NYU, Tandon School of Engineering

CS-UY 1134 Data Structures and Algorithms — Summer 2024

# Homework #2
# Due by Friday 5/31, 11:59pm

**Submission instructions:**

1. For this assignment, you should turn in 5 .py files,
   name your files using the pattern:
   'YourNetID_hwX_qX.py'

Note: your netID follows an abc123 pattern, not N12345678.

2. **You should submit your homework via Gradescope**.

For Gradescope's autograding feature to work:

a. Name all classes, functions and methods **exactly as they are in the assignment specifications**.

b. Make sure there are **no print statements** in your code. If you have tester code, please put it in a "main" function and **do not call it**.

**Question 1:**

The Fibonacci Numbers Sequence, Fn, is defined as follows:
$F_0$ is 1, $F_1$ is 1, and $F_n = F_{n-1} + F_{n-2}$ for n = 2, 3, 4, ...
In other words, each number is the sum of the previous two numbers.
The first 10 numbers in Fibonacci sequence are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Note:
Background of Fibonacci sequence:
https://en.wikipedia.org/wiki/Fibonacci_number

Implement a function `def fibs(n)`. This function is given a positive integer n and returns a generator, that when iterated over, it will have the first n elements in the Fibonacci sequence.

For Example, if we execute the following code:

```
for curr in fibs(8):
     print(curr)
```

The expected output is:

```
1 1 2 3 5 8 13 21
```

**Question 2:**

Implement a function `def factors(num)`. This function is given a positive integer num, and returns a generator, that when iterated over, it will have all of num's divisors in an ascending order.

For Example, if we execute the following code:

```
for curr_factor in factors(100):
     print(curr_factor)
```

The expected output is:

```
1 2 4 5 10 20 25 50 100
```

Implementation requirement: Pay attention to the running time of your implementation. The for loop like the above, should run in a total cost of $\theta(\sqrt{num})$.

**Question 3:**

Define a generator that takes in a number n and returns the powers of 2 up to n:
`def powers_of_two(n)`

For example:
`powers_of_two(6)`
will return
`1, 2, 4, 8, 16, 32`

**Question 4:**

Implement the function `def findChange(lst01).`
This function is given `lst01`, a list of integers containing a sequence of 0s followed by a sequence of 1s.
When called, it returns the index of the first 1 in lst01.

For example, if lst01 is a list containing [0, 0, 0, 0, 0, 1, 1, 1], calling
`findChange(lst01)` will return 5.

Note: Pay attention to the running time of your function. If lst01 is a list of size n,
an efficient implementation would run in logarithmic time, $\Theta(\log_2(n))$.

**Question 5:**

a. Write a function `def shift(lst, k)` that is given a list of N numbers, and some positive integer k (where k<N). The function should shift the numbers circularly k steps to the left.

   The shift has to be done in-place. That is, the numbers in the parameter list should reorder to form the correct output (you shouldn't create and return a new list with the shifted result).

   For example, if `lst = [1, 2, 3, 4, 5, 6]`
   after calling `shift(lst, 2)`,
   `lst` will be `[3, 4, 5, 6, 1, 2]`

b. Modify your implementation, so we could optionally pass to the function a third argument that indicates the direction of the shift (either 'left' or 'right').

   Note: if only two parameters are passed, the function should shift, by default, to the left.

   Hint: Use the syntax for default parameter values.

   **Note:**
   For both part a and b, you are not allowed to use the pop() or insert() method of python list. The runtime would not be linear if you use those methods. Instead try to use the `reverse_list()` function in lab 1 question 3 to solve the problem.