

## Programming Assignment #4 – Monday, 26 November

### Problem Statement:

For this programming assignment, you will gain some experience programming in Scheme using basic expressions, lists, list operations, recursive functions, and higher-order functions.

### Pascal's Triangle:

The philosopher and mathematician Blaise Pascal (1623-1662) is famous among modern day computer scientists for Pascal's Triangle, and the programming language Pascal was named in his honor. Pascal's Triangle is an arithmetical triangle representing the integer coefficients of the expansion of the binomial equation  $(x+1)^n$ , for  $n=0,1,\dots,n$ . For example, here is Pascal's Triangle for  $n = 0,1,\dots,7$ :

```

n=0
          1
n=1
        1  1
       1  2  1
      1  3  3  1
     1  4  6  4  1
    1  5 10 10  5  1
   1  6 15 20 15  6  1
n=7
  1  7 21 35 35 21  7  1

```

To compute Pascal's Triangle in Scheme start with the list '(1). To construct the  $i^{\text{th}}$  row, compute the sum of the first two entries in the row above, then the next entry as the sum of the 2<sup>nd</sup> & 3<sup>rd</sup> entries, and so on, until the last entry of the row above. To compute the 1's on the edges, you pretend that each row above the current one is 0 1 ... 1 0.

**Task 1: Implement a function that named "pascals-triangle" that given an argument  $n$  returns a list of lists representing rows 0,1,...,n. I.E., ((1) (1 1) (1 2 1) (1 3 3 1) ...).**

When we refer to a given entry in Pascal's Triangle, we give a row number and a place in that row, beginning with row zero and place zero. For instance, the number 20 appears in row 6, place 3. This value can be computed using a closed formula of factorials:

$$C(n, k) = \frac{n!}{k! (n-k)!} \quad \text{for } k \leq n$$

Notice that in computing  $n!$ ,  $k!$  is computed along the way. For example,  $C(6,3) = 6!/(3!(6-3)!) = 20$ .

**Task 2: Implement a recursive function named "n-choose-k" that implements the  $C(n,k)$  equation above so that no redundant multiplications are needed for any factorial calculation. (Hint: use let or let\* expressions).**

### Horner's Method:

In 1819, the mathematician William Horner rediscovered (Isaac Newton discovered it first) a method for efficiently evaluating an  $n^{\text{th}}$  degree polynomial in less than the  $(n^2+n)/2$  multiplications required if the naïve algorithm suggested by the standard equation for a polynomial is used:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_1 x^1 + a_0 x^0 \quad \text{requires } O(n^2) \text{ multiplications}$$

Horner recognized that a polynomial can be written in the form:

$$P_n(x) = (((a_n x + a_{n-1})x + a_{n-2})x + \dots)x + a_1)x + a_0 \quad \text{requires } O(n) \text{ multiplications}$$

This form of a polynomial requires  $2n+1$  multiplications and  $n$  additions. From this equation, we can see directly how to implement Horner's Method as a recursive function  $H_n(x)$  for a given list of coefficients  $a_0, a_1, \dots, a_n$ , where  $H_n(x)$  is defined recursively as:

$$\begin{aligned} H_0(x) &= a_0 \\ H_i(x) &= H_{i-1}(x) * x + a_i \quad \text{where } 0 \leq i \leq n \end{aligned}$$

To evaluate a polynomial  $P_n(x)$  we just compute  $H_n(x)$  for a given value of  $x$  and a list of  $n$  coefficients  $a_i$ . In Scheme, we can represent the coefficients as a list of integer or real values, depending on whether we need to evaluate the polynomial using integer or real coefficients. For example, a function to evaluate a 4<sup>th</sup> degree polynomial with real coefficients  $a_0=1.0$ ,  $a_1=5.0$ ,  $a_2=0.0$ , and  $a_3=3.0$ , for some value of  $x$ , we be written as:

```
(rec-eval-poly x '(1.0 5.0 0.0 3.0))
```

**Task 3: Implement a recursive function named “rec-eval-poly” function using Horner’s method.**

**Task 4: Implement a function named “foldl-eval-poly” using the higher-order foldl function to implement Horner’s method.**

**Task 5: Implement a function named “eval-pascals-triangle” that given arguments  $n$  and  $x$ , which evaluates the polynomial  $(x+1)^n$  whose coefficients are the  $n^{\text{th}}$  row of Pascal’s Triangle. You must use your  $n$ -choose- $k$  function and your evaluation function must allow either the rec-eval-poly or foldl-eval-poly to be given as a function argument. You must use your previously defined functions. It is incorrect to just compute  $x+1$  and then raise it to the power  $n$ .**

#### Submission Requirements:

You are to hand in at the start of class on the due date a complete source code listing of your Scheme program printed using `enscript -2Gr -C -Escheme`. You are also to submit an electronic copy of your Scheme source program using the “turnin” program BEFORE class begins on the due date. Late assignments will not be accepted.

#### Honor Policy:

This assignment is a individual learning opportunity that will be evaluated based on your ability to think independently, work through a problem in a logical manner and implement a software program on your own. You may however discuss verbally or via email the general nature of the conceptual problem to be solved with your classmates, the course TA or the course instructor, but you are to complete the actual programming for this assignment without resorting to help from any other person or other resources (electronic or otherwise) that are not authorized as part of this course. If in doubt, ask the course instructor. *Any indication of improper cooperation or the use of non-approved course materials will result in a grade of zero being assigned for this project and referral to the College for possible disciplinary action with respect to your continued participation in this course.*