COL226: Programming Languages
II semester 2022-23

Assignment: Integer square-root by long division

Given any positive integer a, its integer square root is a positive integer b satisfying the inequality

$$b^2 \leq a < (b+1)^2 \tag{1}$$

Some of you may have been taught the following method for computing the *integer square root* of an abitrarily large positive integer. Since most positive integers are not perfect squares there is also a non-negative integer remainder $r = a - b^2$. Since these methods are taught by example, you need to understand the method by the following example. The example is given as a sequence of snapshots of the computation by hand. Suppose $a = 3578210349689$ we proceed as follows.

-

	_						
_	3	57	82	10	34	96	89

-

	1						
1	3 1	57	82	10	34	96	89

-

	1	_					
1	3 1	57	82	10	34	96	89
2_	2	57					

-

	1	8					
1	3 1	57	82	10	34	96	89
28	2 2	57 24					

-

	1	8	_				
1	3 1	57	82	10	34	96	89
28	2 2	57 24					
36_		33	82				

-

	1	8	9				
1	3 1	57	82	10	34	96	89
28	2 2	57 24					
369		33 33	82 21				

-

	1	8	9	_			
1	3 1	57	82	10	34	96	89
28	2 2	57 24					
369		33 33	82 21				
378_			61	10			

-

	1	8	9	1			
1	3 1	57	82	10	34	96	89
28	2 2	57 24					
369		33 33	82 21				
3781			61 37	10 81			

- Continuing in this fashion we finally get

	1	8	9	1	6	1	5
1	3 1	57	82	10	34	96	89
28	2 2	57 24					
369		33 33	82 21				
3781			61 37	10 81			
37826			23 22	29 69	34 56		
378321				59 37	78 83	96 21	
3783225				21 18	95 91	75 61	89 25
				3	04	14	64

with the remainder $r = 3041464$.

What you have to do:

1. Describe the method as an algorithm (in pseudo-code) for arbitrarily large positive integers.
2. Give a proof of correctness of the algorithm for an arbitrary number a whose digits are $a_{n-1}, \ldots, a_0$ in decimal for $n > 0$. The algorithm and the proof should be stored in a pdf[1] file called `isqrtld.pdf`.
3. Define a function in SML called `isqrtld` to program your algorithm for arbitrarily large positive integers in decimal. *You are not allowed to use any of the imperative constructs of SML. It should be a purely functional (recursive) program on lists.*

   Input. The function `isqrtld` takes as input a string of digits representing the number a in decimal, and

   Output. returns an ordered pair of strings of digits where the first component is the string represeting the integer square root of the given number a and the second component is the remainder r represented as a string of digits.

   Your SML function and all other code is stored in a text file called `isqrtld.sml`.
4. Zip up both the files `isqrtld.pdf` and `isqrtld.sml` into a single file called *entryno*`.zip` and submit it on **Gradescope** (`https://www.gradescope.com/courses/488514/`) where *entryno* denotes your entry no.

Since the integers can be larger than the maximum integer that your SML implementation may be allowed, the integer a is actually given as a string `''3578210349689''` and the output is also a string and in the case of the example its output is the string `''1891615''` along with the remainder `''3041464''`.

[1] You are encouraged to write the algorithm and the proof in LaTeX

Note for all assignments in general

1. Some instructions here may be overridden explicitly in the assignment for specific assignments
2. Upload and submit a single zip file called `<entryno>.zip` where `entryno` is your entry number.
3. You are *not* allowed to change any of the names or types given in the specification/signature. You are not even allowed to change upper-case letters to lower-case letters or vice-versa.
4. The evaluator may use automatic scripts to evaluate the assignments (especially when the number of submissions is large) and penalise you for deviating from the instructions.
5. You may define any new auxiliary functions/predicates if you like in your code besides those mentioned in the specification.
6. Your program should implement the given specifications/signature.
7. You need to think of the *most efficient way* of implementing the various functions given in the specification/signature so that the function results satisfy their definitions and properties.
8. In a large class or in a large assignment, it is not always possible to specify every single design detail and clear each and every doubt. So you are encouraged to also include a README.txt or README.md file containing
    - all the decisions (they could be design decisions or resolution of ambiguities present in the assignment) that you have taken in order to solve the problem. Whether your decision is "reasonable" will be evaluated by the evaluator of the assignment.
    - a description of which code you have "borrowed" from various sources, along with the identity of the source.
    - all sources that you have consulted in solving the assignment.
    - name changes or signature changes if any, along with a full justification of why that was necessary.
9. The evaluator may look at your source code before evaluating it, you must explain your algorithms in the form of comments, so that the evaluator can understand what you have implemented.
10. Do *not* add any more decorations or functions or user-interfaces in order to impress the evaluator of the program. Nobody is going to be impressed by it.
11. There is a serious penalty for code similarity (similarity goes much deeper than variable names, indentation and line numbering). If it is felt that there is too much similarity in the code between any two persons, then both are going to be penalized equally. So please set permissions on your directories, so that others have no access to your programs.
12. To reduce penalties, a clear section called "**Acknowledgements**" giving a detailed list of what you copied from where or whom may be be included.