CODEFEST- PYTHON 3

FACULTY :	**INAM UL HAQ**
CAMPUS :	**GULSHAN**
SESSION :	**3:30– 6:30**

QUESTION NO 1 :

FIND A LEAP YEAR

We add a Leap Day on February 29, almost every four years. The leap day is an extra, or intercalary day and we add it to the shortest month of the year, February.

In the Gregorian calendar three criteria must be taken into account to identify leap years:

1) The year can be evenly divided by 4, is a leap year, unless:

2) The year can be evenly divided by 100, it is NOT a leap year, unless:

3) The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years.Source

Task :

You are given the year, and you have to write a function to check if the year is leap or not.

Note that you have to complete the function and remaining code is given as template.

Input Format :

Read y, the year that needs to be checked.

Output Format :

Output is taken care of by the template. Your function must return a boolean value (True/False)

Sample Input :

1990

Sample Output :

False

Explanation :

1990 is not a multiple of 4 hence it's not a leap year.

CODE STRUCTURE :

```
def is_leap(year):
    leap = False
    # Write your logic here
    return leap
year = int(input())
print(is_leap(year))
```

QUESTION NO 2 :

The Farm Problem

You've got chickens (2 legs), cows (4 legs) and pigs (4 legs) on your farm. Return the total number of legs on your farm.

Examples :

1) animals(2, 3, 5) ⟶ 36

2) animals(1, 2, 3) ⟶ 22

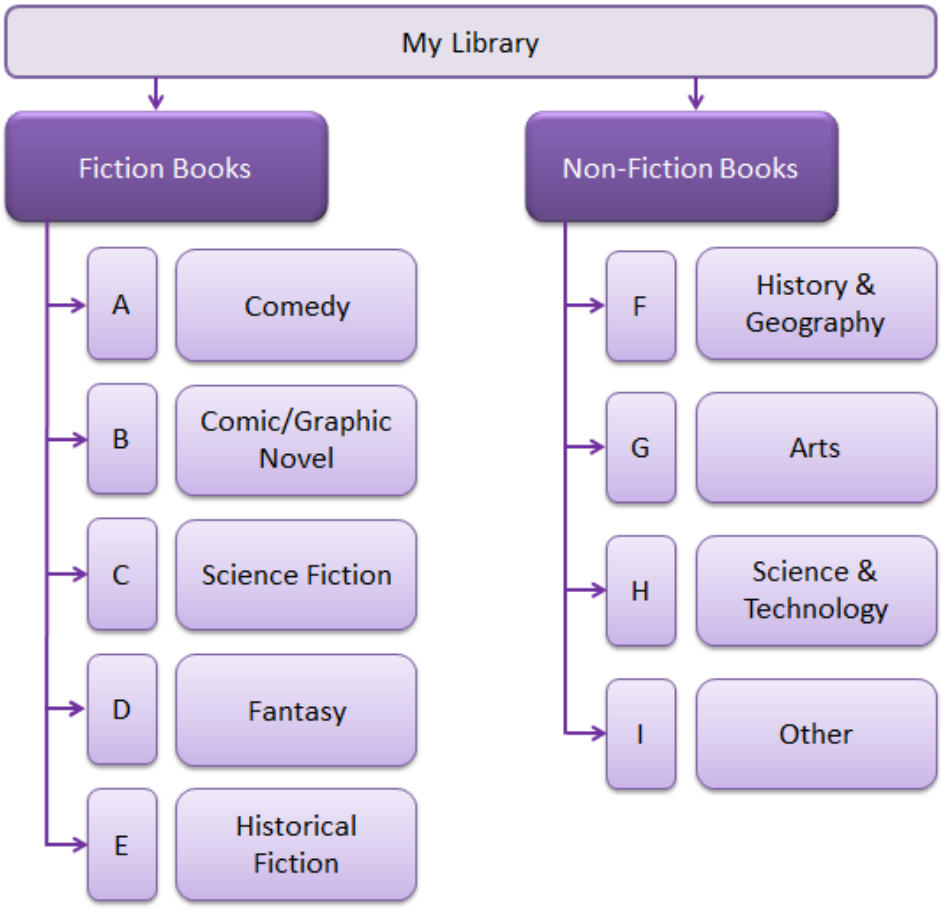
3) animals(5, 2, 8) ⟶ 50

QUESTION NO 3 :

MY LIBRARY

For this challenge you will work for the librarian who needs a computer program to help pupils find out where books can be found in the library.

