In [1]:
```
inputString = input()

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

```
naman
Hello, World.
naman
```

In [3]:
```
j = int(input())
e = float(input())
t = input()
i=4
d=4.0
s='Hacker rank'
# Print the sum of both integer variables on a new line.
print(i+j)
# Print the sum of the double variables on a new line.
print(d+e)
# Concatenate and print the String variables on a new line
# The 's' variable above should be printed first.
print(s+t)
```

```
6
7.8
is the best place for coding
10
11.8
Hacker rankis the best place for coding
```

In [6]:
```
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 ."(int((totalcost))))
```

```
<>:8: SyntaxWarning: 'str' object is not callable; perhaps you missed a comma?
<>:8: SyntaxWarning: 'str' object is not callable; perhaps you missed a comma?

700
20
15

<ipython-input-6-61371adacb8a>:8: SyntaxWarning: 'str' object is not callable;
perhaps you missed a comma?
  print ("The total meal cost is ."(int((totalcost))))

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-61371adacb8a> in <module>
      6 totalcost=mealCost+tip+tax;
      7
----> 8 print ("The total meal cost is ."(int((totalcost))))

TypeError: 'str' object is not callable
```

In [7]:
```
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))))
```

```
700
20
15
The total meal cost is 945 dollars.
```

In [8]:
```
n = int(input().strip())

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

elif n>20:
    ans = "Not Weird"

elif n>=6:
    ans = "Weird"

else:
    ans = "Not Weird"

print(ans)
```

```
6
Weird
```

In [15]:
```
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
```

In [16]:
```
N = int(input().strip())
for i in range(1, 11):
    print(str(N) +" x " + str(i) + " = " + str(N*i))
```

```
6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

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

```
2
naman
nmn aa
```

In [3]:
```
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)
```

```
3
7 8 9
9 8 7
```

In [2]:
```
import sys
inputList=[]

for line in sys.stdin:
    inputList.append(line)

n = int(inputList[0])
entries = inputList[1:n+1]
queries = inputList[n+1:]

phoneBook = {}

for entry in entries:
    name, id = entry.split()
    phoneBook[name] = id

for query in queries:
    stripQuery = query.rstrip() #Eliminates the newline character
    if stripQuery in phoneBook:
        print(stripQuery + "=" + str(phoneBook[stripQuery]))
    else:
        print("Not found")
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-2-faa7124a668e> in <module>
      5     inputList.append(line)
      6
----> 7 n = int(inputList[0])
      8 entries = inputList[1:n+1]
      9 queries = inputList[n+1:]

IndexError: list index out of range
```

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

n = int(input())
print(factorial(n))
```

```
5
120
```

In [5]:
```
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)
```

```
7
3
```

In [8]:
```
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]
        if i==0 and j==0:
            max = sum
        if sum > max:
            max =sum

print(max)
```

```
1 1 0 1 0 1
1 1 1 1 1 1
0 0 0 0 0 0
1 2 1 2 1 1
1 3 2 3 1 2
1 3 4 5 2 3
19
```

In [22]:
```
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())
scores=list(map(int,input().split()))
s=Student(firstName,lastName,idNum,scores)
s.printPerson()
print("Grade:",s.calculate())
```

```
Namann Bhan 7091
89
90

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-22-5ac25b0ce5c0> in <module>
     37 scores=list(map(int,input().split()))
     38 s=Student(firstName,lastName,idNum,scores)
---> 39 s.printPerson()
     40 print("Grade:",s.calculate())

AttributeError: 'Student' object has no attribute 'printPerson'
```

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

```
harry potter
jk
400
Title: harry potter
Author: jk
Price: 400
```

In [7]:
```
class Difference:
    def __init__(self, a):
        self.elements=a
        self.maximumDifference = 0

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

d=Difference(a=[1,2,5])
d.computeDifference()
print(d.maximumDifference)
```

```
4
```

In [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);
```

```
5
4
1
3
5
4
4 1 3 5 4
```

In [23]:
```
S = input().strip()
try:
    r = int(S)
    print(r)
except ValueError:
    print("Bad String")
```

```
rv
Bad String
```

In [ ]:
```
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)
```

```
3
2 4
16
```

In [2]:
```
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.")
```

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

In [1]:
```
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)
```

```
9
13
```

In [4]:
```
import sys


n = int(input().strip())
a = list(map(int, input().strip().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]) )
```

```
5
1 6 9 3 4
Array is sorted in 4 swaps.
First Element: 1
Last Element: 9
```

In [5]:
```
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)
```

```
9
7
6
4
6
4
3
2
1
5
Height of tree:  1
```

In [5]:
```
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())
myTree=Solution()
root=None
for i in range(T):
    data=int(input())
    root=myTree.insert(root,data)
myTree.levelOrder(root)
```

```
6
4
6
3
7
5
1
4 3 6 1 5 7
```

In [11]:
```
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
        return head

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
2
2
5
4
4
1
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-7bb6405349ff> in <module>
     38     data=int(input())
     39     head=mylist.insert(head,data)
```

```
---> 40 head=mylist.removeDuplicates(head)
     41 mylist.display(head);

AttributeError: 'Solution' object has no attribute 'removeDuplicates'
```

In [1]:
```
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))
```

```
4
13
Prime
9
Not prime
7
Prime
6
Not prime
```

In [4]:
```
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)
```

```
9 6 2019
4 6 2019
75
```

In [7]:
```
def minimum_index(seq):
    if len(seq) == 0:
        raise ValueError("Cannot get the minimum value index from an empty sequer
    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 [8]:
```
import sys
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
r r@gmail.com
v v@gmail.com
r
v
```

In [*]:
```
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 2
1
```

In [ ]:
```

```