CSE1729 - Introduction to Programming | Fall 2018

Laboratory Assignment 2

Objectives

- Work with recursive functions
- Work with the conditional `if` special form

Activities

1. **Combinations:** Often, we are interested in the number of ways a group of k objects can be formed from a total of n objects. For instance, if I needed to select three students from a lab section containing twenty students. How many ways are there to make this selection? First, notice that for the first student selected, I can choose any one out of the twenty in the section. Therefore, there are twenty ways to make this selection. For the second student, I am selecting from the nineteen remaining students. Therefore, there are nineteen ways to make this selection. Finally, I am making the selection of the last student from the remaining eighteen. So, there are eighteen ways to make the final selection. But, the ordering of our selections don't make any difference. Consider the number of ways we can select a group of three students A, B and C. ABC, ACB, BAC, BCA, CBA and CAB give us six ways the same group of three have been counted. So, the total number of groups that can be counted is:

$$\frac{20 \cdot 19 \cdot 18}{3 \cdot 2 \cdot 1}$$

In general, we use the following equation to compute the number of groups of size k that can be formed from n objects:

$$\frac{n!}{(n-k)!k!}$$

You will notice that the $(n-k)!$ term in the denominator cancels the last $(n-k)$ terms in the product of $n!$ in the numerator, leaving just the product of the top k terms from $n!$. Also notice, the $k!$ term prevents us from counting the duplicate ways of selecting the same k objects. This computation has it's own terminology and notation. The notation is $\binom{n}{k}$ and is read "n choose k" indicating this quantity represents the number of ways of choosing k objects from a group of n objects. So,

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

Use the `factorial` function discussed in lecture to write an `(n-choose-k n k)` function that takes two parameters, n and k and computes the value of $\binom{n}{k}$ shown above.

Remark 1. *Note that* `(n-choose-k n k)` *should evaluate to zero when* $n < k$ *or* $k < 0$.

2. **Exponentiation** Write a SCHEME procedure to compute b^e by multiplying b a total of e times. Name your function `(pow b e)`.

3. Write a recursive function, named `(zeno n)`, that computes the sum of the first n terms of the following series from Zeno's Dichotomy Paradox, $Z_n = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^n}$. Use your `(pow b e)` function to compute the denominator of each term.

4. **Number of Digits** Write a SCHEME procedure, named `(num-digits n)`, which accepts an integer `n` and returns the number of digits in `n`. For example, both `(num-digits 10000)` and `(num-digits 12345)` evaluate to 5.

5. Young Jeanie knows she has two parents, four grandparents, eight great grandparents, and so on.

   (a) Write a recursive function, named `(a n)` to compute the number of Jeanie's ancestors in the n^{th} previous generation. The number of ancestors in each generation back produces a sequence that may look familiar:

   $$2, 4, 8, 16, \ldots$$

   For each generation back, there are twice the number of ancestors than in the previous generation back. That is, $a_n = 2a_{n-1}$. Of course, Jeanie knows she has two ancestors, her parents, one generation back.

   (b) Write a recursive function to compute Jeanie's total number of ancestors if we go back n generations. Specifically, `(num-ancestors n)` should return:

   $$2+4+8+\cdots+a_n$$

   Use your function in part (**??**) as a "helper" function in the definition of `(num-ancestors n)`[1].

[1] Of course, we can use the closed-form solution for the geometric progression to compute `num-ancestors` ($ancestors(n) = 2^{n+1} - 2$) but that doesn't give us any experience with recursive functions. However, this is a useful fact we can use when testing our functions to ensure they are correct.