ASSIGNMENT-1

Ahnaan Merchant J036

In [1]:
```
#problem 0
n=input()
print("Hello World")
print(n)
```

```
Welcome to 30 days of code
Hello World
Welcome to 30 days of code
```

In [1]:
```
#problem 1
#python doesnt have double data type .

a=int(input()) #int
b=float(input()) #no double in python so float
c=str(input()) #str
s='hackerank'
print(a+b)
print(a-b)
print(s+' '+'c')
```

In [37]:
```
#problem 2
a=int(input('Meal Amount:'))
b=int(input('Tip %:'))
c=int(input('Tax %:'))

def calcPercent(x,y):
    z=(x*y)/100
    return z

print('Total Amount:',calcPercent(b,a)+calcPercent(c,a)+a)
```

```
Meal Amount:12
Tip %:20
Tax %:8
Total Amount: 15.36
```

In [7]:

```
#problem 3
n=int(input('Enter:'))

if n%2!=0:
    print('Weird')
else:
    if n>=2 and n<=5:
        print('Not Weird')
    elif n>=6 and n<=20:
        print('Weird')
    else:
        print('Not Weird')
```

```
Enter:24
Not Weird
```

In [10]:

```
#problem 4
class Person:
    def __init__(self,initialAge):
        self.age=initialAge
        if self.age<0:
            self.age=0
            print('Age is not valid,Setting Age to 0')
        else:
            pass

    def amiold(self):
        if self.age>=18:
            print('You are old')
        if self.age<18 and self.age>=13:
            print('You are a teenager')
        if self.age<13:
            print('You are young')

    def yearincrease(self):
        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.yearincrease()
    p.amiold()
    print("")
```

```
4
-1
Age is not valid,Setting Age to 0
You are young
You are young

10
You are young
You are a teenager

16
You are a teenager
You are old

118
You are old
You are old
```

In [15]:

```
#problem 5
n=int(input('Enter Number:'))

for i in range(1,11):
    print(str(n)+'x'+str(i)+'=',i*n)
```

```
Enter Number:2
2x1= 2
2x2= 4
2x3= 6
2x4= 8
2x5= 10
2x6= 12
```

```
2x7= 14
2x8= 16
2x9= 18
2x10= 20
```

In [19]:

```
#problem 6
s=input('Enter:')
s=list(s)
s1=[]
s2=[]
for i in range(0,len(s),2):
    s1.append(s[i])
for i in range(1,len(s),2):
    s2.append(s[i])
res1=''
res1=res1.join(s1)
res2=''
res2=res2.join(s2)

print(res1)
print(res2)
```

```
Enter:hacker
hce
akr
```

In [27]:

```
#problem 7
n=input('Enter array sep by space:')
n=n.split(' ')
n=[int(i) for i in n]
res=[]
res=list(reversed(n))
res=[str(i) for i in res]
s=' '
s=s.join(res)
print(s)
```

```
Enter array sep by space:1 2 3 4
4 3 2 1
```

In [30]:

```
#problem 8
```

```
t=int(input('Number of Entries:'))
data={}
for i in range(t):
    n=input('Enter Details:')
    n=n.split()
    data[str(n[0])]=int(n[1])

n1=input('Enter Name:')

if n1 in data:
    print(data[n1])
else:
    print('Invalid entry')
```

```
Number of Entries:3
Enter Details:aari 1234
Enter Details:stuu 4321
Enter Details:xyze 1324
Enter Name:aum
Invalid entry
```

In [34]:

```
#problem 9
n=int(input('Enter Number:'))
def calcFactorial(n):
    if n==1:
        return n
    else:
        return n*calcFactorial(n-1)
print(calcFactorial(n))
```

```
Enter Number:3
6
```

In [53]:

```
#problem 10
n=int(input('Enter Number:'))

def function(x):
    count = 0
    while (x!=0):
        x = (x & (x << 1))
        count=count+1
    return count

print(function(n))
```

```
Enter Number:3
2
```

In [2]:

```
#problem 11
arr=[]

for i in range(6):
    arr.append(list(map(int,input().rstrip().split())))


def find_max_sum(arr):
    for i in range(4):
        for j in range(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
    return max
print(find_max_sum(arr))
```

```
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
19
```

In [9]:

```
#problem 12

class Person:
    def __init__(self,firstname,lastname,idnum):
        self.firstname=firstname
        self.lastname=lastname
        self.idnum=idnum
    def printdetails(self):
        print("Name:",self.lastname+",",self.firstname)

class Student(Person):
    def __init__(self,firstname,lastname,idnum,scores):
        self.firstname=firstname
        self.lastname=lastname
        self.idnum=idnum
        self.scores=scores
    def calculate(self):
        avg=sum(scores)/len(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'
        elif avg <100:
            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)
print('+'*10,'OUTPUT',"+"*10)
s.printdetails()
print("Grade:", s.calculate())
```

