CSC241 – HW5

Reading: Chapter 5

Problems:

Place your solutions to the following problems in a file hw5.py. Make sure you run the doctest before you submit.

1. Write a function `subStrings` that takes a list of strings as an argument. It **returns** a list that contains every string in the list that is a substring of the one that precedes it (comes immediately before it in the list). If the list is empty or contains only one string, the function doesn't print anything. Note that this means that the function will never print the first item in the list since it has no predecessor in the list. The following shows several sample runs of the function:

```
>>> subStrings(['apple','app','pp','orange','range','ra'])
['app', 'pp', 'range', 'ra']
>>>
subStrings(['apple','app','pp','orange','range','pineapple'])
['app', 'pp', 'range']
>>> subStrings(['apple'])
[]
>>> subStrings([])
[]
>>>
```

2. A polynomial of degree n with coefficients a0, a1, a2, a3, ..., an is the function: p(x) = a0 + a1x + a2x**2 + a3x**3 + ... + anx**n. This function can be evaluated at different values of x. For examples, if p(x) = 1 + 2x + x**2, then p(2) = 1 + 2 * 2 + 2**2 = 9. If p(x) = 1 + x**2 + x**4 then p(2) = 21 and p(3) = 91. Write a function `poly` that takes as a parameter a list of coefficients a0, a1, a2, a3, ... an of a polynomial p(x) and a value x. It returns p(x) which is the value of the polynomial when evaluated at x. Note that the usage below is for the three examples discussed in this paragraph. Your function must work correctly for any polynomial on any value:

```
>>> poly(  [1,2,1], 2 )
9
>>> poly( [1,0,1,0,1], 2 )
21
>>> poly( [1,0,1,0,1], 3 )
91
```

3. Write a function `collatz` that takes one argument, a positive integer n. The function then generates and returns the Collatz sequence starting at n (and ending at 1). The Collatz sequence is a sequence obtained by repeatedly applying the following rule to the previous number in the sequence:

    a. If a number is even, then the next number is that number divided by 2. Eg. 10->5
    b. If a number is odd, then the next number is 3 times the number plus 1. Eg 5->16

    Your function should stop when the sequence gets to 1. Note that it is an open question whether the Collatz sequence for every positive integer always ends at 1. Sample usage:

```
>>> collatz(10)
[10, 5, 16, 8, 4, 2, 1]
>>> collatz(22)
[22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
```

4. The mathematical constant π (pi) is an irrational number with value approximately 3.1415928... The precise value of π is equal to the following infinite sum: π = 4/1 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + ... We can get a good approximation of π by computing the sum of the first few terms. Write a function `approxPi` that takes as a parameter a floating point value error and approximates the constant π within error by computing the above sum, term by term, until the absolute value of the difference between the current sum and the previous sum (with one fewer terms) is no greater than error. Once the function finds that the difference is less than error, it should return the new sum. Please note that this function should not use any functions or constants from the math module. You are supposed to use the described algorithm to approximate π, not use the built-in value in Python. The following shows the execution of the function on some examples:

```
>>> approxPi(.5)
3.3396825396825403
>>> approxPi(.05)
3.1659792728432157
>>> approxPi(.005)
3.144086415298761
>>> approxPi(.0005)
3.1418425911015158
```