CS-1117 Introduction to Programming 2015-2016

Assignment #19

Recursive List Processing

Write *recursive* definitions for each of the following Python functions, and for each function, include a clear and concise comment to describe its purpose — as always, this comment should be self-contained and should mention each parameter explicitly.

1. `Count( x, s )`

   The number of times item 'x' occurs in sequence 's'

   ```
   Count( "i", "floccipaucinihilipilification" ) ⇒ 9
   Count( "z", "floccipaucinihilipilification" ) ⇒ 0
   Count(  2,  [ 2, 5, 2, 2, 4, 7, 2, 3 ]      ) ⇒ 4
   ```

2. `IsSorted( s )`

   Is sequence 's' sorted in ascending order? (adjacent items may be equal)

   ```
   IsSorted( [ 2, 5, 5, 7, 8 ] ) ⇒ True
   IsSorted( "abcdefg" )         ⇒ True
   IsSorted( "abcedfg" )         ⇒ False
   IsSorted( [ 3 ] )             ⇒ True
   IsSorted( [ ] )               ⇒ True
   ```

3. `IsPalindrome( s )`

   Does sequence 's' read the same forwards and backwards?

   ```
   IsPalindrome( [ 1, 2, 3, 2, 1 ] ) ⇒ True
   IsPalindrome( "rotavator" )       ⇒ True
   IsPalindrome( "" )                ⇒ True
   IsPalindrome( "abcdba" )          ⇒ False
   ```

4. `Max( s )`

   The maximum item in sequence 's', or `None` if 's' is empty; pay attention to efficiency here, and ensure that, for example, `Max( range( 100 ) )` returns its answer almost instantly

5. `Map( f, lst )`

   Equivalent to the list comprehension: `[ f( x ) for x in lst ]`

   ```
   Map( lambda n : 2 * n, [ 3, 1, 4, 2 ] ) ⇒ [ 6, 2, 8, 4 ]
   ```

6. `Filter( p, lst )`

   Equivalent to the list comprehension: `[ x for x in lst if p( x ) ]`

   ```
   Filter( lambda n : n > 0, [ 0, 1, -2, 3, -4 ] ) ⇒ [ 1, 3 ]
   ```

Program Submission:

Store the function definitions in a file named 'a19.py', and turn it in for grading by typing:

```
submit-cs1117 a19.py
```

Due Date: Fri Mar 11, 10:30am