Machine Learning(ML): Assignment 1

Charvi Dave J015

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)
```

```
charvi
Hello, World.
charvi
```

Day 1

In [2]:

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

# Read and save an integer, double, and String to your variables.
j = int(input())
o = float(input())
g = input()

# Print the sum of both integer variables on a new line.
print(a+j)

# Print the sum of the double variables on a new line.
print(b+o)

# Concatenate and print the String variables on a new line
print(c+g)
# The 's' variable above should be printed first.
```

```
5
7
11
9
11.0
HackerRank 11
```

Day 2

In [3]:

```
mealCost = float(input())
tip = int(input())
tax = int(input())
```

```
tip=tip*mealCost/100;
tax=tax*mealCost/100;
totalcost=mealCost+tip+tax;

print ("The total meal cost is %s dollars." %str(int(round(totalcost, 0))))
```

```
21
3
4
The total meal cost is 22 dollars.
```

Day 3

In [4]:

```
import sys


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

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

elif n>20:
    answer = "Not Weird"

elif n>=6:
    answer = "Weird"

else:
    answer = "Not Weird"

print(answer)
```

```
7
Weird
```

Day 4

In [5]:

```
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):
        # Do some computations in here and print out the correct statement to the console
        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

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

```
3
7
You are young.
You are young.

19
You are old.
You are old.

11
You are young.
You are a teenager.
```

Day 5

In [6]:

```
import sys


N = int(input().strip())
for i in range(1, 11):
    print(str(N) +" x " + str(i) + " = " + str(N*i))
```

```
8
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
```

Day 6

In [7]:

```
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

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

```
3
6
6
```

```
7
7
1
1
```

Day 7

In [11]:

```
import sys


p = int(input().strip())
arr = list(map(int,input().rstrip().split(' ')))
answer = ""
for i in range(len(arr)-1 , -1, -1):
    answer += str(arr[i]) + " "

print(answer)
```

```
3
2 56 7
7 56 2
```

Day 8

In [12]:

```
import sys
inputList=[]
num=int(input("Enter number of phone numbers: "))
for i in range(num):
    inputList.append(input("Enter number: "))

entries=inputList
phoneBook={}
for entry in entries:
    name,p_number=entry.split()
    phoneBook[name]=p_number
while True:
    query=input("Enter query: ")
    if query.lower()=="done":
        break
    else:
        stripQuery=query.rstrip() #Eliminates the newline character
        if stripQuery in phoneBook:
            print(stripQuery+"="+str(phoneBook[stripQuery]))
        else:
            print("Not found")
```

```
Enter number of phone numbers: 2
Enter number: charvi 9848494040
Enter number: hetvi 3739505544
Enter query: abel
Not found
Enter query: charvi
charvi=9848494040
Enter query: done
```

Day 9

In [13]:

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

```
n = int(input("Enter a number: "))
print(factorial(n))
```

```
Enter a number: 5
120
```

Day 10

In [16]:

```
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)
```

```
72
1
```

Day 11

In [19]:

```
import sys


arr = []
for arr_i in range(6):
   arr_temp = list(map(int,input().strip().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 1 1 1 1
2 2 2 2 2 2
3 1 2 1 2 3
1 0 0 1 1 1
4 1 1 2 2 1
1 1 3 3 3 1
13
```

Day 12

In [20]:

```
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())
```

```
Charvi Dave 1015
97
87
Name: Dave, Charvi
ID: 1015
Grade: E
```

Day 13

In [21]:

```
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("Enter the title of the book: ")
author=input("Enter the author of the book: ")
price=int(input("Enter the price of the book: "))
new_novel=MyBook(title,author,price)
new_novel.display()
```

```
Enter the title of the book: We were liars
Enter the author of the book: Erin Lockhart
Enter the price of the book: 340
Title: We were liars
Author: Erin Lockhart
Price: 340
```

