Homework 1

Assigned: 11/17

Due: 11/24, 11:59pm

Possible Points: 100

Submit on Canvas a zip file containing your solution code for each problem (just the py files, please don't submit the solution/project files), turn in one .py file for each problem. If a problem has multiple parts, it should still be one file. Each problem specifies what you should title the file containing your solution so that the autograder can find it. Incorrectly named programs will not be graded and thus will get 0 points for the problem.

Since the assignment will be graded using an automated grading system you should have your program's output match the sample output shown. If your program fails to run using Visual Studio 2017 it will receive zero points. If your program fails to run properly (runtime errors or wrong output) points will be deducted based on how correct the program is.

If you have any questions or concerns about the assignment do not hesitate to post on the discussion board on canvas, send us mail on canvas, or see us after class or during office hours.

Students are expected to write their source code from scratch, those who copy code from each other or online (including from Canvas examples) will be reported for cheating. We will also look at your code so definitely don't just hard-code the answer and print it.

1. Testing Cases: 10 points

File: `testing_cases.py`

This program will handle a simple situation of reading the number of cases to be tested and will then iterate through each case, printing the case number as `Case #:` and echoing back the word input by the user. Your case numbers should start counting from 0. The case number printed out is a header so our autograder knows when to start grading a new case. The echoing of the user input is the code being tested by the grader.

For this problem, you should use `input()` function to read strings of characters from the standard input, and store them. Remember all the input from 'input()' function will be read as strings, you may need to convert input to int.For example, if you want to input number of cases your program will run:

`number_of_case = int(input())`

For output, use `print()` function.

Example Input

```
7
Hello!
Goose Dog
Chicken
Duck
Hello
Computer
Whoops!
```

Expected Output

```
Case 0:
Echo: Hello!
Case 1:
Echo: Goose Dog
Case 2:
Echo: Chicken
Case 3:
Echo: Duck
Case 4:
Echo: Hello
Case 5:
Echo: Computer
Case 6:
Echo: Whoops!
```

---

2. Find the Minimum and Maximum of a List of Numbers: 30 points

File: `find_min_max.py`

Write a program that reads some number of integers from the user and finds the minimum and maximum numbers in this list. The first number denotes number of cases. The input for each case consists of a single line containing list of integers. Your program will read in this list elements and find the minimum and maximum values and print them back out. Do not use any built-in functions to find the maximum and minimum in the list, e.g, max() and min().

Example Input:

```
4
100 -50 14 32 -5 124
-502 123 12 -42
10 -60 9993 -1230 412 510 -142 -23 90 0 13 -45
```

```
89 32 -82 16 0 -2 78
```

Expected output:

```
Case 0:
Min: -50
Max: 124
Case 1:
Min: -502
Max: 123
Case 2:
Min: -1230
Max: 9993
Case 3:
Min: -82
Max: 89
```

---

3. Flip Flop: 30 points

File: `flip_flop.py`

We want to model the behavior of a strange sort of fish over some time. On seconds divisible by *a* it flips, on those divisible by *b* it flops and on those divisible by both it flips and flops. To simulate this behavior your program should print "flip" when it flips, "flop" when it flops and "flipflop" when it flips and flops. If the fish doesn't do any action then you should just print out the current second.

Input to your program for each case will be the second to start at, the number of seconds to simulate and the values for *a* and *b*. Your program should then log the behavior of the fish by printing its action or the current time as specified above for the desired number of seconds. Note that 0 counts as divisible by any number, recall that `0 % x` is always 0 for any value of `x`.

Example Input

```
3
0 16 3 5
10 10 2 7
4 20 12 4
```

Expected Output

```
Case 0:
flipflop
1
2
flip
```

```
4
flop
flip
7
8
flip
flop
11
flip
13
14
flipflop
Case 1:
flip
11
flip
13
flipflop
15
flip
17
flip
19
Case 2:
flop
5
6
7
flop
9
10
11
flipflop
13
14
15
flop
17
18
19
flop
21
22
23
```

4. Estimating π using Leibniz's formula: 30 points

File: `leibniz.py`

The German mathematican Leibniz (1646 - 1716) discovered the following formula to approximate π:

$\pi/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + ...$

Write a program to compute an approximation of π using the first n terms in Leibniz's series, where $1 \leq n \leq 10000000$ is input. Print your output in fixed-point notation, to up to 8 digits of accuracy. For example, to output the variable `num` in fixed point notation, to 4 digits of accuracy after the decimal point, do as follows:

```
num = 13.94222222222
print('{:.4f}'.format(num))
```

Example input

```
6
1
100
1000
10000
100000
1000000
```

Example output

```
Case 0:
Pi estimated as: 4.00000000
Case 1:
Pi estimated as: 3.13159290
Case 2:
Pi estimated as: 3.14059265
Case 3:
Pi estimated as: 3.14149265
Case 4:
Pi estimated as: 3.14158265
Case 5:
Pi estimated as: 3.14159165
```