!

- (●) True
- ( ) False

Question 29

1 / 1 pts

The following two methods will both compute the same thing when invoked with the same value of x. That is, method1(x) == method2(x).

```
public int method1(int x) {
   if (x > 0) {
      return method1(x-1) + 1;
   } else {
      return 0;
   }
}

public int method2(int x) {
   if (x <= 0) {
      return 0;
   } else {
      return 1 + method2(x-1);
   }
}
```

Correct!

- (●) True
- ( ) False

Question 30

2 / 2 pts

The following method is designed to count the number of odd numbers in an array.

```
public int countOdds(int[] array) {
   return countOddsHelper(array, 0, array.length);
}

private int countOddsHelper(int[] array, int start, int
stop) {
   int count = 0;
   if(start < stop) {
      if(array[start] % 2 == 1) {
         count++;
      }
     countOddsHelper(array, start+1, stop);
   }
   return count;
}
```

Which of the following is true if the method is invoked with a non-empty array?

- The code will run, but then will either crash or go into infinite recursion.
- none of these is correct
- There is a compiler (syntax) error.
- The code is correct.

Correct!

- The code will run, but there is a logical error.

Question 31

2 / 2 pts

The following method will return true if the int parameter is even and non-negative (meaning positive or 0), and false otherwise. For example, invoking with 0, 2, 4, 6, or 8 will return true. Invoking with -4, -3, -2, -1, 1, 3, 5, or 7 will return false.

Which set of code should be added so that the method works appropriately?

```
public boolean evenPosZero(int x)  {
   if(x<0)
      return false;

   // ??? what goes here

}
```

else if (x==0) return false;
else return evenPosZero(x – 1);

none of these is correct

else return(x==0);

Correct!

else if (x== 0) return true;
else return evenPosZero(x – 2);

there is more than one answer that will work correctly

else if (x==0) return true;
else return evenPosZero(x – 1);

Question 32

2 / 2 pts

The following method is designed to count the number of zeros in an

array.

```
public int countZeros(int[] array) {
 return countZerosHelper(array, 0, array.length, 0);
}

private int countZerosHelper(int[] array, int start, in
t stop, int count) {
 if(start < stop) {
 if(array[start] == 0) {
 count++;
 }
 countZerosHelper(array, start+1, stop, count);
 }
 return count;
}
```

Which of the following is true if the method is invoked with a non-empty array?

- The code will run, but then will either crash or go into infinite recursion.
- The code is correct.
- There is a compiler (syntax) error.

Correct!

- (selected) The code will run, but there is a logical error.
- The code will run, but then will either crash or go into infinite recursion.

Optional Extra Credit: 10 points

Write a recursive method that counts the number of positive integers in a bag from the client perspective.

- The bag should **not** be changed as a result of the method call.
- For full credit, do **not:**

- create a duplicate copy of the bag
- invoke the toArray method
- create a temporary data structure

- Hint: think about how you can add code both before and after the recursive call!

The method header is

```
public static int countPositives(BagInterface<Integ
er> bag)
```

Question 33

10 / 0 pts

Paste the countPositives method here.

Your Answer:

```
public static int countPositives(BagInterface<Integer> bag) {
    int posIntegerCount = 0;

    if (!bag.isEmpty()) {
        Integer integerFromBag = bag.remove();

        // conditional check for incrementing counter
        if (integerFromBag > 0) {
            posIntegerCount++;
        }
        // recursive call and store
        posIntegerCount = posIntegerCount + countPositives(bag);

        // add Integer back to bag
        bag.add(integerFromBag);
    }
    return posIntegerCount;
}
```

Quiz Score: **106** out of 100