CMPSC 461: Programming Language Concepts

Spring 2018

Programming assignment 2: Scheme Programming
Total: 21 points. Due on March 23rd at 12:20pm in Canvas.

1. (3 points) Write a Scheme `remove-if` function. It takes two arguments. The first is a boolean function f, and the second is a list l. The `remove-if` function returns a list with all elements x in l such that $f(x) = \#t$ being removed.

   For example,

   ```
   (remove-if (lambda (x) (> x 3)) '(10 1 7 2))
   ```

   should return the list (1 2), since both 10 and 7 are greater than 3.

   Define the `remove-if` function by using case analysis and recursion.

2. (3 points) Define a function called `removeLast` that takes a list and returns a new list and the returned list is the same as the input list except the last element in the input list is removed (if there is one). For example, `removeLast '(1 2 3 4)` returns `'(1 2 3)`. Use case analysis and recursion to write the function.

3. (3 points) The `rev` function takes two lists as arguments. It returns a list that is the concatenation of the *reverse* of the first list with the second list. For example, (rev '(1 2 3) '(4 5)) should return (3 2 1 4 5).

   (a) (2 points) Define the `rev` function by using case analysis and recursion.

   (b) (1 point) Now we want to define a `reverseList` function, which reverses a list. Define `reverseList` based on the `rev` function.

4. (4 points) Write a Scheme function `int-to-words` that takes a nonnegative integer between 0 and 99, inclusive, as a parameter and returns a list of words corresponding to the integers reading in English. Make your function as short as possible; i.e., don't write a COND statement with 100 tests. Example calls to your function are given below:

```
(int-to-words 13) ; should return (thirteen)
(int-to-words 42) ; should return (forty two)
```

Note, in order to simplify things, we are not expecting dashes in the output (i.e., 42 is output as forty two not forty-two). Hint: You may want to use the built-in integer division functions `quotient` and `remainder`.

5. (8 points) A single-variable polynomial of degree n is written as

$$a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$$

where $a_n, ..., a_0$ are coefficients. Suppose we represent such a polynomial as a list $(a_0, a_1, a_2, \ldots, a_n)$.

In this question, you are asked to write a Scheme function `polyMult` that performs polynomial multiplication of two polynomials. For instance, when given `(1 2 1)` and `(1 2 1)`, `polyMult` should return `(1 4 6 4 1)`, because

$$(1 + 2x + x^2)(1 + 2x + x^2) = 1 + 4x + 6x^2 + 4x^3 + x^4$$

We are going to implement the polynomial multiplication by converting it into a series of polynomial addition operations. For the example, the multiplication can be performed in the following way:

$$\begin{aligned}
& (1 + 2x + x^2)(1 + 2x + x^2) \\
= \; & 1 * (1 + 2x + x^2) + 2x * (1 + 2x + x^2) + x^2 * (1 + 2x + x^2) \\
= \; & (1 + 2x + x^2) + \\
& (0 + 2x + 4x^2 + 2x^3) + \\
& (0 + 0 \;\; + \;\; x^2 + 2x^3 + x^4) \\
= \; & 1 + 4x + 6x^2 + 4x^3 + x^4
\end{aligned}$$

Do the following steps to implement the above polynomial multiplication procedure.

(a) Write a function `nzero`, which takes a number n and returns a list of n zeros. For instance, calling `nzero` on 3 returns `(0 0 0)`.

(b) Write a polynomial addition function `polyAdd` that adds two polynomials. For instance, when given `(1 2 1)` and `(0 2 4 2)`, it returns `(1 4 5 2)`.

(c) Write a function `polyAddList`, which takes a list of polynomials, and adds all of them together. It should be based on `polyAdd`. For instance, when given `((1 2 1) (0 2 4 2) (0 0 1 2 1))`, it returns `(1 4 6 4 1)`.

(d) Write `polyMult` based on `polyAddList`.

You may follow the hint below to implement `polyMult`. (You don't need to follow the hint if you can come up with another way of implementing `polyMult`.)

Hint: Write a function `polyMultHelper`, which takes two lists of numbers, $l1$ and $l2$, and a number n as parameters and returns a list of lists of numbers. The first element in the result is the result of appending the list of n zeros to the list of mapping the function that multiples the first element in $l1$ to $l2$; the second element in the result is the result of appending the list of $n+1$ zeros to the list of mapping the function that multiples the first element in $l1$ to $l2$; ... For instance,

```
polyMultHelper '(1 2 1) '(1 2 1) 0
```

should produce the result

```
((1 2 1) (0 2 4 2) (0 0 1 2 1))
```

Notes For programming assignments, examples are given only for the purpose of clarification. By no means that our tests will solely be based on those examples. It's your responsibility to thoroughly test your code by designing your own test cases.

Chapter 6 of the Scheme standard `http://www.schemers.org/Documents/Standards/R5RS/HTML/` contains a list of standard procedures provided by Scheme (for example, the remainder and the square root functions). You may find them helpful during your programming.

Submission format: Put all your code into one DrRacket file and submit your file through Canvas. Please clearly mark your answers using comments so that we can tell the correspondence between your code and questions. Please ensure that your file starts with a comment that includes your name and your Penn State network user id.