**CS3460:** Data Structures Fall 2022
**Mid Term**
**Due Date:** Thursday, October 13; 23:00 hrs
**Total Points:** 20

**Honor Code:** The following document is an individual exam. As such, this exam should be solved individually, without any discussion, exchange of electronic or hard documents, or any other malpractice that constitutes to cheating in an academic institution. Even a pleasant conversation that involves revealing how much of the test one completes constitutes a violation. If you are unsure what constitutes plagiarism, please refer to the following guidelines set forth by the Appalachian State University `http://academicintegrity.appstate.edu/`, or consult with your instructor. By continuing to participate in this exam, you implicitly agree to uphold the honor code. Failure to uphold the honor code will result in an **F grade** for the entire course.

Please write well-tested Java programs to solve the following problems. You are allowed to use any code that is posted on ASULearn, or code that you have written for any of the homework problems. You may also use the Internet as a resource for general information, for example, in figuring out the syntax and the semantics of the Java programming language, or obtaining general descriptions of the data structures studied in class. However, you are not allowed to search for specific solutions in the Internet, which includes code forums like GeeksForGeeks, StackOverflow, and other such online platforms.

You may consider to logically split your solutions into several components, and solve each of those components in Java. For example, to use a Hash Table, you may have a class that implements a single Node structure, and a class that implements a hash table that resolves collisions through chaining. For any of the problems, you are allowed to create multiple files that cater to your implementation, but please submit all the necessary Java code for a particular solution, so that they can be run simply by compiling and executing in a Java environment. If you are creating a `Node` class for linked lists, hash tables and binary search trees, consider naming them `ListNode`, `HashNode` and `BSTNode` respectively to avoid confusion during compilation.

Good Luck!

1. **Broken Keyboard (10 points):** Alan Huff is trying to finish his program for his data structures class, but is typing on a broken keyboard. Specifically, the keyboard automatically presses the home ('^') and the end ('$') keys, which makes the cursor move to the beginning and end of the line respectively. Luckily, the file shows when the home and end keys are automatically pressed. Treating the entire input as a giant string[1], please write a program `BrokenKeyboard.java` that takes in a string as input, possibly with home and end keys pressed several times, and prints the reconstructed original text as output.

   Some scaffolding code is provided to get you started. Specifically, the input (got from `System.in`) file is read into a giant string named `text`. Your task is to reconstruct the original text and print (to `System.out`). A sample input and output is provided below.

[1] Newline characters are included in the string.

```
Sample Input (see broken-keyboard-short):

men
I^ egg $ am t^e the$he wa^ey ar$lrus
^an
Th$Goo g^egg m$oo g'^ the $joob
^I am

Sample Output:

I am the egg man
They are the egg men
I am the walrus
Goo goo g'joob
```

The file `broken-keyboard-short` contains the above sample input. You may execute your Java program by redirecting the input:

```
$ java BrokenKeyboard < broken-keyboard-short
```

The file `hip^ows$_of^ell$_th^e_F$e_R^-Th$ing^OTR$` contains over 1 million characters. Your program should work for this large an input. Feel free to build other class files with data structures that are appropriate to your implementation.

2. **Printing Users (10 points):** Our distinguished system administrator, Tim Tron, likes to monitor the student machine for routine maintenance. He also would like to do his Data Structures homework when there are few students logged into the system. He typically runs the following command to print a list of users logged into the system.

```
[trontw@student2 ~]$ who | awk -F' ' '{ print $1}'
trontw
blakelywm
testavt
fultonem
martinezei
taylorme5
tripptd
becki
cookejm1
davislr2
izquierdogarciap
testavt
zalan
tackercg
herbaal
myersjw1
tillercj
mohanr
```

```
xiaoa
issaa
waitejm1
[trontw@student2 ~]$
```

Users can login and logout of the system any time they want. Your input file contains information about users logging in and logging out of the system, and information about when Tim Tron executes his print statements. For the purposes of this problem, we will assume that Tim has written a simple Unix script called PRINT, to execute the command shown above. Each line in the input contains either the name of a user logging in or out, or PRINT statements. A user may log in or out many times, but a user must be logged in before logging out. In other words, every odd occurrences of the users in the file indicate they are logging in, while every even occurrences of the users in the file indicate they are logging out. Consider, for example, the sample input file `printingusers.in1`.

`Sample Input (see printingusers.in1):`

```
guillra
correaleam
waitejm1
magliettotj
PRINT
correaleam
roachemj
patelvs1
waitejm1
PRINT
magliettotj
olofintuyita
correaleam
patelvs1
tackercg
olofintuyita
PRINT
correaleam
tackercg
patelvs1
PRINT
```

The first four lines indicate users `guillra`, `correaleam`, `waitejm1`, and `magliettotj` are logging in. Then `correaleam` logs out, `roachemj` logs in, `patelvs1` logs in, `waitejm1` logs out, and so on. Wherever the input says PRINT, we would like to print a list of users who are logged in at that time. You may print these users in any order. A sample output is given below, and is also contained in file `printingusers.out1`.

```
Sample Output (see printingusers.out1):

-----------------
correaleam
guillra
waitejm1
magliettotj
-----------------
guillra
patelvs1
roachemj
magliettotj
-----------------
correaleam
guillra
roachemj
tackercg
-----------------
guillra
patelvs1
roachemj
-----------------
```

Between every PRINT output, and before and after the first and last PRINT statements, please output a line with exactly seventeen hypens as indicated above. Your program should work efficiently for an input that contains $10^6$ lines. Please name your program `PrintingUsers.java`.

**Submission:** Please submit all the files as a single zip archive `mid-term.zip` through ASULearn. The zip archive should only contain the individual files of the assignment, and these files should not be inside folders. Moreover, please do not include other extraneous files. Only include all files that belong to your solution.

**Input/ Output Instructions:** For all programs, until and otherwise stated, we will be taking the input from standard input (`System.in`) and will be sending the output to standard output (`System.out`). Since we will be using an automatic grading system, please make sure that your output format exactly matches the description above. So make sure that there are no extraneous output in terms of spaces, newlines and other characters.

**Notes on Coding:** Please do not include user-defined packages in your code. Your code should run in the Unix/Linux machine using the commands `javac` and `java`.

Please note that you are **not allowed to use Java Collections** which contain pre-defined libraries for many of the data structures that we learn in class. The purpose of these assignments is to build these data structures from first principles. Programs that includes these objects will receive 0 points.