Course: CSC220.01

Student: Mark Kim

Instructor: Duc Ta

Assignment Number: 02

Due Date & Time: 07-13-2019 at 11:55 PM

Assignment 02

PART A – The Bag

The goal of this portion of the assignment was to complete the code provided to remove all occurrences of the letters contained in a two-dimensional array. The purpose of this part of the assignment was to familiarize us with linked abstract data types and how they behave.

```
    public int removeAllOccurences(T[][] entries) {

        // Convert 2D array to 1D array
        ArrayList<T> tmpArray = new ArrayList(convert2dtoldArrayList(e
        // Remove duplicates in array
        removeArrayDuplicates(tmpArray);  // Call helper method to rem
        // Remove all occurences of given items
        int counter = 0;
        if (!isEmpty()) {                                       // Che
            Node currentNode = firstNode;                       // Poi
            T currentEntry;                                     // Ini
            while (currentNode != null) {                       // Ite
                for (int i = 0; i < tmpArray.size(); i++) {     // Loo
                    currentEntry = currentNode.data;            // Ret
                    if (currentEntry.equals(tmpArray.get(i))) { // Che
                        currentNode.data = firstNode.data;      // Mov
                        firstNode = firstNode.next;             // Rem
                        numberOfEntries--;                      // Red
                        counter++;                              // Inc
                    }
                }
                currentNode = currentNode.next;                 // Poi
            }
        }
        return counter; // return a count of the number of entries rem
```

```
asmt02Part01.LinkedBag >  toArray >
utput - CSC220.01_Assignment_02 (run-single)
ant -f C:\\Users\\Mark\\Documents\\NetBeansProjects\\CSC220.01_Assignment_02 -Dnb.
init:
Deleting: C:\Users\Mark\Documents\NetBeansProjects\CSC220.01_Assignment_02\build\b
deps-jar:
Updating property file: C:\Users\Mark\Documents\NetBeansProjects\CSC220.01_Assignm
Compiling 1 source file to C:\Users\Mark\Documents\NetBeansProjects\CSC220.01_Assi
Note: C:\Users\Mark\Documents\NetBeansProjects\CSC220.01_Assignment_02\src\asmt02P
Note: Recompile with -Xlint:unchecked for details.
compile-single:
run-single:
[+] Creating an empty bag...
- The bag now contains 0 string(s):

[+] Creating bag items...
[+] Adding the bag items to the bag:   A    G Bb A   u n o A o d Bb A A l l
- The bag now contains 18 string(s):   l l A A Bb d o A o n u   A Bb G    A

[+] Creating a 2D testArray...
                                      A A A A A A
                                      B Bb Bb
                                      C
                                      n u l l

[+] Removing testArray items from the bag...
- The bag now contains 4 string(s):    G o o d

BUILD SUCCESSFUL (total time: 0 seconds)
```

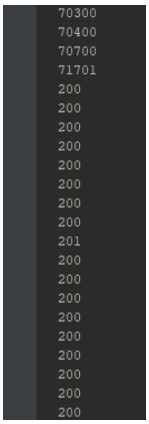
Conceptually, this task was relatively easy to grasp. After initially completing the code, however, I found that the order of the strings did not match the output. I spent several hours trying to figure out why my code was not completing the task correctly. Unfortunately, it required a tip from the instructor for me to solve the problem. The solution was to loop through the test array instead of the linked list by comparing the node to all items in the test array before moving to the next node. The solution seemed so obvious when the instructor gave me the hint.

PART B – The Efficiency of Algorithms

This first part of part B of this assignment was probably the least difficult portion. Although I had some trouble with the worksheet on counting operations at first, after practicing for a little bit, I became better at it. Below is the result for B.1:

```
sumList (aList, n) {
    thisSum = 0;                                1
    lastSum = 5000;                             1
    for (int i = 0; i <= n; i++) {              1 + (n + 2) + 2(n + 1)
        thisSum = thisSum + aList[i] * 2;                 3(n + 1)
    }
    return thisSum >= lastSum;                  2
                                                5 + n + 2 + 5(n + 1) = 6n + 12
```

The second part of B was not difficult in and of itself. First, I separated each loop into its own method so that testing would be easier and the code easier to understand. Unfortunately, after completing the code, the behavior of the methods was erratic. I ended up dismantling the code and tested the loops alone and found some strange behavior. After some iterations, the time of execution reduced to nearly zero. To the left is a screenshot of the console output. Because of this behavior, I almost gave up and just figured out a way to find the values manually. I learned that this behavior was being caused by including both loops in the same "n" loop. After separating each loop into its own "n" loop, the behavior was consistent and predictable.



After figuring this part out, I decided to go several steps further and create a two-dimensional array containing several tests of each loop and finding an average to more accurately estimate the point of intersection.

PART C – Stacks and Stack Implementation

This assignment confused me a little because I was not completely sure how the assignment wanted this to be implemented. Although the OurStack class included an initialized Stack object, I did not use it and simply implemented a Node class within the OurStack class and filled in missing code within each of the methods in the class. Once I completed the OurStack class, the rest was just a matter of comparing each letter of the string to the top of the stack as each item is popped out.

PART D – Recursion

This, along with part E were the most difficult part of the assignment for me. I relied heavily on help from others to complete this assignment. Despite the code outputting correctly, I am still not at all confident with recursion. I am perfectly capable of tracing a recursion, but I am not very capable of implementing it on my own. This is a subject with which I will need much more practice.

I believe some of the difficulty I had with this was that I was trying to develop a method for the recursion to create blurbs whereby the number of letters outputted had equal chance of happening. The method I incorporated in my code creates blurbs where certain numbers of letters have greater chance of occurring than others.

PART E – Recursion, Pascal's Triangle

Believe it or not, I actually had an easier time implementing this program than for part D. Again, I am still not very confident with the recursion portion of this program. Unfortunately, my time was very limited for this assignment, so I did not comment the code nearly as well as I would have liked (especially with the latter portions of the assignment). Nevertheless, I tried to keep the code clean and clear so that it can be easily deciphered.

Zip file contents:

MarkKim-Assignment-02-Report.pdf	(this report)
asmt02Part01	(folder containing all java files for part A)
asmt02Part02	(folder containing all java files for part B)
asmt02Part03	(folder containing all java files for part C)
asmt02Part04	(folder containing all java files for part D)
asmt02Part05	(folder containing all java files for part E)