ML Assignment

Day 0

In [1]:

```
# Read a full line of input from stdin and save it to our dynamically typed variable, inp
ut_string.
input_string = input()

# Print a string literal saying "Hello, World." to stdout.
print('Hello, World.')

# TODO: Write a line of code here that prints the contents of input_string to stdout.
print(input_string)
```

```
my name is riya
Hello, World.
my name is riya
```

Day 1

In [2]:

```
i = 4
d = 4.0
s = 'HackerRank '
# Declare second integer, double, and String variables.
b=3
c=7.0
e='Passed'

# Read and save an integer, double, and String to your variables.
f=int(input())
g=float(input())
h=str(input())
# Print the sum of both integer variables on a new line.
print(i+f)

# Print the sum of the double variables on a new line.
print(d+g)
# Concatenate and print the String variables on a new line
print(s+h)
# The 's' variable above should be printed first.
```

```
3
4
6
7
8.0
HackerRank 6
```

Day 2

In [3]:

```
import math
import os
import random
import re
import sys


def solve(meal_cost, tip_percent, tax_percent):
    # Write your code here
```

```
    meal_cost=float(meal_cost)
    tip_percent=int(tip_percent)
    tax_percent=int(tax_percent)

    tax=meal_cost*(tax_percent/100)
    tip=meal_cost*(tip_percent/100)
    return round(meal_cost + tax + tip)

if __name__ == '__main__':
    meal_cost = float(input().strip())

    tip_percent = int(input().strip())

    tax_percent = int(input().strip())

    print( solve(meal_cost, tip_percent, tax_percent))
```

```
500
5
5
550
```

Day 3

In [4]:

```
import math
import os
import random
import re
import sys



if __name__ == '__main__':

    n = int(input().strip())

# if 'n' is NOT evenly divisible by 2 (i.e.: n is odd)
    if n%2==1:
       ans = "Weird"

    elif n>20:
       ans = "Not Weird"

    elif n>=6:
       ans = "Weird"

    else:
       ans = "Not Weird"

    print(ans)
```

```
70
Not Weird
```

Day 4

In [ ]:

```
class Person:
    def __init__(self,initialAge):
        # Add some more code to run some checks on initialAge
        if(initialAge > 0):
            self.age = initialAge
        else:
            print("Age is not valid, setting age to 0.")
            self.age = 0

    def amIOld(self):
```

```
        if self.age >= 18:
            print("You are old.")
        elif self.age >= 13:
            print("You are a teenager.")
        else: # age < 13
            print("You are young.")

    def yearPasses(self):
        # Increment the age of the person in here
         self.age += 1

t = int(input())
for i in range(0, t):
    age = int(input())
    p = Person(age)
    p.amIOld()
    for j in range(0, 3):
        p.yearPasses()
    p.amIOld()
    print("")
```

```
5
18
You are old.
You are old.

45
You are old.
You are old.

19
You are old.
You are old.
```

Day 5

In [ ]:

```
import math
import os
import random
import re
import sys



if __name__ == '__main__':
    n = int(input().strip())
    for i in range(1, 11):
        print(str(n) +" x " + str(i) + " = " + str(n*i))
```

Day 6

In [2]:

```
import sys

def Even(s):
    l = len(s)
    output = ""
    for i in range(0,l,2):
        output += s[i]
    return output

def Odd(s):
    l = len(s)
    output = ""
    for i in range(1,l,2):
        output += s[i]
```

```
    return output

t = int(input())
for a0 in range(0,t):
    s = input()
    print(Even(s) + " " + Odd(s))
```

```
1
Pantagruel
Pnare atgul
```

Day 7

In [3]:

```
import math
import os
import random
import re
import sys



if __name__ == '__main__':
    n = int(input().strip())
    arr = list(map(int,input().rstrip().split(' ')))

    ans = ""
    for i in range(len(arr)-1 , -1, -1):
        ans += str(arr[i]) + " "

    print(ans)
```

```
3
9 5 3
3 5 9
```

Day 8

In [ ]:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
n = int(input())
phone_book = dict(input().split() for _ in range(n))

while True:
    try:
        query = input()
        print(f"{query}={phone_book[query]}")
    except KeyError:
        print('Not found')
    except EOFError:
        break
```

```
2
john 9999966666
kante 1236547899
joe
Not found
kante
kante=1236547899
```

Day 9

In [6]:

