In [ ]:

```
#Day 0; take input and print 'Hello World'

input_string = int(input("Enter a number"))

print('Hello, World.')
```

In [14]:

```
#Day 1; Intro to datatypes
i = 4
d = 4.0
s = 'HackerRank '
second_integer = 5
second_float = 5.0
second_string = "Charles Leclerc "
print(s)
print(i+ second_integer)
print(d + second_float)
print(s + second_string)
```

```
HackerRank
9
9.0
HackerRank Charles Leclerc
```

In [17]:

```
#Day 2;
import math
import os
import random
import re
import sys


def solve(meal_cost, tip_percent, tax_percent):
    # Write your code here
    tip = (tip_percent*meal_cost)/100
    tax = (tax_percent * meal_cost)/100
    total_cost = meal_cost + tip + tax
    round(total_cost)
    return total_cost

meal_cost = float(input("Enter meal cost"))

tip_percent = int(input("Enter tip percent"))

tax_percent = int(input("Enter tax percent"))

solve(meal_cost, tip_percent, tax_percent)
```

```
Enter meal cost20
Enter tip percent8
Enter tax percent12
```

Out[17]:

```
24.0
```

In [ ]:

```
#Day 3;

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


if n%2 == 1:  #odd
```

```
    print("weird")

elif n>20:  #even and more than 20
    print("not weird")

elif n>=6:  #even and more than equal to 6
    print("weird")

else:       #b/w 2 to 5 (less than 6)
    print("not weird")
```

In [15]:

```
#Day 4; Person class

class Person:
    def __init__(self,initialAge):
        if initialAge < 0:
            self.age = 0
        else:
            self.age = initialAge
    def amIOld(self):
        if(self.age < 13):
            print("You are young")
        elif(self.age < 18):
            print("You are a teenager")
        else:
            print("You are old")
    def yearPasses(self):
        self.age += 1


age = int(input("Enter your age "))
p = Person(age)
p.amIOld()
for j in range(0, 3):
    p.yearPasses()
p.amIOld()
print("")
```

```
Enter your age 16
You are a teenager
You are old
```

In [20]:

```
#Day 5; Multiples

n = int(input("Enter a number for its multiples"))
for i in range(1,11):
    print(f"{n} * {i} = {n*i}" )
```

```
Enter a number for its multiples20
20 * 1 = 20
20 * 2 = 40
20 * 3 = 60
20 * 4 = 80
20 * 5 = 100
20 * 6 = 120
20 * 7 = 140
20 * 8 = 160
20 * 9 = 180
20 * 10 = 200
```

In [23]:

```
#Day 6; String positions

inputString = input("Enter a string ")
evenString = ""
```

```
oddString = ""
for i in range(0, len(inputString)):
    if i % 2 == 0:
        evenString = evenString + inputString[i]
    else:
        oddString = oddString + inputString[i]

print(evenString)
print(oddString)
```

```
Enter a string abcdef
ace
bdf
```

In [27]:

```
#Day 7; Working with arrays

arr = []
lim = int(input("Enter the number of integers to be put in the array: "))
for i in range(0,lim):
    num = int(input(f"Integer {i+1}: "))
    arr.append(num)
for i in reversed(range(0,lim)):
    print(arr[i])
```

```
Enter the number of integers to be put in the array: 4
Integer 1: 10
Integer 2: 12
Integer 3: 14
Integer 4: 11
11
14
12
10
```

In [7]:

```
#Day 9; Factorial
def factorial(n):
    if n == 0:
        return 1

    return n * factorial(n-1)

n = int(input("Enter a number to calculate the factorial: "))
factorial(n)
```

```
Enter a number to calculate the factorial: 5
```

Out[7]:

```
120
```

In [16]:

```
#Day 10; Binary conversion
num = int(input("Enter a number in decimal system: "))
n = num
count = 0
while n >= 1:
    rem = n % 2
    n = n//2
    if rem == 1:
        count += 1
    else:
        count = 0
print("\n")
if count == 1:
    print("0 ones are together")
else:
    print(f"{count} ones are together")
```

```
Enter a number in decimal system: 63


6 ones are together
```

In [28]:

```python
#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,firstName,lastName,idNumber,scores):
        super().__init__(firstName,lastName,idNumber)
        self.studentSC = scores

    def calculate(self):
        sum = 0
        for mark in self.studentSC:
            sum += mark
        avg = sum/len(self.studentSC)
        if avg >= 90:
            grade = 'O'
        elif avg >= 80:
            grade = 'E'
        elif avg >= 70:
            grade = 'A'
        elif avg >= 55:
            grade = 'P'
        elif avg >= 40:
            grade = 'D'
        else:
            grade = 'T'

        return grade



    #   Return: A character denoting the grade.
    #
    # Write your function here
fname = input("Enter your first name: ")
lname = input("Enter your last name: ")
studentId = input("Enter your student ID")
print("Enter your scores for PCM out of 100: ")
scores = []
for i in range(0,3):
    subject = int(input(f"Subject {i + 1}: "))
    scores.append(subject)

p = Person(fname,lname,studentId)


s = Student(fname, lname, studentId, scores)
s.printPerson()
print("Grade:", s.calculate())
```

```
Enter your first name: Aum
Enter your last name: Rajadnye
Enter your student ID7009
Enter your scores for PCM out of 100:
Subject 1: 90
Subject 2: 87
Subject 3: 78
```

```
Name: Rajadnye, Aum
ID: 7009
Grade: E
```

In [29]:

```
#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

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()
```

```
Not a Penny more, not a Penny less
Jeffery Archer
400
Title: Not a Penny more, not a Penny less
Author: Jeffery Archer
Price: 400
```

In [30]:

```
#Day 14;
class Difference:
    def __init__(self, a):
        self.__elements = a
    def computeDifference(self):
        self.maximumDifference = max(self.__elements) - min(self.__elements)
        return None

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

```
4
```

In [31]:

```
#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):
        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
3
5
2
1
8
35218
```

In [33]:

```
#Day 16;
import sys

s = input().strip()
try:
    r = int(s)
    print(r)
except ValueError:
    print("Bad String")
```

```
jss
Bad String
```

In [ ]:

```
#Day 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,p = map(int, input().split())
    try:
        ans = myCalculator.power(n,p)
        print(ans)
    except Exception as e:
        print(e)
```

In [46]:

```
#Day 18;
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.")
```

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

In [49]:

```
#Day 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)
```

```
9
13
```

In [53]:

```
#Day 20;
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]))
```

```
5
```

```
9 1 7 3 10
Array is sorted in 4 swaps.
First Element: 1
Last Element: 10
```

In [ ]:

```
#Day 22;
def getHeight(self,root):
    if not root:
        return -1
    else:
        a = self.getHeight(root.left)
        b = self.getHeight(root.right)
        if (a > b):
            return (a + 1)
        else:
            return (b + 1)
    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
```

In [5]:

```
#Day 23;
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
7
3
1
2
9
4 3 7 1 9 2
```

In [7]:

```
#Day 24;
```

```
4
10
Not prime
17
Prime
27
Not prime
10
Not prime
```

In [8]:

```
#Day 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))
```

```
4
10
Not prime
17
Prime
23
Prime
96
Not prime
```

In [5]:

```
#Day 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)
```

```
6 7 2020
4 12 2020
0
```

In [9]:

```
#Day 28;
import re
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)
```

```
2
a a@gmail.com
r r@gmail.com
a
r
```

In [6]:

```
#Day 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 7
5
8 2
1
2 2
0
```