University of Waterloo
CS 234 - Data Types and Structures
Spring 2014
Assignment 3
Due Fri June 20th at 4:00 pm

Assignment Guidelines

- Clarifications about the assignment will be posted on Piazza in the thread "Assignment 3 [Official]."
- Instructions on how to submit your assignments to MarkUs will also be posted in Piazza. Your programs must work in the linux.student.cs.uwaterloo.ca environment using Python 2.7.3.
- Use A3-coversheet.pdf for the first page of your assignment or format the top of the first page of your assignment in *exactly* the same manner.
- You may lose up to 20% on a question because it is difficult to read or difficult to understand.

Question 1: Testing Assumptions: Stock.py [25 marks]

In the file StockExchangeList.py (from Supplementary Material section of Learn) find the worst case asymptotic time complexity of the following functions. In particular, identify and state the complexity of any lines that have more than a O(1) worst case asymptotic time complexity. Also state the asymptotic worst case time complexity of the function as a whole.

a) __init__()
b) add()
c) remove()
d) getPrice()
e) sell()

You may assume that operations on strings and files have similar worst case time complexity as comparable operations on lists. You may also assume that the size of the input file is *n* bytes, it has *m* lines of stock data in it, there are 12 columns in each line and that the width of each column is at most 20 characters. For __init__() you may express the time complexity in terms of two parameters *n* and *m*. e.g. O($n^2 + m^2$). You may find in helpful to consult the Stock.py file (also in Learn) to help you with your task.

Question 1 Deliverables: Submit this work on paper in the assignment slots near the Tutorial Centre on the fourth floor of the MC building. Use the A3-coversheet.pdf for your coversheet.

Question 2: Course Project [50 marks]

For this part of the course project you will create a data structure that will track information that flows between your browser and websites you visit.

a) In a file called strCounts.py create a class called StringCounts. This class keep track of how many times a particular string has been "seen." For example, the following snippet of code…

```
sc = StringCounts()
sc.add("www.google.ca")
sc.add("www.youtube.com")
sc.add("www.google.ca")
sc.add("www.youtube.com")
```

```
sc.add("www.google.ca")
sc.report()
```

would yield the following output.

```
www.google.ca                          3
www.youtube.com                        2
```

There are four functions in this class.

StringCounts() - initializes any variables the class needs
add(aString) - if this is the first time aString has been added, add aString and a make the count 1. If the string has already been added increment the count by one.
report() - on the console print out all the strings that have been added and their counts with the following string format "%-50s %4d"
reset() - remove all strings and counts from the object

Raise ValueError and TypeError exceptions where appropriate. Note that issuing the class method report() or reset() on an empty StringCount object should not be an error. Your mark will be based on performance in test cases [10 marks] as well inspection of your source code with regard to clarity (i.e. well documented class and function headers, identified preconditions and postconditions, meaningful error message and variable names) as well as organization, simplicity and efficiency. [10 marks]

b) In a file called strCountsTest.py import the StringCounts class and test it thoroughly [5 marks].

c) In a file called strCommands.py create a function called `getValue(anHTTPmsg, paramName)` where both the two parameters and the return type are strings. For this part of the course project you will generalize the function `getHost`() you created in Assignment 2 so that it will take as a parameter an HTTP parameter name and return an HTTP variable value. For example, in the following HTTP command that your browser might send when you request the webpage `http://www.lib.uwaterloo.ca/hours/` may look something like the following…

```
GET http://www.lib.uwaterloo.ca/hours/ HTTP/1.1
Host: www.lib.uwaterloo.ca
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:29.0) Gecko/20100101 Firefox/29.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Call the above HTTP command *aMsg*. The first word in the first line of *aMsg*, GET, is the name of the function and the first word in each other line (Host:, User-Agent:, Accept:, Accept-Language:, etc) are the parameters names for the GET function. Note that the colon is included in the parameter name. The rest of each line is the parameter values. If the command was `getValue(aMsg,"Host:")` then the value it would return would be the string "`www.lib.uwaterloo.ca`" and for the command `getValue(aMsg, "Accept-Encoding:")` would return the string "`gzip, deflate`". The string methods *splitlines*() and *split*() may prove useful for this task. If a particular parameter name is not in a particular message then raise a ValueError exception.

Create a second function called `hasParam(anHTTPmsg, paramName)` that given anHTTPmsg, as above, and a paramName, returns True if anHTTPmsg has paramName as a parameter name or false otherwise. For example the message above would return True for the paramNames "Host: ", "User-Agent: ", "Accept: ", and "Accept-Language: " since these parameter names occur in the HTTP msg. However it would return False for the "gzip" "keep-alive" or "Cookie:" since the first two are not parameter *names* (they are parameter *values*) and the last does not occur in the HTTP message.

Raise ValueError and TypeError exceptions where appropriate. As in part a), your mark will be based on performance in test cases [10 marks] as well inspection of your source code with regard to clarity (i.e. well documented class and function headers, identified preconditions and postconditions, meaningful error message and variable names) as well as organization, simplicity and efficiency. [10 marks]

d) In a file called strCommandsTest.py import the functions `getValue()` and `hasParam()` and test them thoroughly. Use your programs from the previous two assignment to generate a few HTTP commands from your browser to get some "correct" HTTP commands. You can also edit these commands to create some "incorrect" HTTP commands. [5 marks]

Question 2 Deliverables: submit the following four files to MarkUs: strCounts.py, strCountsTest.py, strCommands.py, and strCommandsTest.py.