CSC241 – Lab 6

Instructions: Submit a single file lab6.py containing solutions to the following problems. Make sure you run the doctest to check your solutions before you submit. Copy and paste the results of the doctest in a multiline comment at the top of your submission.

1. Write a function `dromes` that takes one argument, the name of a text file. The function then returns all the words in the file that are palindromes of length more than 1 ('`a`' and '`I`' don't count). Recall that palindromes are words that are the same forwards and backwards. They should be recognized regardless of lower/upper case. Keep in mind also that the file contains the punctuation marks, period, comma, or question mark following some words, and this should not prevent them from being recognized. Sample usage:

```
>>> dromes('text1.txt')
['wow', 'Malayalam', 'wow', 'tattarrattat', 'wow', 'Senones',
'SIXAXIS', 'soosoos', 'Tacocat', 'Zerorez', 'wow', 'wow',
'wow', 'wow', 'wow', 'wow']
>>> dromes('text2.txt')
['wow', 'wow', 'wow', 'did', 'wow', 'wow', 'Kanak', 'kayak',
'kelek', 'level', 'Liril', 'wow']
```

2. Write a function `multiplesOf` that takes two arguments, `n` - a positive integer and `lst` – a list of integers. The function then **returns** the list of indices of the numbers in `lst` that are multiples of `n`. `lst` should not be modified by the function Sample usage:

```
>>> multiplesOf( 3, [3,1,6,2,7,9,10,20,3] )
[0, 2, 5, 8]
>>> multiplesOf( 5, [3,1,6,2,7,9,10,20,5] )
[6, 7, 8]
>>> multiplesOf( 13, [3,1,6,2,7,9,10,20,5] )
[]
```

3. Write a function `mod` that takes two positive integers, `n` and `k`, as arguments. The function then returns `n%k`. It is, however, not allowed to use the % operator. Instead it should repeatedly subtract `k` from `n` as many times as possible. Sample usage:

```
>>> mod(10,3)
1
>>> mod(10,7)
3
>>> mod(10,13)
10
>>> mod(10,10)
```

```
0
```

4. Write a function `coins` that determines how to break a number of cents into standard coins (quarters, dimes, nickels, and pennies). Given a positive integer amount, the function will return a list of denominations in descending order that sum to the given amount. It must use the largest coins possible (i.e., not allowed to give back all pennies). For example, 81 cents should be converted to 3 quarters, a nickel and 1 penny. (Hint: nested iteration where the outer loop iterates over the denominations and the inner loop is a while that subtracts.) Sample usage:

```
>>> coins(26)
[25, 1]
>>> coins(81)
[25, 25, 25, 5, 1]
>>> coins(234)
[25, 25, 25, 25, 25, 25, 25, 25, 25, 5, 1, 1, 1, 1]
>>> coins(22)
[10, 10, 1, 1]
>>> [ (i,coins(i)) for i in range(0,100,7)]
[(0, []), (7, [5, 1, 1]), (14, [10, 1, 1, 1, 1]), (21, [10,
10, 1]), (28, [25, 1, 1, 1]), (35, [25, 10]), (42, [25, 10, 5,
1, 1]), (49, [25, 10, 10, 1, 1, 1, 1]), (56, [25, 25, 5, 1]),
(63, [25, 25, 10, 1, 1, 1]), (70, [25, 25, 10, 10]), (77, [25,
25, 25, 1, 1]), (84, [25, 25, 25, 5, 1, 1, 1, 1]), (91, [25,
25, 25, 10, 5, 1]), (98, [25, 25, 25, 10, 10, 1, 1, 1])]
>>>
```