Introduction to Programming

Labs – Week 13a

Exercise 1

Rational numbers are numbers that can be represented as a fraction p/q where p is an integer number and q is a positive integer ($q \neq 0$).

Design and implement an abstract data type (ADT) `Rational` for representing such numbers. Implement methods to add and multiply rational numbers. Implement a method for return the value of a rational number as a `double` value. Make sure that the numerator p and denominator q do not have common divisors in your implementation. Use the algorithm for calculating the *greatest common divisor (gcd)* to ensure this property.

Also, ensure that $q \neq 0$ by adding the line '`if(q==0) throw new RuntimeException()`' in your constructor.

Use the following skeleton.

```
public class Rational {

      // construct a rational number given p and q
      public Rational(int p, int q)

      // compute gcd, a helper method, not part of API
      private static int gcd(int a, int b)

      // return p/q as double value
      public double abs()

      // negate a rational number and return as new a Rational object
      public Rational negate()

      // add this and b and return as a new rational number
      public Rational plus(Rational b)

      // compute this += b
      public void plusEq(Rational b)

      public Rational multiply(Rational b)

      public void multiplyEq(Rational b)

      // convert this rational number to String
      public String toString()

}
```

Exercise 2

Write a function `uniqueChar()` that accepts a string as argument and returns the first character that appears exactly once in the string. Ex: ABCDBADDAB → C.

If every character repeat in the given string than the function should return 0 (the null character in ASCII)

Exercise 3

Write a function `linearIn()` that given two arrays of `int` values sorted in increasing order, `outer` and `inner`, return `true` if all of the numbers in `inner` appear in `outer`. The best solution makes only a single *linear* pass of both arrays, taking advantage of the fact that both arrays are already in sorted order.

Examples

- `linearIn([1, 2, 4, 6], [2, 4])` returns `true`
- `linearIn([1, 2, 4, 6], [2, 3, 4])` returns `false`
- `linearIn([1, 2, 4, 4, 6], [2, 4])` returns `true`

Exercise 4

Given a DNA string, find all genes it contains.

Background: Biologists use a simple model to represent the building blocks of life, in which the letters A, C, G, and T represent the four bases in the *DNA* of living organisms. A *gene* is a substring that represents a functional unit of critical importance in understanding life processes.

A gene has following properties:

- It begins with the start codon ATG.
- Its length is a multiple of 3.
- It ends with one of the stop codons TAG, TAA, or TGA.
- It has no intervening stop codons.

Example: If the string `DNA="ATAGATGCATAGCGCATAGCTAGATGTGCTGAC"`, then ATGCATAGCGCATAG and ATGTGCTGA are two genes inside DNA.