CSE 30
Programming Abstractions: Python
Lab Assignment 1

In this assignment you will write a Python program containing generator functions, i.e. functions that return generator objects. Such functions will be discussed at length in class. Read the section on Generators in Part I of *Programming Abstractions in Python* (PAP), which contains a nice explanation of the `yield` command in Python. Also study the examples LinearSearch.py and Generators.py posted on the class webpage under /Examples/Generators to learn more about use of the `yield` command, and use of the `next()` function on iterators.

Write a function called power_remainder(n, m, r) that returns a generator object which produces the infinite sequence of positive integers that (1) are exact powers of n, and (2) have remainder r upon division by m. For instance the sequence $(1, 4, 16, 25, 49, 64, \ldots\ldots\ldots)$ consisting of all perfect squares having remainder 1 upon division by 3, is generated by power_remainder$(2, 3, 1)$. Hint: write an infinite loop that produces all perfect n^{th} powers, and `yield` only those that have remainder r when divided by m.

Next, write a function called common_terms(g, h) that returns a generator object which produces the sequence of common terms in the sequences produced by generators g and h, respectively. For instance if the generator g produces the sequence $(2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, \ldots\ldots\ldots)$ of even numbers, and the generator h produces $(5, 10, 15, 20, 25, 30, \ldots\ldots\ldots)$ produces the multiples of 5, then the generator returned by common_terms(g, h) will produce the sequence $(10, 20, 30, 40, \ldots\ldots\ldots)$ of numbers that are multiples of both 2 and 5 (i.e. multiples of 10). It is assumed that both g and h produce increasing sequences. Hint: step through the sequences for g and h simultaneously, advancing only on a sequence if it is behind the other. If at some point the sequences are on a common element, `yield` that value.

Test your two functions thoroughly. The following template will be posted on the webpage under /Examples/lab1.

```
#-------------------------------------------------------------------------
#  SequenceTemplate.py
#-------------------------------------------------------------------------

def power_remainder(n, m, r):
    """
    Returns a generator of the (infinite) sequence of positive integers that
    are (1) exact powers of n and (2) have remainder r upon division by m.
    """


# end

def common_terms(g, h):
    """
    Returns a generator of the sequence of terms that are common to the two
    (increasing) sequences produced by generators g and h.
    """


    # end
# end
```

```
def main():

   A = power_remainder(2, 3, 1)
   B = power_remainder(3, 5, 4)
   C = common_terms(power_remainder(2, 3, 1), power_remainder(3, 5, 4))

   print()
   for i in range(15):
      s = "  {0:<12}{1:<12}{2:<12}".format(next(A), next(B), next(C))
      print(s)
   # end
   print()

# end

#-------------------------------------------------------------------------
if __name__=='__main__':

   main()

# end
```

Fill in the function definitions, leave the part of the program below the conditional unchanged, and save file as Sequence.py. The expected output of the above 'main' program is given below.

```
  1           64          64
  4           729         117649
  16          2744        262144
  25          6859        4826809
  49          13824       24137569
  64          24389       113379904
  100         39304       148035889
  121         59319       481890304
  169         85184       1073741824
  196         117649      2565726409
  256         157464      3010936384
  289         205379      6321363049
  361         262144      10779215329
  400         328509      19770609664
  484         405224      22164361129
```

What to turn in

Submit the file Sequence.py to the assignment lab1 on Gradescope before the due date.