```
import math
import os
import random
import re
```

```
import sys

def factorial(n):
    if n<=1:
        return 1
    else:
        return n*factorial(n-1)

n = int(input())
print(factorial(n))
```

```
14
87178291200
```

`Day 10

In [7]:

```
import math
import os
import random
import re
import sys

def max(a,b):
    return a if a>b else b

n = int(input().strip())

max_num = 0
count = 0

while n:
    while n&1:
        count += 1
        n>>=1
    max_num = max(count, max_num)
    if not n&1:
        count = 0
        n>>=1

print(max_num)
```

```
14
3
```

Day 11

In [4]:

```
import math
import os
import random
import re
import sys



if __name__ == '__main__':
    arr = []
for arr_i in range(6):
   arr_temp = list(map(int,input().rstrip().split(' ')))
   arr.append(arr_temp)
max = 0

for i in range(0,4):
    for j in range(0,4):
        sum = 0
        sum= arr[i][j]+arr[i][j+1]+arr[i][j+2]+arr[i+1][j+1]+arr[i+2][j]+arr[i+2][j+1]+
arr[i+2][j+2]
```

```
        if i==0 and j==0:
            max = sum
        if sum > max:
            max =sum

print(max)
```

```
1 1 0 0 1 0
0 0 0 0 0 0
1 1 2 0 2 2
0 0 0 0 1 1
0 0 0 2 4 3
1 1 0 0 2 1
14
```

Day 12

In [10]:

```
class Person:
    def __init__(self, firstName, lastName, idNumber):
        self.firstName = firstName
        self.lastName = lastName
        self.idNumber = idNumber
    def printPerson(self):
        print("Name:", self.lastName + ",", self.firstName)
        print("ID:", self.idNumber)


class Student(Person):
    def __init__(self, fName, lName, sId, scores):
        super().__init__(fName, lName, sId)
        self.scores = scores

    def calculate(self):
        avg = 0.0
        for score in self.scores:
            avg += score

        avg = avg/len(self.scores)
        if avg < 40:
            return 'T'
        elif avg < 55:
            return 'D'
        elif avg < 70:
            return 'P'
        elif avg < 80:
            return 'A'
        elif avg < 90:
            return 'E'
        else:
            return 'O'

line = input().split()
firstName = line[0]
lastName = line[1]
idNum = line[2]
numScores = int(input()) # not needed for Python
scores = list( map(int, input().split()) )
s = Student(firstName, lastName, idNum, scores)
s.printPerson()
print("Grade:", s.calculate())
```

```
Riya Vaze 1401
98
89
Name: Vaze, Riya
ID: 1401
Grade: E
```

Day 13

In [6]:

```
from abc import ABCMeta, abstractmethod
class Book(object, metaclass=ABCMeta):
    def __init__(self,title,author):
        self.title=title
        self.author=author
    @abstractmethod
    def display(): pass

class MyBook(Book):
    def __init__(self, title, author, price):
        Book.__init__(self, title, author)
        self.price = price

    def display(self):
        print("Title: %s\nAuthor: %s\nPrice: %s" %(title, author, price))

title=input()
author=input()
price=int(input())
new_novel=MyBook(title,author,price)
new_novel.display()
```

```
Percy Jackson and The Greek Heroes
Rick Riordan
350
Title: Percy Jackson and The Greek Heroes
Author: Rick Riordan
Price: 350
```

Day 14

In [2]:

```
class Difference:
    def __init__(self, a):
        self.__elements = a

    def computeDifference(self):
        self.maximumDifference=max(self.__elements)-min(self.__elements)
        return None

# End of Difference class

d = Difference(a=[1,2,5])
d.computeDifference()

print(d.maximumDifference)
```

```
4
```

Day 15

In [3]:

```
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None
class Solution:
    def display(self,head):
        current = head
        while current:
            print(current.data,end=' ')
            current = current.next

    def insert(self,head,data):
        if head is None:
```

```
            head = Node(data)
        elif head.next is None:
            head.next = Node(data)
        else:
            self.insert(head.next, data)
        return head

mylist= Solution()
T=int(input())
head=None
for i in range(T):
    data=int(input())
    head=mylist.insert(head,data)
mylist.display(head);
```

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

Day 16

In [11]:

```
import sys


S = input().strip()
try:
    r = int(S)
    print(r)
except ValueError:
    print("Bad String")
```

```
rv
Bad String
```

Day 17

In [6]:

```
class Calculator(Exception):
    def power(self,n,p):
        if (n<0 or p<0):
            raise Calculator("n and p should be non-negative")
        else:
            return pow(n,p)

myCalculator=Calculator()
T=int(input())
for i in range(T):
    n,p = map(int, input().split())
    try:
        ans=myCalculator.power(n,p)
        print(ans)
    except Exception as e:
        print(e)
```

```
3
2 6
64
-1 3
n and p should be non-negative
3 4
81
```

Day 18

In [12]:

```
import sys
from collections import deque

class Solution:
    def __init__(self):
        self.stack = deque()
        self.queue = deque()

    def pushCharacter(self,char):
        self.stack.append(char)

    def popCharacter(self):
        return self.stack.pop()

    def enqueueCharacter(self,char):
        self.queue.append(char)

    def dequeueCharacter(self):
        return self.queue.popleft();

s=input()
obj=Solution()

l=len(s)
# push/enqueue all the characters of string s to stack
for i in range(l):
    obj.pushCharacter(s[i])
    obj.enqueueCharacter(s[i])

isPalindrome=True
'''
pop the top character from stack
dequeue the first character from queue
compare both the characters
'''
for i in range(l // 2):
    if obj.popCharacter()!=obj.dequeueCharacter():
        isPalindrome=False
        break

if isPalindrome:
    print("The word, "+s+", is a palindrome.")
else:
    print("The word, "+s+", is not a palindrome.")
```

```
mom
The word, mom, is a palindrome.
```

Day 19

In [13]:

```
class AdvancedArithmetic(object):
    def divisorSum(n):
        raise NotImplementedError

class Calculator(AdvancedArithmetic):
    def divisorSum(self, n):
        s = 0
        for i in range(1,n+1):
             if (n%i == 0):
                s+=i
        return s


n = int(input())
my_calculator = Calculator()
s = my_calculator.divisorSum(n)
```

```
print(s)
```

```
9
13
```

Day 20

In [14]:

```
import math
import os
import random
import re
import sys


if __name__ == '__main__':

    n = int(input().strip())
    a = list(map(int, input().rstrip().split(' ')))
    numberOfSwaps = 0
    for i in range(0,n):
        for j in range(0, n-1):
            if (a[j] > a[j + 1]):
                temp=a[j]
                a[j] = a[j+1]
                a[j+1] = temp
                numberOfSwaps += 1
        if (numberOfSwaps == 0):
            break
print( "Array is sorted in " + str(numberOfSwaps) + " swaps." )
print( "First Element: " + str(a[0]) )
print( "Last Element: " +  str(a[n-1]) )
```

```
5
1 6 9 3 6
Array is sorted in 3 swaps.
First Element: 1
Last Element: 9
```

Day 22

In [18]:

```
class Node:
    def __init__(self,data):
        self.right=self.left=None
        self.data = data
class Solution:
    def insert(self,root,data):
        if root==None:
            return Node(data)
        else:
            if data<=root.data:
                cur=self.insert(root.left,data)
                root.left=cur
            else:
                cur=self.insert(root.right,data)
                root.right=cur
        return root

    def getHeight(self,root):
        if root is None or (root.left is None and root.right is None):
            return 0
        else:
            return max(self.getHeight(root.left),self.getHeight(root.right))+1

T=int(input())
myTree=Solution()
root=None
```

```
for i in range(T):
    data=int(input())
    root=myTree.insert(root,data)
height=myTree.getHeight(root)
print("Height of tree: ",height)
```

```
8
4
3
2
6
7
1
8
9
Height of tree:  4
```

Day 23

In [20]:

```
import sys

class Node:
    def __init__(self,data):
        self.right=self.left=None
        self.data = data
class Solution:
    def insert(self,root,data):
        if root==None:
            return Node(data)
        else:
            if data<=root.data:
                cur=self.insert(root.left,data)
                root.left=cur
            else:
                cur=self.insert(root.right,data)
                root.right=cur
        return root

    def levelOrder(self,root):
        output = ""
        queue = [root]
        while queue:
            current = queue.pop(0)
            output += str(current.data) + " "
            if current.left:
                queue.append(current.left)
            if current.right:
                queue.append(current.right)
        print(output[:-1])

T=int(input())
myTree=Solution()
root=None
for i in range(T):
    data=int(input())
    root=myTree.insert(root,data)
myTree.levelOrder(root)
```

```
6
4
6
3
7
5
1
4 3 6 1 5 7
```

Day 24

In [21]:

```
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None
class Solution:
    def insert(self,head,data):
            p = Node(data)
            if head==None:
                head=p
            elif head.next==None:
                head.next=p
            else:
                start=head
                while(start.next!=None):
                    start=start.next
                start.next=p
            return head
    def display(self,head):
        current = head
        while current:
            print(current.data,end=' ')
            current = current.next

    def removeDuplicates(self,head):
        current = head
        while (current.next):
            if (current.data == current.next.data):
                current.next = current.next.next
            else:
                current = current.next

        return head

mylist= Solution()
T=int(input())
head=None
for i in range(T):
    data=int(input())
    head=mylist.insert(head,data)
head=mylist.removeDuplicates(head)
mylist.display(head);
```

```
7
3
2
2
5
4
4
1
3 2 5 4 1
```

Day 25

In [1]:

```
import math

def check_prime(num):
    if num == 1:
        return "Not prime"
    sq = int(math.sqrt(num))
    for x in range(2, sq+1):
        if num % x == 0:
            return "Not prime"
    return "Prime"


t = int(input())
```

```
for i in range(t):
    number = int(input())
    print(check_prime(number))
```

```
3
13
Prime
19
Prime
21
Not prime
```

Day 26

In [1]:

```
return_date= [int (i) for i in input().split()]
due_date= [int (i) for i in input().split()]
if return_date[2] > due_date[2]:
    print(10000)
else:
    if return_date[2] == due_date[2]:
        if return_date[1] > due_date[1]:
            print(500* (return_date[1] - due_date[1]))
        elif return_date[1] == due_date[1] and return_date[0] > due_date[0]:
            print(15* (return_date[0] - due_date[0]))
        else:
            print(0)
    else:
        print(0)
```

```
9 6 2019
2 6 2019
105
```

Day 27

In [2]:

```
def minimum_index(seq):
    if len(seq) == 0:
        raise ValueError("Cannot get the minimum value index from an empty sequence")
    min_idx = 0
    for i in range(1, len(seq)):
        if seq[i] < seq[min_idx]:
            min_idx = i
    return min_idx

class TestDataEmptyArray(object):

    @staticmethod
    def get_array():
        return []

class TestDataUniqueValues(object):

    def get_array():
        return [7, 4, 3, 8, 14]

    def get_expected_result():
        return 2

class TestDataExactlyTwoDifferentMinimums(object):

    def get_array():
        return [7, 4, 3, 8, 3, 14]

    @staticmethod
    def get_expected_result():
        return 2
```

```
def TestWithEmptyArray():
    try:
        seq = TestDataEmptyArray.get_array()
        result = minimum_index(seq)
    except ValueError as e:
        pass
    else:
        assert False


def TestWithUniqueValues():
    seq = TestDataUniqueValues.get_array()
    assert len(seq) >= 2

    assert len(list(set(seq))) == len(seq)

    expected_result = TestDataUniqueValues.get_expected_result()
    result = minimum_index(seq)
    assert result == expected_result


def TestiWithExactyTwoDifferentMinimums():
    seq = TestDataExactlyTwoDifferentMinimums.get_array()
    assert len(seq) >= 2
    tmp = sorted(seq)
    assert tmp[0] == tmp[1] and (len(tmp) == 2 or tmp[1] < tmp[2])

    expected_result = TestDataExactlyTwoDifferentMinimums.get_expected_result()
    result = minimum_index(seq)
    assert result == expected_result

TestWithEmptyArray()
TestWithUniqueValues()
TestiWithExactyTwoDifferentMinimums()
print("OK")
```

```
OK
```

Day 28

In [17]:

```
import math
import os
import random
import re
import sys



if __name__ == '__main__':

     N = int(input().strip())
     names = []
for a0 in range(N):
    firstName,emailID = input().rstrip().split(' ')
    firstName,emailID = [str(firstName),str(emailID)]
    match = re.search(r'[\w\.-]+@gmail.com', emailID)

    if match:
        names.append(firstName)
names.sort()
for name in names:
    print( name )
```

```
2
r r@gmail.com
v v@gmail.com
r
v
```

Day 29

In [4]:

```
t = int(input().strip())
for a0 in range(t):
    n, k = input().strip().split(' ')
    n, k = [int(n), int(k)]
    print(k-1 if ((k-1) | k) <= n else k-2)
```

```
3
5 2
1
8 5
4
2 2
0
```