The library contains nine bookshelves labelled from A to I. Each bookshelf specializes in one genre. For instance, bookshelf A is for comedy fiction books.



TASK :

Your task is to write a computer program that asks the user if they are looking for a fiction or a non-fiction book. Based on the user answer the program will ask the user to choose the genre from a list of available genres. Finally the program will return the location (A to I) of books of this genre.

QUESTION NO 4 :

RUNNER UP SCORE :

Given the participants' score sheet for your University Sports Day, you are required to find the runner-up

score. You are given scores. Store them in a list and find the score of the runner-up.

Input Format :

The first line contains . The second line contains an array of integers each separated by a space.

Output Format:

Print the runner-up score.

Sample Input :

52

3 6 6 5

Sample Output :

5

Explanation :

Given list is [2,3,6,6,5]. The maximum score is 6 , second maximum is 5 . Hence, we print 5 as the runner-up score.

QUESTION NO 5 :

Create a function that takes two strings as arguments and return either True or False depending on whether the total number of characters .

in the first string is equal to the total number of characters in the second string.

Examples :

1) ("AB", "CD") ➞ True

2) ("ABC", "DE") ➞ False

3) ("hello", "edabit") ➞ False

QUESTION NO 6 :

PASSWORD CHECKER :

Create a function that validates a password to conform to the following rules:

- **Length between 6 and 24 characters.**
- **At least one uppercase letter (A-Z).**
- **At least one lowercase letter (a-z).**
- **At least one digit (0-9).**
- **Maximum of 2 repeated characters.**

"aa" is OK 👍

"aaa" is NOT OK 👎

Supported Special characters: ! @ # $ % ^ & * ( ) + = _ - { } [ ] : ; ” ’ ? < > , .

Examples :

1) validatePassword("P1zz@") ➞ false // too short

2) validatePassword("iMissYou") ➞ false // missing number

3) validatePassword("Fhg93@") ➞ true // OK

QUESTION NO 7 :

MAJORITY VOTE

Create a function that returns the majority vote in a list. A majority vote is an element that occurs > N/2 times in a list (where N is the length of the list).

Examples :

1) majority_vote(["A", "A", "B"]) ➞ "A"

2) majority_vote(["A", "A", "A", "B", "C", "A"]) ➞ "A

3) majority_vote(["A", "B", "B", "A", "C", "C"]) ➞ None

NOTE :

 # The frequency of the majority element must be strictly greater than 1/2.

 # If there is no majority element, return None.

 # If the list is empty, return None.

QUESTION NO 8 :

POSSIBLE PALINDROME

Create a class "possible_palindrom" and a class method "checking_palindrom" that determines whether it is possible to build a palindrome from the characters in a string.

Examples :

1) ("acabbaa") ➞ True

Can make the following palindrome: "aabcbaa"

2) ("aacbdbc") ➞ True

Can make the following palindrome: "abcdcba"

3) ("aacbdb") ➞ False

4) ("abacbb") ➞ False

QUESTION NO 9 :

PIZZA POINTS

Google is launching a network of autonomous pizza delivery drones and wants you to create a flexible rewards system (Pizza Points™) that can be tweaked in the future.

The rules are simple: if a customer has made at least N orders of at least Y price, they get a FREE pizza!

TASK :

Create a function that takes a dictionary of customers, a minimum number of orders and a minimum order price. Return a list of customers that are eligible for a free pizza.

EXAMPLES :

```
customers = { "Batman": [22, 30, 11, 17, 15, 52, 27, 12] ,
              "Spider-Man": [5, 17, 30, 33, 40, 22, 26, 10, 11, 45]
             }
```

1) pizza_points(customers, 5, 20) ➞ ["Spider-Man"]

2) pizza_points(customers, 3, 10) ➞ ["Batman", "Spider-Man"]

3) pizza_points(customers, 5, 100) ➞ []

CODE STRUCTURE :

```
def pizza_points(customers, min_orders, min_price):
```

QUESTION NO 10:

REMOVE DUPLICATES FROM A LIST

Create a function that takes a list of items, removes all duplicate items and returns a new list in the same sequential order as the old list (minus duplicates). **(SET built-in function is not allowed)**

Examples :

1) removeDups(["John", "Taylor", "John"]) ➞ ["John", "Taylor"]
2) removeDups([1, 0, 1, 0]) ➞ [1, 0]
3) removeDups(['The', 'big', 'cat']) ➞ ['The', 'big', 'cat']