CS 11 Exercise 04
1st Semester, AY2018-2019

University of the Philippines, Diliman

September 10, 2018

Instructions

- For this exercise, you have to write five different computer programs. A submission link is available in UVLe for each program.
- Make sure that your algorithm works given the sample input and output. You must also check if your algorithm can also handle input other than the ones given.
- Please remove any prompt messages (e.g. `Enter number:`) when getting input. Prompt messages will mess with your output, making your solution invalid.
- See the sample input and output to guide you on what and how your program must display output.
- Submit your solutions on or before Sunday, September 17 at 11:59pm.

Preliminaries

Input and Output Redirection

Several operating systems have terminals that allow the user to redirect the input and output from the terminal to files. For example, if we are running a Python program named `sample.py` through the terminal, we can tell Python to get the input from another file (say, `input.txt`) by entering the following command:

```
python3 sample.py < input.txt
```

Similarly, the output of the program can be also be redirected to a file:

```
python3 sample.py > output.txt
```

Redirection can be performed to the input and the output simultaneously.

```
python3 sample.py < input.txt > output.txt
```

Note that this technique is not the same as reading/writing a file through programming. File reading and file writing in Python will be covered in the near future.

The diff Command

The Linux operating system has a program called `diff` which compares the contents of two files. If the files are equal, the `diff` tool will not print anything. In the files have different contents, the diff tool will list the difference between the two files, along with the list of changes necessary to convert one file into the other.

Printing New Lines

When the print function is executed without any arguments (i.e. `print()`), Python will print a newline character (`'\n'`). When a newline character is printed, the next symbols will be printed on the next line.

Methods

Several data types in Python come with methods. A **method** is similar to a function. It is a named sequence of instructions that may take arguments and may also return a value. However, unlike functions, a method must always be associated with a value. A method always works on the value along with the arguments provided, and any value that it may return is somehow related to the value that is associated with it.

Here are some examples of methods on strings:

```
>>> word = 'banana     '
>>> word
'banana     '
>>> word = word.upper()
```

```
>>> word
'BANANA     '
>>> word = word.lower()
>>> word
'banana     '
>>> word = word.capitalize()
>>> word
'Banana     '
>>> word = word.rstrip()
>>> word
'Banana'
```

Notice that to call a method, we append a '.' to the value where the method is associated, followed by the call to the method.

1 Multiplication Table

Write a program that prints an $n \times n$ multiplication table, given a positive integer n.

1.1 Input Format

Input will start with a positive integer t, followed by t test cases. Each test case is a positive integer n.

1.1.1 Sample Input

```
5
1
2
3
4
5
```

1.2 Output Format

For each test case n, output an $n \times n$ multiplication table. Each entry in the table must be separated with a tab character (`'\t'`). Each row must be printed on a separate line. Each table must be followed by a new line.

1.2.1 Sample Output

```
1

1	2
2	4

1	2	3
2	4	6
3	6	9

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
```

```
5    10   15   20   25
```

2 Diamond

Write a program that prints a diamond using the '*' symbol, given the side length s.

2.1 Input Format

Input will start with a positive integer t, followed by t test cases. Each test case is a positive integer s, the side length of the diamond. Assume that $t, s > 0$.

2.1.1 Sample Input

```
4
1
2
3
4
```

2.2 Output Format

For each test case, print a diamond with side length s. A unit of the side is equivalent to a '*' symbol. Each diamond must be followed by a new line.

2.2.1 Sample Output

```
*

 *
***
 *

  *
 ***
*****
 ***
  *

   *
  ***
 *****
*******
 *****
  ***
   *
```

3 Pyramid of Pyramids

Write a program that prints a pyramid of pyramids using the '*' symbol, given height h.

3.1 Input Format

Input will start with a positive integer t, followed by t test cases. Each test case is a positive integer h, which is the height of the pyramid. Assume that $h, s > 0$.

3.1.1 Sample Input

```
3
1
2
3
```

3.2 Output Format

For each test case, print a pyramid of pyramids with height h. A unit of height is equivalent to a pyramid of size n. On each line of the pyramid of pyramids, the rightmost '*' must not be followed by any other whitespace characters (see sample output for details). Each pyramid of pyramids must be followed by a new line.

