In [1]:

```
#1
s = input("enter the input")
```

```
enter the inputWelcome to 30 Days of Code!
```

In [2]:

```
print("Hello World\n",s)
```

```
Hello World
 Welcome to 30 Days of Code!
```

In [3]:

```
#2
i = 4
d = 4.0
s = "HackerRank"
```

In [6]:

```
inp1 = int(input("enter an integer : "))
inp2 = float(input("enter a double value : "))
inp3 = input("enter a string : ")
```

```
enter an integer : 14
enter a double value : 2.0
enter a string : is the best place to learn and practice coding!
```

In [7]:

```
print(inp1 + 4)
print(inp2 + 4.0)
print(s + " " + inp3)
```

```
18
6.0
HackerRank is the best place to learn and practice coding!
```

In [8]:

```
#3
meal_cost = float(input("Enter meal cost : "))
tip = int(input("Enter the tip percentage : "))
tax = int(input("Enter the tax percentage : "))
```

```
Enter meal cost : 200
Enter the tip percentage : 12
Enter the tax percentage : 6
```

In [9]:

```
total_meal_cost = meal_cost + (tip*meal_cost/100) + (tax*meal_cost/100)
print("The total meal cost is : ", round(total_meal_cost))
```

```
The total meal cost is :  236
```

In [10]:

```
#4
n = int(input("Enter a value : "))
```

```
Enter a value : 3
```

In [12]:

```
if n%2==0 :
```

```
    if n>2 and n<5 :
        print("Not Wierd")
    elif n>6 and n<20 :
        print("Wierd")
    elif n>20 :
        print("Not Wierd")
else:
    print("Wierd")
```

```
Wierd
```

In [14]:

```
#5
class Person :

    def __init__(self, InitialAge) :
        if InitialAge>0 :
            self.age = InitialAge
        else :
            age = 0
            print("Age is invalid")

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

    def amIOLD(self) :
        if age<13 :
            print('You are young')
        elif age>=13 and age<=18 :
            print("you are a teenager")
        else:
            print("You are old")
```

In [21]:

```
#6
n = int(input("Enter a number : "))
```

```
Enter a number : 3
```

In [22]:

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

```
3  x  1  =  3
3  x  2  =  6
3  x  3  =  9
3  x  4  =  12
3  x  5  =  15
3  x  6  =  18
3  x  7  =  21
3  x  8  =  24
3  x  9  =  27
3  x  10  =  30
```

In [23]:

```
#7
def printnewformat(s):
    l = len(s)
    output1 = ""
    output2 = ""
    for i in range(0,l,2):
        output1 += s[i]
    for i in range(1,l,2):
        output2 += s[i]
    print(output1, " ", output2)
```

In [24]:

```
t = int(input("Enter number of test cases : "))
for i in range(t):
    s = input("enter the string : ")
    printnewformat(s)
```

```
Enter number of test cases : 2
enter the stringHacker
Hce   akr
enter the stringRank
Rn   ak
```

In [30]:

```
#8
A = [1, 2, 3, 4]
out = ""
for i in range(3,-1,-1):
    out = out + str(A[i]) + " "
print(out)
```

```
4 3 2 1
```

In [1]:

```
#9
def factorial(n):
    if n<=1:
        return 1
    else:
        return n*factorial(n-1)

n = int(input("enter the number : "))
print(factorial(n))
```

```
6
720
```

In [10]:

```
#10
n = int(input("Enter a number : "))
binary = bin(n)
binary = binary[2:]
print(binary)
maxnum = 0
count = 0
for i in range(len(binary)) :
    if binary[i]=='1' :
        count += 1
        maxnum = max(count, maxnum)
    else:
        count = 0
print(maxnum)
```

```
Enter a number : 13
1101
2
```

In [11]:

```
#11
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 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 [12]:

```
#12
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'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-17dad00b5b6e> in <module>
      1 #12
----> 2 class Student(Person):
      3     def __init__(self, fName, lName, sId, scores):
      4         super().__init__(fName, lName, sId)
      5         self.scores = scores

NameError: name 'Person' is not defined
```

In [ ]:

```
#13
class MyBook(Book):
    def __init__(self, title, author, price):
        super().__init__(self, title, author)
        self.price = price

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

In [23]:

```
#14
class Difference:
    def __init__(self, arr):
        self.elements=arr
```

```
        self.maximumDifference = 0

    def computeDifference(self):
        self.maximumDifference = max(self.elements)-min(self.elements)
```

In [5]:

```
#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("enter the number of elements : "))
head=None
for i in range(T):
    data=int(input("enter element %d : " %(i+1)))
    head=mylist.insert(head,data)
mylist.display(head);
```

```
enter the number of elements : 5
enter element 1 : 4
enter element 2 : 6
enter element 3 : 7
enter element 4 : 3
enter element 5 : 2
4 6 7 3 2
```

In [7]:

```
#16
S = input().strip()
try:
    N = int(S)
    print(N)
except ValueError:
    print("Bad String")
```

```
Z
Bad String
```

In [8]:

```
#17
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 = int(input("enter the number : "))
    p = int((input("enter the power : ")))
    try:
```

```
        ans=myCalculator.power(n,p)
        print(ans)
    except Exception as e:
        print(e)
```

```
2
enter the number : 1
enter the power : 2
1
enter the number : -1
enter the power : 3
n and p should be non-negative
```

In [1]:

```
#18
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)
for i in range(l):
    obj.pushCharacter(s[i])
    obj.enqueueCharacter(s[i])

isPalindrome=True

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.")
```

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

In [2]:

```
#19
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)
```

```
10
18
```

In [3]:

```
#20
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]) )
```

```
4
2 7 8 3
Array is sorted in 2 swaps.
First Element: 2
Last Element: 8
```

In [4]:

```
#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("Height of tree: ",height)
```

```
3
5
4
7
Height of tree:  1
```

In [5]:

```
#23
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)
```

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

In [7]:

```
#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)
mylist.display(head);
```

```
7
2
2
3
4
5
6
8
2 3 4 5 6 8
```

In [8]:

```
# 25
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))
```

```
2
3
Prime
4
Not prime
```

In [10]:

```
# 26
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)
```

```
10 5 20
5 5 20
75
```

In [11]:

```
#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):

    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
```

In [12]:

```
#28

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 )
```

```
6
riya riya@gmail.com
julia julia@julia.me
julia sjulia@gmail.com
julia julia@gmail.com
samantha samantha@gmail.com
tanya tanya@gmail.com
julia
julia
riya
samantha
tanya
```

In [13]:

```
#29
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 3
1
```

In [ ]: