Department of Computing

COMP1126 – Introduction to Computing
2013-2014 Semester II
Assignment

The prime number, 197, is called a circular prime because all the integers that are created after repeated, circular, left shifts of the digits 1, 9 and 7 are themselves prime i.e. 971 and 719 are both prime numbers. There are 13 such primes below 100: 2, 3, 5, 7, 11, 13, 17, 31, 37, 71, 73, 79, 97. For this assignment you will create a solution to the problem of determining how many circular primes there are below a given value.

Read the instructions below and ensure you understand what is being asked ***before*** attempting to write any code. I hope you enjoy creating your solution.

1. Write a function called `digits` that counts the number of digits of an integer. The function takes a single, positive integer and returns the number of digits of the integer. For example,

   ```
   >>> digits(197)
   3
   ```

   `digits` must incorporate the use of a locally defined (i.e. within the `digits` function itself), tail recursive, helper function.

2. Next, write a function called `lcs` which performs a single digit, left circular shift of the digits of an integer. The function takes a single, positive integer and returns an integer which is the result of a single digit, left circular shift of the digits of the input. For example,

   ```
   >>> lcs(197)
   971
   ```

   Think of the process like this: the leftmost digit is first removed, then each of the other digits is shifted one position to the left and the original leftmost digit (the 1 in the above example) is appended to the rightmost position. NB, `lcs` operates on the integer datatype only.

3. This one should be quite familiar to you. Write a function called `isPrime` that accepts a single, positive integer and determines whether the integer is a prime number or not. `isPrime` therefore returns a boolean result. For example,

```
>>> isPrime(197)
True
```

Do a bit of research and find a procedure for determining the primality of an integer that is more efficient than what we used in lecture. If your chosen method involves repetition, that repetition can be performed using either an iterative or a recursive process, you choose.

4. Now for the crux of the solution. Write a function called `isCircular` which accepts a prime number and determines whether it is circular or not. It therefore returns a boolean result. `isCircular` utilizes the services of all three functions you previously wrote i.e. `digits`, `lcs` and `isPrime`. `isCircular` returns `True` if the prime number it is provided as input is a circular prime, otherwise it returns `False`. For example,

   ```
   >>> isCircular(197)
   True
   >>> isCircular(23)
   False
   ```

   `isCircular` must be written using a locally defined (i.e. within the `isCircular` function itself), tail recursive, helper function.

5. Finally, write the function called `nbrCirPrimes`. This function takes a single, positive integer and returns the number of circular primes that exist that are less than (and excluding) the integer. For example,

   ```
   >>> nbrCirPrimes(100)
   13
   ```

   `nbrCirPrimes` utilizes the functions `isPrime` and `isCircular` to complete its task. Write this function iteratively.

This assignment is worth 15% of your coursework mark. It is due on February 24th 2014, at 11pm. Post your submission via OurVLE. Look for the container named "Assignment". You are permitted to work in pairs, therefore *ensure that your code has BOTH members' ID numbers included as a comment*. Only one member of the pair is to submit. Each person may submit as a member of one (1) programming pair only. No late submissions will be accepted.

Name your file according to the following convention,

***IF** your ID numbers are 620000001 and 620000002*
 ***THEN** the file name should be 620000001_620000002.py*