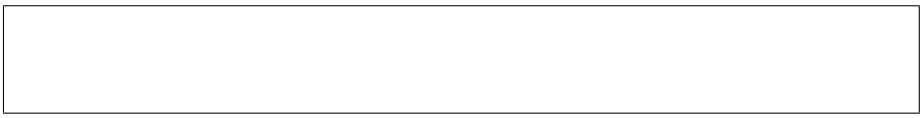
Hint: Python strings have a method called `rstrip` that removes the trailing whitespace characters, from a string.

3.2.1 Sample Output

```
*

    *
   ***
 *  *  *
*********

           *
          ***
         *****
       *    *    *
      ***  ***  ***
     ***************
  *    *    *    *    *
 ***  ***  ***  ***  ***
***********************
```



4 Fizz Buzz A, B, C

The Fizz-Buzz test is a popular interview question designed to "help filter out the 99.5% of programming job candidates."[1]. On the Fizz-Buzz test, you have to write a program that prints the numbers from 1 to 100, except for some values. For numbers that are for multiples of three, you have to print "Fizz" instead of the number. For multiples of five, print "Buzz," and for numbers which are multiples of both three and five, print "FizzBuzz."

On this problem, you have to program a modified version of the Fizz-Buzz test:

> For each number between A to B (inclusive) in *increments* (or decrements) of C (starting from A), print out its corresponding Fizzbuzz text.

On our version of Fizz Buzz, the criteria for printing the Fizz-Buzz text stays the same:

- If the number is divisible by both 3 and by 5, output "FizzBuzz".
- If the number is divisible by 3 but not by 5, output "Fizz".
- If the number is divisible by 5 and but not by 3, output "Buzz".
- If the number is not divisible by either 3 or by 5, output the number itself.

4.1 Input

Input will start with a positive integer t, followed by t test cases. Each test case consists of three different numbers, A, B, and C, each on a separate line.

4.1.1 Sample Input

```
3
17
15
1
16
12
-2
10
15
1
```

4.2 Output

If the combination of A, B, and C is invalid, output *"Invalid Input!"*. Invalid input is as follows:

[1] http://wiki.c2.com/?FizzBuzzTest

- C is 0.
- C is positive, A is greater than B.
- C is negative, A is less than B.

If the combination of A, B, and C is valid, output the Fizz-Buzz text sequence (see sample input and output). Assume that all other combinations of A, B, and C are valid.

Each corresponding number and Fizz-Buzz text should be on a separate line. The output of each test case must end with a new line.

4.2.1 Sample Output

```
Invalid Input!

16
14
Fizz

Buzz
11
Fizz
13
14
FizzBuzz
```

5 Modular Inverse

The *modulo*, or $\mod$, is a binary operation that takes two integers, say a and b, and gives the remainder of a when divided to b. Given integers a, b, and r, the equation

$$r = a \mod b$$

tells us that r is the remainder when we divide a by b. Here, we refer to r as the remainder and b as the modulus.

Your goal for this problem is to write a program that computes the modular inverse of a, given an integer $a > 0$ and an integer modulus $n > 0$. The modular inverse of a with respect to n, denoted as a^{-1}, is an integer greater than 0 such that

$$aa^{-1} \mod n = 1$$

5.1 Input

Input will start with a positive integer t, followed by t test cases. Each test case consists of two integers. The first integer is a positive integer a, and the second input is the modulus n. Assume that $0 < a < n$.

5.1.1 Sample Input

```
4
3
7
10
33
16
63
4
42
```

5.2 Output

For each test case, print the modular inverse a^{-1} with respect to n. Note that some values of a have no inverse with respect to n, in which case your program must output `DOES NOT EXIST`. Note that for $a \in (0 \ldots n)$, if a^{-1} exists then its value must also be in $(0 \ldots n)$

5.2.1 Sample Output

```
5
10
4
DOES NOT EXIST
```