COMP 3007 - Assignment #2

Recursion & First-class Functions

Due: Friday February 7th @ 11:55pm

Question 1

a. [5 marks] Newton's method for cube roots is based on the fact that if y is an approximation to the cube root of x, then a better approximation is given by the value: `(x/y`^{2}`+2y)/3`
   Use this formula to implement a cube-root procedure analogous to the square-root procedure from the lecture notes. Your code should use nested functions and free variables wherever possible.
b. [2 marks] Consider the `(new-if)` procedure as shown below. Replace the standard if inside cbrt-iter with a call to new-if instead. Does the new version work? Explain why or why not.

```
(define (new-if predicate consequent alternate)
    (cond (predicate consequent)
        (else alternate)))

;example usage
(new-if (< x 0) (- x) x)
```

Question 2

Recall the definition of general summation of numbers between two values as described in lecture (copied here for reference):

```
(define (sum a b term next)
    (if (> a b)
        0
        (+ (term a)
           (sum (next a) b term next))))
```

a. [2 marks] Write a higher-order procedure called `product` analogous to sum procedure from lecture. The procedure should generate a *recursive process*.
   E.g. `(product 1 5 (lambda(x)x)(lambda(x)(+ x 1))) → 120`
b. [5 marks] Rewrite your solution to the previous problem using an *iterative process*. Call this procedure `(product-it)`
c. [3 marks] Using your solution to either of the previous problems, produce the given PI notations below.

- $\prod_{j=1}^{25} (j^3 + j)$
- $\prod_{k=1}^{10} (2k + 1)^2$
- $\prod_{i=1}^{10} \prod_{j=i}^{10} 3ij$

Question 3

a. [5 marks] Write a procedure called `(palindrome? s)` that takes a string `s` as argument and returns true (#t) if `s` is palindrome.
b. [7 marks] A k-palindrome is a string that can be made into a palindrome by ignoring up to k characters. Write a procedure called `(k-palindrome? s k)` that takes a string `s` and a non-negative integer `k` as arguments and returns true if s is a k-palindrome. For example,

```
(k-palindrome? "tahcohcat" 2) → #t
(k-palindrome? "tahcohcat" 1) → #f
```

You may find the following built-in procedures useful for the above problems:

- `(string-length str)` - returns the number of characters in a given string
- `(string-ref str i)` - returns the character at a specified index in a given string
- `(substring str start [end])` - returns a substring of a given string between two indices [the ending index is optional].

Question 4

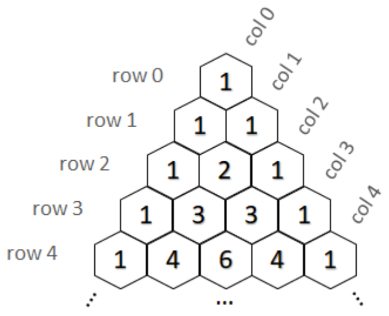
A function f is defined by the rules:

```
f(n) = n, if n<3
f(n) = f(n-1) + 2f(n-2) + 3f(n-3), otherwise
```

a. [4 marks] Write a procedure that computes f by means of a *recursive process*. Illustrate that your answer is recursive by showing the substitution model for (f 5) in comments below your code.
b. [7 marks] Write a procedure that computes f by means of an *iterative process*. Illustrate that your answer is iterative by showing the substitution model for (f 5) in comments below your code.

Question 5

The following pattern of numbers is called Pascal's triangle.



The numbers at the edge of the triangle are all 1, and each number inside the triangle is the sum of the two numbers above it.

a. [5 marks] Write a procedure that computes elements of Pascal's triangle given row and column indices. Any invalid indices should return a value of 0.
For example,

```
(pascals 0 0) → 1
(pascals 2 0) → 1
(pascals 2 1) → 2
(pascals 4 2) → 6
```

b. [7 marks] Using your solution to the previous part, write a procedure (printTriangle n) that prints n rows of Pascal's triangle to the screen.
For example,

```
(printTriangle 5)

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Question 6

A hyperoperation is a repeated sequence of arithmetic operations where, a hyperoperation of rank n is a repeated sequence of hyperoperations of rank n-1.

For example, Addition is a hyperoperation of rank 1. Multiplication is a hyper operation of rank 2.
Multiplication can be expressed as a sequence of additions: `a * b = a + (a + (a + ... + a))` (b times)
Exponentiation is a hyper operation of rank 3: $a^b = a * (a * (a * ... * a))$ (b times)

Tetration is the 4th rank hyper operation: $a\uparrow\uparrow b = a^{(a^{(a^{\cdot^{\cdot^{a}}})})}$
This pattern continues with pentation, hexation, septation, etc...

[6 marks] Write a function called `(hyper x)` that takes a hyperoperator x as argument and returns the next higher rank hyperoperator.
For example:

```
(define my-mult (hyper +))
(my-mult 3 4) → 12

(define my-exp (hyper my-mult))
(my-exp 2 4) → 16

(define my-tetra (hyper my-exp))
(my-tetra 2 4) → 65536
```

Some notes:

- This approach will have many special cases for zeroes and negative numbers, so you may assume exclusively positive integers in your solution.
- For testing you should create the three hyperoperations shown here (multiplication, exponentiation, and tetration), and validate that they work as they should.
- Technically, addition is a hyperoperation made up of repeated increments, but this again leads to some special cases and inconsistencies, so you may assume addition as the lowest rank hyperoperation for your solution. That is, your function need only generate hyperoperations of rank 1 and up.
- For any hyperoperation (H) with rank > 1, `aH1 = a`.

Documentation & Testing

Documentation

- Ensure that your name and student number are in comments at the top of all files.
- Document the purpose of each function including its expected inputs (parameters) and output (return).
- Ensure that your code is well-formatted and easily readable; a happy TA is a generous TA.

Testing

- You are required to include testing runs of every function in your submission.
- The specific tests required depend on the question at hand, but should cover all valid inputs and all possible branches of your code.
- The example runs provided in the guidelines above may not be sufficient.
- Unless otherwise specified, you may assume inputs supplied are of the correct type.
- Fabricated test outputs will result in 0 marks for a question.
- For best practices: Comment your testing as to what you are testing and why, giving expected output as well as observed output and explanations for any differences.

[10 marks]

An example submission including documentation and testing can be found here: primes.scm

Submission

- Any files that are not runnable (in DrRacket using R5RS) will result in a mark of 0 for that question.
- Your solutions for this assignment are expected to use correct functional style as demonstrated in class. This means no use of looping constructs or mutable variables.
- Combine all files into a single .zip file for your submission.
- Submit your assignment using cuLearn before the due date.

- Marks will be deducted for late submissions.
- You are responsible for submitting all files and ENSURING that the submission is completed successfully.
- If you are having issues with submission, contact me **before** the due date, afterwards late deductions will apply.
- Please see the course outline for all submission guidelines.