Day 14

In [7]:

```
import numpy as np
class Difference:
    def __init__(self,a):
        self.elements=a
        self.max_diff=0
    def calc_diff(self):
        self.max_diff=np.max(a)-np.min(a)
        return self.max_diff
a=[int(e) for e in input("Enter numbers: ").split(' ')]
diff=Difference(a)
print(diff.calc_diff())
```

```
Enter numbers: 2 7 8 11 15
13
```

Day 15

In [25]:

```
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);
```

```
4
20
1
```

```
45
3
20 1 45 3
```

Day 16

In [26]:

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

```
cd
Bad String
```

Day 17

In [27]:

```
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)
```

```
2
7 11
1977326743
2 7
128
```

Day 18

In [28]:

```
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();
```

```
# read the string s
s=input()
#Create the Solution class object
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
#finally print whether string s is palindrome or not.
if isPalindrome:
    print("The word, "+s+", is a palindrome.")
else:
    print("The word, "+s+", is not a palindrome.")
```

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

Day 19

In [29]:

```
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("I implemented: " + type(my_calculator).__bases__[0].__name__)
print(s)
```

```
7
I implemented: AdvancedArithmetic
8
```

Day 20

In [9]:

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

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

```
a=list(map(int,input().rstrip().split()))
no_swaps=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
            no_swaps+=1
    if no_swaps==0:
        break
print(f"Array is sorted in {str(no_swaps)} swaps")
print(f"First Element: {str(a[0])}")
print(f"Last Element: {str(a[n-1])}")
```

```
3
5 4 3
Array is sorted in 3 swaps
First Element: 3
Last Element: 5
```

Day 22

In [1]:

```
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("Enter a number: "))
myTree=Solution()
root=None
for i in range(T):
    data=int(input())
    root=myTree.insert(root,data)
height=myTree.getHeight(root)
print("Height of the tree is: ",height)
```

```
Enter a number: 3
1
7
6
Height of the tree is:  2
```

Day 23

In [2]:

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

```
Enter a number here: 5
7
11
42
223
43
7 11 42 223 43
```

Day 24

In [3]:

```
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
3
4
1
65
8
8
3 4 1 65 8
```

Day 25

In [1]:

```
import math

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


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

```
7
2
Prime
3
Prime
80
Not prime
9
Not prime
11
Prime
5
Prime
6
Not prime
```

Day 26

In [2]:

```
da, ma, ya = input().split(' ')
```

```
da, ma, ya = input().split(' ')
da = int(da)
ma = int(ma)
ya = int(ya)
de, me, ye = input().split(' ')
de = int(de)
me = int(me)
ye = int(ye)
fine = 0
if(ye==ya):
    if(me < ma):
        fine = (ma - me) * 500
    elif((me == ma) and (de < da)):
        fine = (da - de) * 15
elif(ye < ya):
    fine = 10000

print( fine )
```

```
21 72 31
22 56 45
0
```

Day 27

In [3]:

```
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

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):

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

    @staticmethod
    def get_expected_result():
        return 2

class TestDataExactlyTwoDifferentMinimums(object):

    @staticmethod
    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 [4]:

```
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().strip().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
charvi charvi@gmail.com
hetvi hetvi@gmail.com
hetvi
```

Day 29

In [5]:

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

#
# Complete the 'bitwiseAnd' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER N
#  2. INTEGER K
#

def bitwiseAnd(N, K):
    import math
import os
import random
import re
import sys

#
# Complete the 'bitwiseAnd' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER N
#  2. INTEGER K
#

def bitwiseAnd(N, K):
    if __name__ == '__main__':
        fptr = open(os.environ['OUTPUT_PATH'], 'w')

    t = int(input().strip())

    for t_itr in range(t):
        first_multiple_input = input().rstrip().split()

        count = int(first_multiple_input[0])

        lim = int(first_multiple_input[1])

        res = bitwiseAnd(count, lim)

        fptr.write(str(res) + '\n')

        fptr.close()
```

In [6]:

```
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)
```

```
2
4 7
5
23 11
10
```

In [ ]:

```

```