CSC326: Assignment 1

Problem 1. The greatest common divisor (GCD) of a and b is the largest number that divides both of them with no remainder. Create a file `q1.py` containing a function `gcd` that takes as input two positive integer numbers and returns their greatest common divisor. For example, `gcd(12,16)=4`, and `gcd(4,3)=1`.

Problem 2. Create a file `q2.py` containing a function `rotate_word` that takes as input a string and a positive integer i and rotates the characters in the string i times to the left. For example, `rotate_word("abc", 2)="cab"`, `rotate_word("abc", 9)="abc"`.

Problem 3. Consider a *recursive, pure* function (a function without side effect) like `fib` below, which computes the fibinacci number for a given input.

```
def fib( n ) :
    if n == 0 or n == 1 :
        return n
    else :
        return fib(n-1) + fib(n-2)
```

A common strategy to speed up the evaluation of such functions is to employ *dynamic programming*: recognizing that the return values of pure function calls are always the same for the same input argument value, one could remember the computation result for an input value the first time a call for that value is made, and use it later, thus avoiding redundant calculations. For example, when calculating `fib(5)`, calls to `fib(3)` will be made 2 times, one in `fib(5)`, and the other in `fib(4)`, which is in turned called by `fib(5)`. As explained, dynamic programming can eliminate such redundant computations.

Create a file `q3.py` containing a functions `fib` that takes as input a non-negative integer n and a list l, returns the fibinacci number for the given n, and appends the given n in list l. For example, `l=[]; f=fib(6, l)` should assign `8` to f and `[6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4]` to l. Note that you must implement `fib` with dynamic programming.

Problem 4. A Python function factorize(N) takes an integer as an argument, and returns an ordered list of prime integers whose product equals to `N`. For example, `factorize(6) returns [1, 2, 3]`.

Follow the *test-driven design methodology* and create a file `q4.py` containing the implementation of the function `factorize(N)` and the unit tester for the function. You should use the doctest framework.

Problem 5. Function `Q` calculates and returns a list of results according to the given formula: Q = Square root of [(2 * C * D)/H], where C is 50 and H is 30. D is the variable whose values should be input to the function in a comma-separated sequence.

For example, assume the following comma separated input sequence is given to the function: 100,150,180. The list of results returned by the function should be: 18,22,24.

Create a file `q5.py` containing a function `Q(D)` that takes as input a list D and returns a list of results.

Submission Instructions Compress all of your files and save them either as asn1.zip or asn1.tar.gz and submit the files from a computer in one of the following labs: GB243, GB251E, or SF2102, or by SSH by logging on to one of the aformentioned computers (ug132.eecg - ug180.eecg, ug201.eecg - ug249.eecg).

Submission Command Example

```
ug132:~% submitcsc326f 1 asn1.zip
```