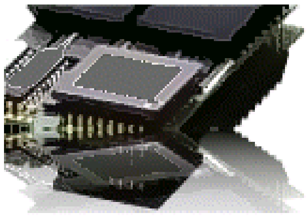

Brock University
Department of Computer Science

COSC 2P13: Computer Systems

Winter 2015 Instructor: **Vlad Wojcik** TA Support: **Anthony Awuley**

Assignment # 1: Due: 13 Feb 2015, 4 PM.

ASSIGNMENT OBJECTIVE:

To familiarize students with limitations of real number arithmetic capabilities of ANY digital computer while solving a simple numerical problem.

THE PROBLEM OF ARITHMETIC WITH REAL NUMBERS

It is common knowledge that human and computer arithmetic of real numbers is full of errors. For example, it is impossible to compute exactly the values of **Pi**, or square root of 2, or even 1/3. To express these values exactly one would need infinite number of bits (or amount of paper), thus making us and our computers infinitely large and slow (exact computation of, say, sqrt(2) would take infinite amount of time). We would need to be immortal, too..

Yes, yes, we all know: Mathematicians talk a lot about real numbers, but observe: they practically do not use them. They limit themselves to use only those reals which can be written precisely (on paper); for all others (which constitute the infinitely greater majority; they can prove it!) they represent them by symbols like **Pi**, **e**, etc. In that sense: humans cannot do more than computers..

Therefore, it has been lectured that you should never, Never, **NEVER** compare two real numbers for exact equality on ANY digital computer, in ANY program, written in ANY programming language. From that it follows that real numbers should not be used as array indices, or **for**-loop indices, etc. Such mistakes would earn you an automatic mark of 0 (**zero**), in ANY assignment, of ANY course, and later, when employed, could get you fired with poor chances of future re-employability. (Did we say **NEVER**? Oh, yeah!)

We cannot remedy this problem inherent to real numbers, but we can improve on the accuracy of digital computer arithmetic, by introducing the concept of rational numbers. A number X is rational if it can be expressed as a ratio of two integers a, b (i.e. we will write X = a/b but **we will NOT perform the division**, which could lead to the loss of accuracy. Of course b cannot be zero).

In real life we frequently use this scheme. For example, the diameters of certain drill bits sold in Canadian Tire are expressed as 3/8 or 7/16 of an inch, instead of 0.375 or 0.4375 of an inch. (Indeed, this is not the best example because the values 3/8 and 7/16 can still be represented exactly decimally or on digital computers; however, the value 2/9 cannot: the best we can do is to write 0.2222222222222222222222222...)

This is an improvement, but not full remedy. Mathematicians can still prove that there are infinitely more real number than rationals. However, we have achieved at least this: Between any two different rationals X and Y we can insert another rational. (Prove it!). Furthermore, the arithmetics on rationals is **exact**, as long as the results stay within the range of representation of integers on a digital computer in use. Indeed, we can increase the chances of that happening by simplifying the fractions, when needed. Old high school stuff, eh?

THE PROGRAM: Rational number calculator

Write a Rational Calculator program, which would perform exact calculations on integers and rational numbers. Create the reusable program component (JavaBean) according to the Ada specification file ***rationals.ads*** which is to contain the implementation of the rational number arithmetic. You are to implement this arithmetic, and then to use this package to write your Rational Calculator.

The calculator program should be responsible for:

- The usual dialogue with a human;
- Performing all calculations using rational arithmetic;
- Displaying final and intermediate results in both rational and decimal format;
- Terminating itself when needed.

NOTE: You may download the package specification of RATIONALS in the WORD format, for your convenience, as file ***rationals.doc***.

DIGRESSIONS (and potential test questions):

1. Can real numbers be used to control **while**-loops? Why or why not?
2. Can enumerated types be used for loop control or as array indices? Why or why not?

SUBMISSION FORMAT:

Both hard-copy (paper) and electronic submission is required.

Hard-copy submission: Your submission envelope with the standard Cover Page should contain all relevant printouts and supporting documentation, demonstrating your design and flawless behaviour of your program. The envelope should be dropped in the submission box on or before deadline date / time.

Electronic submission: Please create a directory on Sandcastle and place within it all the files (and only the files) to be submitted. To submit issue the command ***submit2p91***, which is interactive in its nature. Obviously, you are allowed to submit your assignment only once. Should you encounter difficulties, please report them to Mr. Cale Fairchild.

Similarly, the electronic submission should be performed on or before deadline date / time.

PENALTIES:

Possible lateness in assignment submission is counted in days, each period of a day ending at 4 PM. The penalty for late submission of assignments is 25% up to three days (or a part of a day). After that period the penalty is 100%.

While honest cooperation between students is considered appropriate, the Department considers **plagiarism** a serious offense. For clarification on these issues you are directed to the statement of **Departmental Policies and Procedures**.

Instructor: Vlad Wojcik
Revised: 25 January, 2015 1:12 AM
Copyright � 2015 Vlad WOJCIK