Programming Language Concepts

CSCI-344 **Programming 1**
Term 20161 **August 24, 2016**

Impcore Programming

Due: September 2, 2016

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.

Download `prog01.imp` and `prog01_tests.imp`. The former is a template for your submission and also includes a number of supporting functions. The latter is a test suite for the assignment.

2 Description

Complete exercises 4 through 10 of Chapter 1 from *Programming Languages: Build, Prove, Compare* (pp. 70 – 73). Each one is to define one or more Impcore functions.

For Exercise 9b (`all-fours?`), the presence or absence of a negative sign in the decimal representation of the input number should not affect the answer (which is consistent with how negative numbers are treated in Exercise 9c and Exercise 9d). In particular, the *exercise transcripts* should read

```
-> (all-fours? -4)
-1
```

and the *unsatisfying answer* is wrong (and not just unsatisfying).

In addition to the specifications given in the exercises, your functions must use recursion (and must not use iteration via `while`) and must not use global variables. Place your solutions to all exercises in a file named `prog01.imp`.

3 Requirements and Submission

Your submission must be a valid Impcore program. In particular, it must pass the following test:

```
$ cat prog01.imp | /usr/local/pub/mtf/plc/bin/impcore -q > /dev/null
```

without any error messages. If your submission produces error messages (e.g., syntax errors), then your submission will not be tested and will result in zero credit for the assignment.

Submit `prog01.imp` to the `Programming 01` Dropbox on MyCourses by the due date.

4 Hints

- 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: $n = 10 * m + d$ (where $0 < d < 10$ and $m > 0$), recurse on m

  The assignment will require you to use at least one more.
- The reference solution is approximately 90 lines of Impcore.

Document History

August 24, 2016
Original version

A Interpreter

A reference Impcore interpreter is available on the CS Department Linux systems (e.g., `queeg.cs.rit.edu` and ICLs 1, 2, and 3) at:

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

Use the reference interpreter to check your code.

Source code for the interpreter is available on the CS Department file system at:

`/usr/local/pub/mtf/plc/src/bare/impcore`

and

`/usr/local/pub/mtf/plc/src/commented/impcore`

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 prog01.imp | /usr/local/pub/mtf/plc/bin/impcore
```

will run the interpreter on the contents of the file `prog01.imp`, but with prompts printed.

Executing

```
$ cat prog01.imp | /usr/local/pub/mtf/plc/bin/impcore -q
```

will run the interpreter on the contents of the file `prog01.imp` without prompts printed.

Executing

```
$ cat prog01.imp prog01_tests.imp | /usr/local/pub/mtf/plc/bin/impcore -q
```

will run the interpreter on the contents of the files `prog01.imp` and `prog01_tests.imp` without prompts printed.