Assignment 1: 30 Days of Code (HackerRank)

Yash Shah J060

Day 0

```
inputString = input()

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

```
Yash Shah Here!
Hello, World.
Yash Shah Here!
```

Day 1

```
i =2
d = 2.3
s = "5"

j = int(input())
e = float(input())
t = input()

print(i+j)
print(d+e)
print(s+t)
```

```
3
3.2
9
5
5.5
59
```

Day 2

```
def solve(mealCost, tip, tax):
    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))))

if __name__ == '__main__':
    mealCost = float(input().strip())
    tip = int(input().strip())
    tax = int(input().strip())

    solve(mealCost, tip, tax)
```

```
40
2
```

```
9
The total meal cost is 44 dollars.
```

Day 3

```
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

```
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: # age < 13
            print("You are young.")

    def yearPasses(self):
        self.age += 1
```

Day 5

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

```
10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

Day 6

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

def printOddIndexChar(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(printEvenIndexChar(s) + " " + printOddIndexChar(s))
```

```
5
5
5
10
1 0
11
1 1
1922
12 92
3
3
```

Day 7

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

print(ans)
```

```
5
4 3 5 6 9
9 6 5 3 4
```

Day 8

Day 9

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

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    n = int(input().strip())
    result = factorial(n)
    fptr.write(str(result) + '\n')
    fptr.close()
```

Day 10

```
def convert_binary(num):
    bin=[]
    consec=0
    arr=[]


    #converting to binary
    while int(num/2>0):

        bin.append(int(num%2))
        num=int(num/2)
    bin.append(0)

    for i in range(0,len(bin)):
        if bin[i]==1:
            consec=consec+1
        elif bin[i]==0:
            arr.append(consec)
            consec=0

    return max(arr)

if __name__ == '__main__':
    n = int(input().strip())

    print(convert_binary(n))
```

```
13
2
```

Day 11

Day 12

```
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,first,last,id,scores):
        super().__init__(first, last,id)
        self.scores=scores

    def calculate(self):

        #average score
        average= sum(self.scores)/len(self.scores)

        #grades
        if 90<=average<=100:
            return 'O'
        elif 80<=average<90:
            return 'E'
        elif 70<=average<80:
            return 'A'
        elif 55<=average<70:
            return 'P'
        elif 40<=average<55:
            return 'D'
        else:
            return 'T'



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())
```

```
Heraldo Memelli 8135627
2
100 80
Name: Memelli, Heraldo
ID: 8135627
Grade: O
```

Day 13

```
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

#Write MyBook class
class MyBook(Book):

    def __init__(self, title, author, price):
        super().__init__(title, author)
        self.price = price
    def display(self):
        print(f'Title: {self.title}')
        print(f'Author: {self.author}')
        print(f'Price: {self.price}')
```

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

```
The Alchemist
Paulo Coelho
248
Title: The Alchemist
Author: Paulo Coelho
Price: 248
```

Day 14

```
class Difference:
    def __init__(self, a):
        self.__elements = a
    maximumDifference=0


    def computeDifference(self):
        self.__elements.sort()
        self.maximumDifference=self.__elements[-1]-self.__elements[0]

_ = input()
a = [int(e) for e in input().split(' ')]

d = Difference(a)
d.computeDifference()

print(d.maximumDifference)
```

Day 15

```
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):
    #Complete this method
        newNode=Node(data)
        if head == None:
            head=newNode
            return head
        else:
            current=head
            while current.next != None:
                current=current.next
            current.next=newNode
            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

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


if __name__ == '__main__':
    S = input()

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

```
yash
Bad String
```

Day 17

```
#Write your code here
class Calculator():
    def power(self,n,p):
        if n>=0 and p>=0:
            return pow(n,p)
        else:
            return  "n and p should be non-negative"


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)
```

Day 18

```
import sys
class Solution:
    # Write your code here
    def __init__(self):
        self.stack = []
        self.queue = []

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

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

    def enqueueCharacter(self, x):
        self.queue.insert(0, x)

    def dequeueCharacter(self):
        return self.queue.pop()




# 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.")
```

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

Day 19

Day 20

```
if __name__ == '__main__':
    n = int(input().strip())

    a = list(map(int, input().rstrip().split()))

    #number of tracks swapped during a single array traversal
    numberofSwaps=0

    for i in range(0,len(a)-1):

        for j in range(0,len(a)-1):
            if a[j]>a[j+1]:
                a[j+1], a[j] = a[j], a[j+1]
                numberofSwaps= numberofSwaps + 1

        #if no elements swapped during traversal, array is sorted
        if numberofSwaps==0:
            break

    print(f'Array is sorted in {numberofSwaps} swaps.')
    print(f'First Element: {a[0]}')
    print(f'Last Element: {a[-1]}')
```

```
12
2
Array is sorted in 0 swaps.
First Element: 2
Last Element: 2
```

Day 21

Day 22

```
class node:
  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())
```

```
5
```

Day 23

```
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])
```

Day 24

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

    return head
```

Day 25

```
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))
```

```
5
3
Prime
10
Not prime
9
Not prime
2
Prime
111
Not prime
```

Day 26

```
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 )
```

```
4 5 5
6 5 1
10000
```

Day 27

```
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

Day 29

```
import sys


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 9
7
65 89
87
25 11
10
```

✓ 26s completed at 9:22 AM