Ishan Jhanwar J024

In [1]:
```
import math
import os
import random
import re
import sys
```

Day 0

In [23]:
```
input_string=input()
print("Hello World.\n",input_string)
```

```
lol
Hello World.
 lol
```

Day 1

In [4]:
```
i = 4
d = 4.0
s = 'HackerRank '
a=2
b=3.5
c='pro'
```

In [5]:
```
print(i+a)
print(d+b)
print(s+c)
```

```
6
7.5
HackerRank pro
```

Day 2

In [12]:
```
    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))))
```

```
12
20
8
The total meal cost is 15 dollars.
```

Day 3

In [16]:

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



if n%2==1:
    ans = "Weird"

elif n>20:
    ans = "Not Weird"

elif n>=6:
    ans = "Weird"

else:
    ans = "Not Weird"

print(ans)
```

```
4
Not Weird
```

Day 4

In [27]:

```python
class Person:
    def __init__(self,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:
        print("You are young.")
```

```
def yearPasses(self):


    self.age += 1
```

Day 5

In [2]:

```
n = int(input("Enter your number"))
for i in range(1, 11):
 print(f'{n} x {i} = {n*i}')
```

```
Enter your number7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

Day 6

In [4]:

```
s = input('>>> ')
output = [s[i] for i in range(0, len(s), 2)] + [' '] + [s[i] for i in
range(1, len(s), 2)]
print("".join(output))
```

```
>>> abc
ac b
```

Day 7

In [5]:

```
A = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for i in A[::-1]:
    print(i, end=" ")
print()
```

```
9 8 7 6 5 4 3 2 1 0
```

Day 8

In [ ]:

```
phonebook = {}
for i in range(int(input('>>>'))):
 inp = input().split()
 phonebook[inp[0]] = inp[1]
q = input('Query: ')
while q != "":
    try:
```

```
        print(f'{q}={phonebook[q]}')
    except:
        print('Not Found!')
    q = input('Query: ')
```

Day 9

In [23]:

```
def rec_fact(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * rec_fact(n - 1)
print(rec_fact(int(input('>>>'))))
```

```
>>>4
24
```

Day 10

In [ ]:

```
n = int(input('>>>'))
binary = ''
while n > 0:
    binary = str(n % 2) + binary
n //= 2
print(max([len(i) for i in binary.split('0')]))
```

Day 11

In [ ]:

```
def get_mask(i, j):
 ret = [(i, x) for x in range(j, j+3)]
 ret += [(i+1, j+1)]
 ret += [(i+2, x) for x in range(j, j+3)]
 return ret
A = [[1, 1, 1, 0, 0, 0,],
 [0, 1, 0, 0, 0, 0,],
 [1, 1, 1, 0, 0, 0,],
 [0, 0, 2, 4, 4, 0,],
 [0, 0, 0, 2, 0, 0,],
 [0, 0, 1, 2, 4, 0,],]
max_ = -1
for i in range(0, len(A) - 2):
    for j in range(len(A[0]) - 2):
        to_check = get_mask(i, j)
        sum_ = sum([A[i][j] for i, j in to_check])
```

```
        if sum_ > max_:
            max_ = sum_
print(max_)
```

Day 12

In [ ]:

```
class Student:
     def __init__(self, first_name, last_name, id_number, scores):
            self.first_name = first_name
            self.last_name = last_name
            self.id_number = id_number
            self.scores = scores
    def calculate(self):
        d = {
 'T': 40,
 'D': 55,
 'P': 70,
 'A': 80,
 'E': 90,
 'O': 100,

 }

    avg = sum(self.scores) / len(self.scores)

    for grade, score_threshold in d.items():
        if avg < score_threshold:
            return grade
mike = Student('mike', 'ronald', 123456, [100, 99, 89, 91])
print(f'Name: {mike.first_name} {mike.last_name}')
print(f'ID: {mike.id_number}')
print(f'Grade: {mike.calculate()}')
```

Day 13

In [ ]:

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


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

```
    Book.__init__(self, title, author)
    self.price = price

    def display(self):
    print(f'Title: {self.title}\nAuthor: {self.author}\nPrice:
{self.price}')


my_book1 = MyBook('The Alchemist', 'Paulo Coelhi', 1000)
my_book1.display()
```

Day 14

In [ ]:

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

    def maximum_difference(self):
    return max(self.elements_) - min(self.elements_)


diff = Difference([1, 2, 3, 4, 5, 6, 7, 8, 9])
diff.maximum_difference()
```

Day 15

In [ ]:

```
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)
```

Day 16

In [ ]:

```
S = input()
try:
    print(int(S))
except:
    print("Bad String")
```

Day 17

In [ ]:

```
class calculator:
    def power(self, n, p):
    if n < 0 or p < 0:
        print("n and p should be non-negative")
    else:
        print(n ** p)
```

Day 18

In [2]:

```
class Solution:


    def dequeue(self):
    try:
        x = self.queue[0]
    self.queue = self.queue[1:]
    return x
    except:
    return None
```

```
sol = Solution()
S = 'naaakaaan'
for c in S:
    sol.push_char(c)
    sol.enqueue_char(c)
flag = True
while True:
    from_stack = sol.pop_char()
    from_queue = sol.dequeue()

    if from_stack == None:
        break

    if from_stack != from_queue:
        flag = False
        break



if flag:
    print("Palindrome")
else:
    print("Not Palindrome")
```

```
  File "<ipython-input-2-6a5e6dc27fe9>", line 4
    try:
    ^
IndentationError: expected an indented block
```

Day 19

In [1]:

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



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

Day 20

In [ ]:

```
if name ==  '  main  ':
    n = 9
    a = [8, 9, 1, 3, 4, 2, 7, 5, 6]

    num_swaps = 0
    for i in range(n):
        flag = True
    for j in range(n-1):
        if a[j] > a[j+1]:
            a[j], a[j+1] = a[j+1], a[j]
            flag = False
            num_swaps += 1
    if flag:
        break

print(
    f'Array is sorted in {num_swaps} swaps\nFirst Element {a[0]}\nLast
Element {a[1]}')
```

Day 21

In [ ]:

```
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 == None or root.left == None and root.right == None:
        return 0
    else:
```

```
        return 1 + max(self.getHeight(root.left),
self.getHeight(root.right))

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)
```

Day 23

In [ ]:

```
def levelOrder(self, root):
    ret = ""



queue = [root]
while queue:
current = queue.pop(0)
ret += str(current.data) + " "
if current.left:
    queue.append(current.left)
if current.right:
    queue.append(current.right)
print(ret[:-1])
```

Day 24

In [ ]:

```
def removeDuplicates(self,head):
    #Write your code here
    current = head
    while (current.next):
        if (current.data == current.next.data):
            current.next = current.next.next
        else:
            current = current.next

    return head
```

Day 25

In [ ]:
```
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))
```

Day 26

In [ ]:
```
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 )
```

Day 27

In [ ]:
```
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
```

Day 28

In [ ]:

```
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 )
```

Day 29

In [ ]:

```
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)
```