Assignment 3

Due Sunday, October 26, 2014 at midnight

Read the instructions below carefully. The instructions must be followed.

This assignment is worth 5% of your grade. No late assignments will be accepted.

For this assignment, you are allowed to use only the concepts that you have seen in the first 6-7 weeks of class (in particular, what is in chapters 1-6 of the textbook, excluding Section 11 of Chapter 6). For example, you are not allowed to use String class, nor Arrays class nor dynamic arrays.

Put all your code for Question 1, in one file called A3Q1.java. Put all your code for Question 2, in one file called A3Q2.java (for Question 2, this file is provide for you). Place these two files in a folder called A3_xxxxxx, where you should replace xxxxxx by your student number. Compress this folder into A3_xxxxxx.zip file. Submit your assignment, i.e. A3_xxxxxx.zip file, via Blackboard Learn.

Your code must compile, otherwise the assignment will be graded with mark 0. For that reason, if you run out of time and/or your program contains code that does not compile, then comment out that section of your code.

1 Question 1 (6 points) Median

We recall first the definition of median you encountered in Assignment 1. In a group of numbers, any number from that group that satisfies the following property is a *median*: at least half of the elements in the group are smaller or equal to it and at least half of the elements in the group are bigger or equal to it.

For example, in this group of numbers: $10, 11, 13, 15, 16, 23, 26$ the only median is 15. In this group of numbers: $7, 1, 3, 11$, number 3 is a median and 7 is a median. In this group of numbers: $2, 2, 5$, the only median is 2. In this group of numbers: $100, 15, 20, 15, 5, 100$, number 15 is a median and 20 is a median. Finally, in this group of numbers: $100, 15, 15, 5, 100, 15$, number 15 is the only median.

Write a program that has two methods: the main method and a method called median.

The method median take as input (a reference to) an array of integers, and returns an element of that array that is a median of those integers. Inside of this method you must first sort the given array, and then return an element of that array that corresponds to a median (Think where in a sorted array can a median be). In case there is more than one median, the first one is enough.

The main method should:

1. Prompt the user for the total number of numbers that they want a median of.
2. It should then ask the user to enter the numbers (and it should store these numbers in an array).
3. It should then call the median method and print the result it returns.
4. Finally, it should print the array to demonstrate that it is now indeed sorted.

Example run (you can use similar messages, not necessarily identical):

```
your program:  How many numbers do you want a median of
user:  6
your program:  Enter the numbers:
user:  10 15 20 15 5 100
your program:  15 is a median of entered numbers.
Sorted list of entered numbers is:  5, 15, 15, 20, 100, 100
```

2 Question 2 (14 points) Square-Free

A *square-free word* is a word that does not contain any subword twice in a row.

Examples:

ana is square-free.
borborygmus is not square free, since it has subword, bor twice in a row.
abracadabra is square-free.
repetitive is not square-free since subword ti is repeated twice in a row.
grammar is not square-free since subword m is repeated twice in a row.
gaga is not square-free since subword ga is repeated twice in a row.
rambunctious is square-free.
abcab is square-free.
abacaba is square-free.
zrtzghtghtghtq is not square-free since subword ght is repeated twice (in fact three times, but it is enough to find two repetitions to conclude that the word is not square-free).
aa is not square-free since subword a is repeated twice.
zatabracabrac is not square-free since subword abrac is repeated twice in a row.

For this question you will need to write a program that does two things:

(a) tests if a given word is squre-free; and,

(b) given an integer n, where $n \geq 1$, generate and print a square-free word of length n using only three letters a, b and c . To do that you will have to implement an algorithm described below (in Section 2.2).

To do this you will write a java program that has three methods: the main method, a method called isSquareFree that solves part (a), and a method called makeSquareFree that solves part (b).

As part of this assignment, you are provided with a file called A3Q2.java. All your code for Question 2 must go inside this file in clearly indicated spaces (and only in those spaces). You must not change any part of the code that is provided. In particular, the code for the main method is provide to you in its entirety. You cannot add nor delete anything inside of the main method (other than uncommenting the last line in the main method, which allows you to test if your program generates the correct square-free word of length 1000).

Your task is to program the other two methods, isSquareFree and makeSquareFree as detailed below.

2.1 Details for Part (a): method isSquareFree

For this part, implement a method called isSquareFree that takes as input (a reference to) an array of characters. You may assume that the elements of the array are all lower case letters. (In other words, you do not need to worry about a question like: " is Z the same letter as z?")

Your method should test if the given input array of characters is square-free. If it is, the method should print a message stating that, otherwise it should print a message stating that the world is not square-free, where the first subword starts and what that subword is. For example, if the given array contained the word zatabracabrac the method should print:

```
The word, zatabracabrac, is not square free, since it has subword, abrac twice start-
ing at position 4 of the word.
```

2.2 Details for Part(b): method makeSquareFree

(It is easy to see that no word of length at least four that uses only 2 letters of alphabet is square-free.) In 1906, Axel Thue, a Norwegian mathematician, devised a procedure to produce square-free words of arbitrary length using only 3 letters of alphabet.

His method is constructive and uses substitutions over letters a, b and c. More specifically, to get a word of length n the procedure is as follows:

If n=1 the word is a

To get a longer word, you use the following substitution rule. Start with a shorter square-free word (obtained in the previous iteration) and replace:

each letter a in the shorter word by abcab and,
each letter b in the shorter word by acabcb and
each letter c in the shorter word by by acbcacb.

After that is done, if the new word is of length at least n, we are done since we can just take the first n letters of the new word. If it is not done we repeat the procedure until we get a word that is of length at least n. See the provided picture to understand the construction better.

For this part, implement a method called makeSquareFree that takes as input a non-zero positive integer n. The method should produce a square-free word (stored in an array of characters) of length n according to the above specified Thue's procedure. It should return a (reference to) that array. The resulting square-free word will be printed in the main method. Here is an example run:

```
your program:
**************************
        Part (a)
**************************
Enter a word and then press enter:

user:  repetitive

your program:
The word, repetitive, is not square free, since it has subword,
ti twice starting at position 5
Do you want to test another word? Press y for yes, or another key for no

user: y
your program:  Enter a word and then press enter:

user:  mira
your program: The word is square free.
Do you want to test another word? Press y for yes, or another key for no

user: n
your program:

**************************
        Part (b)
**************************
How a long square free word on letters a, b and c do you want?
Enter an integer greater than zero
user: 105
Your program:

abcabacabcbacbcacbabcabacabcbabcabacbcacbabcabacabcbacbcacbacabcbabcabacbcacbacabcbacbcacbabcabacbcacbaca
```