ID# and Name: _____

AMERICAN UNIVERSITY OF ARMENIA
College of Science and Engineering
CS 120 Introduction to Object Oriented Programming in C++ and Java

SAMPLE WAIVER EXAM

Problem 1 (Java)

Write a Java method ***String palindrome(String text)*** that finds and returns the longest palindrome from the specified text. A palindrome is a substring that reads the same way from left to right and from right to left ignoring spaces, punctuation and capitalization. For example, ***palindrome("Let's invite Dr. Awkward!")*** results in ***"drawkward"***.

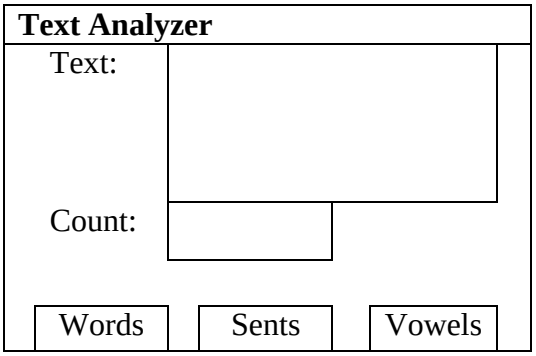
Problem 2 (Java)

Implement a ***public class TextAnalyser*** that extends ***JFrame*** and servers as a simple tool for text analysis. It contains a ***JTextArea*** labeled "**Text:**" for text input, a ***JTextField*** labeled "**Count:**" for result output, and a row of ***JButtons***, as shown on the sketch below. Use the class ***CountTokens*** developed in the Problem 1, if necessary.

The first button is called "Words". By pressing it, the amount of words in the JTextArea must appear in the JTextField. A word is a substring that ends with either a space or '\n' escape sequence.

The second one is called "Sentences". By pressing it, the amount of sentences in the JTextArea must appear in the JTextField. A sentence is a substring that ends with one of the following characters: ***'.'***, ***'!'*** and ***'?'***.

The last one is called "Vowels". By pressing it, the amount of vowels in the JTextArea must appear in the JTextField. Vowels are ***A***, ***E***, ***I***, ***O***, ***U*** and ***Y***. For example, there are 5 vowels in ***"IT IS EASY!"***



Problem 3 (Java)

Implement in black and white a ***public class Clocks*** that extends ***JApplet*** and animates round clocks with hours and minutes arms. No numbers are indicated along the clocks' bound. The motion of the arms should not be synchronized with the real time, but must satisfy the common periodicity rule – the shorter hours arm advances 1 hour after the longer minutes arm completes the entire cycle in 60 steps. The clocks start at 12:00.

ID# and Name: _____

Problem 4 (C++)

Write a C++ function ***void shift(int array[], int length, int offset)*** that takes as an argument of an ***int array*** of the specified ***length*** and cyclically shifts its elements by the specified offset to the left. For example, if ***a[5] = {1, 2, 3, 4, 5}***, then after shifting by ***2***, the elements will be ***{3, 4, 5, 1, 2}***.

Problem 5 (C++)

Write a C++ function ***void spiral(int *center, int odd_size)*** that fills a two-dimensional ***int*** array of the specified odd size ***int oddSize*** with integers from ***1*** to ***odd_size^2*** starting from the central element specified by the pointer ***int *center***. An example of a 5-by-5 spiral is shown below:

13	12	11	10	25
14	3	2	9	24
15	4	1	8	23
16	5	6	7	22
17	18	19	20	21

Problem 6 (C++)

Write a C++ function ***double newton(double (*f)(double), double x1, double x2, double dx)*** that recursively implements the Newton method in find a root of the specified function ***f***. The search for the root is conducted over a specified segment from ***x1*** to ***x2*** (***x1*** $\leq$ ***x2***) assuming that the values of ***f*** at the points ***x1*** and ***x2*** are of different signs. The recursive algorithm is given below:

1. Compute the midpoint ***x0*** of the segment from ***x1*** to ***x2***. If the difference between ***x21*** and ***x1*** is smaller than ***dx*** or the absolute value of ***f*** at ***x0*** is smaller than ***10^{-6}***, return ***x0***;
2. Otherwise, if the values of ***f*** at the points ***x1*** and ***x0*** are of different signs, recursively call ***newton()*** for the range from ***x1*** to ***x0***;
3. Otherwise recursively call ***newton()*** for the range from ***x0*** to ***x2***.