Faculty of Arts & Sciences
Department of Computer Science
CMPS 200—Introduction to Programming
Assignment 6 – Due Sunday Oct 26, 2014

Notes

- You are **NOT** allowed to use any concepts not covered yet in class (if, static variables, while loops, etc.).
- You should write the appropriate static methods to show structure in your solution and to help you in writing other methods.
- Give meaningful names to methods and variables in your code.
- Include a comment at the beginning of your program with basic information and a description of the program and include a comment at the start of each method.

Exercises

1. **Palindromes.** Write a method `isPalindrome` that accepts a string as argument and returns `true` or `false` indicating if the string is a palindrome or not. A palindrome is a string that can be read the same way forward and backward. Your method must handle upper and lower case characters (e.g., the string "Madam" is a palindrome).
   The signature of the method should be:
   ```
   public static boolean isPalindrome(String str)
   ```
   Use this method to write a program `Palindromes.java` that takes an integer command line argument N followed by N strings and prints a sequence of N booleans denoting if the strings are palindromes or not.

   Example:
   ```
   > java Palindroms 3 aabbaa abbc hellohello
   true false false
   ```

2. **RemoveCharacter.** Write a method `removeCharacter` that accepts a string `str`, a character `c` and an integer n as arguments, such that the string `str` contains at least n times the character `c`. The method returns the resulting string after removing the first n occurrences of the character `c` from the original string `str` (the method should not print anything to the console). The signature of the method should be:
   ```
   public static String removeCharacter(String str, character c, int n)
   ```
   Use this method to write a program that takes as command line arguments an integer N followed by N tuples; each tuple consists of a string, a character and an integer (`str, c, n`). The program should print each of the strings after removing the first n occurrences of the character c from the corresponding string str.

   Example:
   ```
   > java RemoveCharacter 3 test t 1 shaaaady a 4 testtt t 2
   est shdy estt
   ```

3. **TimeConverter**. Write a program `TimeConverter.java` that takes as argument the time in hours, minutes and seconds in military format: hh:mm:ss and converts its value to seconds only.

   Example:
   ```
   > java TimeConverter 17:31:20
   Original time: 17:31:20
   Time in seconds: 63080
   ```

   Note that the maximum value for the hour is 23, and for the minute and second, it is 59.

4. **Calculator**. Write a program `Calculator.java` that reads an integer N from ***standard input*** and then asks the user to enter N expressions according to the following format: x plus y where x and y are integers. For each expression, your program must produce the corresponding output (see example below):

```
> java Calculator
Enter the number of expressions: 3
Enter expression 1: 3 plus 4
3 plus 4 is equal to 7
Enter expression 2: 7 plus 3
7 plus 3 is equal to 10
Enter expression 3: 10 plus 0
10 plus 0 is equal to 10
```

5. **GivingChange.** Write a program `GivingChange.java` that directs a cashier how to return change to the customer. The program should take two inputs: 1) the amount due; 2) the amount received by the payer. You should break the returned amount in ***quarters***, ***dimes***, **nickels**, and **pennies**. PS: 1 dollar = 100 cents; 1 quarter = 25 cents; 1 dime = 10 cents; 1 nickel = 5 cents; 1 penny = 1 cent.

Example:

```
> java GivingChange 3.77 5
The cashier should give 1.23 in change as follows:
   4 quarters
   2 dimes
   0 nickels
   3 pennies
```

Submission Instructions

- Your submission must consist of a zip archive that contains five .java files only (called **`Palindroms.java`, `RemoveCharacter.java`, `TimeConverter.java`, `Calculator.java`** and **`GivingChange.java`**). No additional files should exist in the .zip archive.
- The name of the zip file must adhere to the following naming convention `asst6_`*`<netid>`*, where *`<netid>`* stands for you AUBnet user name. These zipped files will be processed automatically so please make sure you use this naming convention. The single zipped file must be uploaded to Moodle.