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

In [11]:
```
#problem 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

class MyBook(Book):
    def __init__(self,title,author,price):
        self.title=title
        self.author=author
        self.price=price
    def display(self):
        print("Title: ",title)
        print("Author: ",author)
        print("Price: ",price)

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

```
The Alchemist
don pablo
248
Title:  The Alchemist
Author:  don pablo
Price:  248
```

In [17]:
```
#problem 14
class Difference:
    def __init__(self, a):
        self.__elements = a
    def computeDifference(self):
        res=0
        maxval=0
        for i in range(len(a)):
            for j in range(i,len(a)):
                res=a[i]-a[j]
                res=abs(res)

                if res>maxval:
                    maxval=res
                else:
                    pass
        return maxval

# End of Difference class

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

d = Difference(a)

print(":OUTPUT:")
print(d.computeDifference())
```

```
3
1 2 5
:OUTPUT:
4
```

In [19]:

```
#problem 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):
        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
2
4
3
1
2 4 3 1
```

In [20]:
```
#problem 16

if __name__ == '__main__':
    S = input()
    try:
        s=int(s)
        print(s)
    except:
        print("Bad String")
```

```
za
Bad String
```

In [25]:
```
#problem 17
class Calculator(Exception):
    def power(self,n,p):
        if n > 0 and p >0:
            return n**p
        else:
            raise Calculator("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)
```

```
4
2 5
32
3 2
9
-2 -4
n and p should be non negative
-2 8
n and p should be non negative
```

In [26]:

```
#problem 18
import sys

class Solution:
    def __init__(self):
        self.stack=[]
        self.queue=[]
    def pushCharacter(self,a):
        self.stack.append(a)
    def popCharacter(self):
        return self.stack.pop()
    def enqueueCharacter(self,a):
        self.queue.append(a)
    def dequeueCharacter(self):
        return self.queue.pop(0)

# 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.")
```

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

In [29]:

```
#problem 19
class AdvancedArithmetic(object):
    def divisorSum(n):
        raise NotImplementedError

class Calculator(AdvancedArithmetic):
    def divisorSum(self, n):
        res=[]
        for i in range(1,n+1):
            if n%i==0:
                res.append(i)
        return sum(res)


n = int(input())
my_calculator = Calculator()
s = my_calculator.divisorSum(n)
print("I implemented: " + type(my_calculator).__bases__[0].__name__)
print(s)
```

```
6
I implemented: AdvancedArithmetic
12
```

In [34]:
```
#problem 20

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

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

    def bubblesort(n,a):
        numberOfSwaps=0
        for i in range(0,n):
            for j in range(0,n-1):
                if a[j]>a[j+1]:
                    a[j],a[j+1]=a[j+1],a[j]
                    numberOfSwaps+=1
        print("Array is sorted in ",numberOfSwaps," swaps.")
        print("First Element:",a[0])
        print("Last Element:",a[-1])

    bubblesort(n,a)
```

```
3
3 2 1
Array is sorted in  3  swaps.
First Element: 1
Last Element: 3
```

In [35]:
```
#problem 21

#solution not in python
```

In [10]:
```
#problem 22
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('OUTPUT:\n',height)
```

```
7
3
5
2
4
1
6
7
OUTPUT:
 3
```

In [12]:

```
#problem 23
import sys

class Node:
```

```python
    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)
print('OUTPUT:')
myTree.levelOrder(root)
```

```
6
3
5
4
7
```

```
2
1
OUTPUT:
3 2 5 1 4 7
```

In [15]:

```
#problem 24
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)
print('OUTPUT:')
mylist.display(head);
```

```
6
1
2
2
3
3
4
OUTPUT:
1 2 3 4
```

In [28]:

```
#problem 25
n=int(input('Enter a number:'))

def is_prime(n):
    if n==1:
        return 'Not Prime'
    else:
        if n%2==0:
            return 'Not Prime'
        else:
            for i in range(2,n):
                if n%i==0:
                    return 'Not Prime'
        return 'Prime'
print(is_prime(n))
```

```
Enter a number:19
Prime
```

In [30]:

```
#problem 26
dd,mm,yyyy=input('Returned Date>').split(' ')
dd,mm,yyyy=int(dd),int(mm),int(yyyy)

dd1,mm1,yyyy1=input('Due Date>').split(' ')
```

```
dd1,mm1,yyyy1=int(dd1),int(mm1),int(yyyy1)

fine = 0
if(yyyy==yyyy1):
    if(mm1 < mm):
        fine = (mm - mm1) * 500
    elif((mm1 == mm) and (dd1 < dd)):
        fine = (dd - dd1) * 15
elif(yyyy1 < yyyy):
    fine = 10000
print('Fine is:',fine,' Hackos')
```

```
Returned Date>9 6 2016
Due Date>6 6 2015
Fine is: 10000  Hackos
```

In [31]:

```
#problem 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


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
```

In [32]:
```
#problem 28


import re


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
aaryansh sahayaaryansh2001@gmail.com
aari aariaari@gmail.com
aari
aaryansh
```

In [34]:
```
#problem 29

t = int(input().strip())
for _ 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
5 5
4
```