1 Introduction

You will write some simple programs in the Impcore programming language in order to gain familiarity with the language and to practice writing recursive functions.

2 Description

In small groups, work on the following problems. In addition to the specifications given below, your functions must use recursion (and must not use iteration via while) and must not use global variables.

For each problem, first develop at least 3 unit tests, written using check-expect definitions as follows:

- \( \text{(check-expect \ (fact 1) 1)} \) ;; Covers base case
- \( \text{(check-expect \ (fact 5) 120)} \) ;; Covers inductive case

The Impcore interpreter automatically runs all unit tests at the end of the source file in which the tests appear.

After developing the unit tests, discuss the appropriate algorithm and then write the code.

1. Define a function \( \text{pow} \) such that \( \text{(pow x p)} \) returns \( x^p \). Assume \( 0 \leq p \) and define \( x^0 = 1 \) for all \( x \) (including \( x = 0 \)).

2. Define a function \( \text{sumPows} \) such that \( \text{(sumPows p n)} \) returns \( 0^p + 1^p + \cdots + n^p \). Assume \( 0 \leq p \) and \( 0 \leq n \).

3. Define a function \( \text{hasDigit?} \) such that \( \text{(hasDigit? d n)} \) returns 1 if the digit \( d \) appears in the decimal representation of the number \( n \) and returns 0 otherwise. Assume \( 0 \leq d < 10 \) and \( 0 \leq n \).

   If your group has extra time, consider generalizing to a function \( \text{hasDigitInBase?} \) such that \( \text{(hasDigitInBase? d b n)} \) returns 1 if the digit \( d \) appears in the base \( b \) representation of the number \( n \) and returns 0 otherwise. Assume \( 0 \leq d < b \) and \( 0 \leq n \).

4. Define a function \( \text{sqrt} \) such that \( \text{(sqrt n)} \) returns the number \( s \) satisfying \( s^2 \leq n < (s+1)^2 \). Assume \( 0 \leq n \). (Hint: You will need a helper function and \( \text{sqrt} \) will simply call the helper function with some initial arguments.)

5. Define a function \( \text{reverse} \) such that \( \text{(reverse n)} \) returns the number with the decimal digits of \( n \) in reverse order. For example, \( \text{(reverse 19771017)} \) should return 71017791. Assume \( 0 \leq n \). (Hint: You will need a helper function and \( \text{reverse} \) will simply call the helper function with some initial arguments.)

6. A narcissistic (decimal) number (see https://rosettacode.org/wiki/Narcissistic_decimal_number) is a non-negative integer \( n \) that is equal to the sum of the \( m \)-th powers of each of the digits in the decimal representation of \( n \), where \( m \) is the number of digits in the decimal representation of \( n \). For example, 153 is a narcissistic number because \( 153 = 1^3 + 5^3 + 3^3 \). Note that all single digit numbers are narcissistic; some other narcissistic numbers are 370, 371, 407, 1634, 8208, 9474, 54748, 92727, and 93084.

   Define a function \( \text{narcissistic?} \) such that such that \( \text{(narcissistic? n)} \) returns 1 if the number \( n \) is a narcissistic number and returns 0 if the number \( n \) is not a narcissistic number. Assume \( 0 \leq n \).

3 Requirements and Submission

You may use the reference interpreter (see Appendix A), but there may only be one active laptop in each group.

At the end of class, submit the group’s solutions (test-cases and code) either as hard-copy or by e-mail to mtf@cs.rit.edu be sure to include the names of all group members in the submission.
4 Hints

• ImpCore provides the following primitive or pre-defined functions: +, -, *, /, mod, =, !=, <, <=, >, >=, not, and, or

• You may define additional helper functions.

• Efficient solutions are not required (although may be entertaining to discover).

• There are various ways in which a recursive function can decompose a natural number:

  – Decrement by one
    Base case: \( n = 0 \)
    Recursive case: \( n = m + 1 \), recurse on \( m \)

  – Split into two pieces
    Base case: \( n = 0 \)
    Recursive case: \( n = k + (n - k) \) (where \( 0 < k < n \)), recurse on \( n - k \)

  – Sequence of decimal digits
    Base case: \( n = d \) (where \( 0 < d < 10 \))
    Recursive case: \( n = 10 \times m + d \) (where \( 0 < d < 10 \) and \( m > 0 \)), recurse on \( m \)

A Interpreter

A reference Impcore interpreter is available on the CS Department Linux systems (e.g., glados.cs.rit.edu and queeg.cs.rit.edu and ICLs 1 and 2) at:

/usr/local/pub/mtf/plc/bin/impcore

Use the reference interpreter to check your code.

A.1 Interactive mode

Simply executing

\$ /usr/local/pub/mtf/plc/bin/impcore

will run the interpreter interactively, but without line editing.

Executing

\$ rlwrap /usr/local/pub/mtf/plc/bin/impcore

or

\$ ledit /usr/local/pub/mtf/plc/bin/impcore

will run the interpreter interactively with line editing. (See the manual pages for rlwrap and ledit for more details.)

A.2 Batch mode

Executing

\$ cat rec01.imp | /usr/local/pub/mtf/plc/bin/impcore

will run the interpreter on the contents of the file rec01.imp, but with prompts printed.

Executing

\$ cat rec01.imp | /usr/local/pub/mtf/plc/bin/impcore -q

will run the interpreter on the contents of the file rec01.imp without prompts printed.