Assignment 1: 30 Days Of Code Challenge

Name: Ashish Nanda

Roll No.: J041

Day 0: Hello, World

Task: To complete this challenge, you must save a line of input from stdin to a variable, print Hello, World. on a single line, and finally print the value of your variable on a second line.

In [2]:

```
input_string = input()

print('Hello, World.')

print(input_string)
```

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

Day 1: Data Types

Task: Complete the code in the editor below. The variables i, d, and s are already declared and initialized for you. You must:

- Declare 3 variables: one of type int, one of type double, and one of type String.
- Read 3 lines of input from stdin (according to the sequence given in the Input Format section below) and initialize your variables.
- Use the + operator to perform the following operations: Print the sum of plus your int variable on a new line. Print the sum of plus your double variable to a scale of one decimal place on a new line. Concatenate with the string you read as input and print the result on a new line.

In [4]:

```
i = 4
d = 4.0
s = 'HackerRank '

i_2 = int(input())
d_2 = float(input())
s_2 = str(input())

print(i + i_2)

print(d + d_2)

print(s + s_2)
```

```
16
8.0
HackerRank is the best place to learn and practice coding!
```

Day 2: Operators

Task: Given the meal price (base cost of a meal), tip percent (the percentage of the meal price being added as tip), and tax percent (the percentage of the meal price being added as tax) for a meal, find and print the meal's total cost. Round the result to the nearest integer.

In [10]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'solve' function below.
#
# The function accepts following parameters:
#  1. DOUBLE meal_cost
#  2. INTEGER tip_percent
#  3. INTEGER tax_percent
#

def solve(meal_cost, tip_percent, tax_percent):
    tip = (tip_percent / 100.0) * meal_cost

    tax = (tax_percent / 100.0) * meal_cost

    total_cost = meal_cost + tip + tax

    total_rounded = round(total_cost)

    print(total_rounded)

if __name__ == '__main__':
    meal_cost = float(input().strip())

    tip_percent = int(input().strip())

    tax_percent = int(input().strip())

    solve(meal_cost, tip_percent, tax_percent)
```

15

Day 3: Intro to Conditional Statements

Task: Given an integer, n, perform the following conditional actions:

- If n is odd, print Weird
- If n is even and in the inclusive range of 2 to 5, print Not Weird
- If n is even and in the inclusive range of 6 to 20, print Weird
- If n is even and greater than 20, print Not Weird

Complete the stub code provided in your editor to print whether or not is weird.

In [11]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

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

    if N % 2 != 0:
        print('Weird')

    else:
        if 2 <= N <= 5:
            print('Not Weird')

        elif 6 <= N <= 20:
            print('Weird')

        else:
            print('Not Weird')
```

```
Weird
```

Day 4: Class vs. Instance

Task: Write a Person class with an instance variable, age, and a constructor that takes an integer, initialAge, as a parameter. The constructor must assign initialAge to age after confirming the argument passed as initialAge is not negative; if a negative argument is passed as initialAge, the constructor should set age to 0 and print Age is not valid, setting age to 0.

In addition, you must write the following instance methods:

1. yearPasses() should increase the instance variable by .
2. amIOld() should perform the following conditional actions:

- If age < 13, print You are young..
- If age >= 13 and age < 18, print You are a teenager..
- Otherwise, print You are old.

In [2]:

```
class Person:

    def __init__(self, initialAge):
        if initialAge < 0:
            self.age = 0
            print('Age is not valid, setting age to 0.')

        else:
            self.age = initialAge

    def amIOld(self):
        if self.age < 13:
            print('You are young.')

        elif self.age >= 13 and self.age < 18:
            print('You are a teenager.')

        else:
            print('You are old.')

    def yearPasses(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.yearPasses()
    p.amIOld()
    print("")
```

```
Age is not valid, setting age to 0.
You are young.
You are young.

You are young.
You are a teenager.

You are a teenager.
You are old.

You are old.
You are old.
```

Day 5: Loops

Task: Given an integer, n, print its first 10 multiples. Each multiple n x i (where 1 <= i <= 10) should be printed on a new line in the form: n x i = result.

In [1]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

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

    for i in range(1, 11):
        print(f'{n} x {i} = {n * i}')
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Day 6: Let's Review

Task: Given a string, S, of length N that is indexed from 0 to N - 1, print its even-indexed and odd-indexed characters as 2 space-separated strings on a single line (see the Sample below for more detail).

Note: 0 is considered to be an even index.

In [16]:

```
if __name__ == '__main__':

    T = int(input())

    for i in range(T):
        S = input()

        even_words = ''
        odd_words = ''

        for j in range(len(S)):
            if j % 2 == 0:
                even_words += S[j]

            if j % 2 != 0:
                odd_words += S[j]

        print(f'{even_words} {odd_words}')
```

```
Hce akr
Rn ak
```

Day 7: Arrays

Task: Given an array, A, of N integers, print A's elements in reverse order as a single line of space-separated numbers.

In [4]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

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

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

    for i in range(n - 1, -1, -1):
        print(arr[i], end=' ')
```

```
2 3 4 1
```

Day 8: Dictionaries and Maps

Task: Given n names and phone numbers, assemble a phone book that maps friends' names to their respective phone numbers. You will then be given an unknown number of names to query your phone book for. For each name queried, print the associated entry from your phone book on a new line in the form name=phoneNumber; if an entry for name is not found, print Not found instead.

In [1]:

```
import sys

n = int(input())

usr_dict = dict()

for i in range(n):
    usr_input = input()
    split_input = usr_input.split()
    usr_dict[split_input[0]] = split_input[1]

input_lines = sys.stdin.readlines()

for i in input_lines:
    search_name = i.rstrip('\n')

    if search_name in usr_dict:
        print(f'{search_name}={usr_dict[search_name]}')

    else:
        print('Not found')
```

Day 9: Recursion 3

Task: Complete the factorial function in the editor below. Be sure to use recursion.

In [6]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'factorial' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER n as parameter.
#

def factorial(n):
    if n == 1:
        return 1

    else:
        result = n * factorial(n - 1)

    return result

n = int(input())

output = factorial(n)
print(output)
```

```
6
```

Day 10: Binary Numbers

Task: Given a base-10 integer, n, convert it to binary (base-2). Then find and print the base-10 integer denoting the maximum number of consecutive 1's in n's binary representation. When working with different bases, it is common to show the base as a subscript.

In [41]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

if __name__ == '__main__':
    n = int(input().strip())
    binary = []
    ones_list = []
    count = 0

    while n >= 1:
        rem = int(n % 2)
        binary.append(rem)
        n //= 2

    binary.reverse()

    for i in range(len(binary)):
        if binary[i] == 1:
            count += 1
        else:
            count = 0

        ones_list.append(count)

    print(max(ones_list))
```

2

Day 11: 2D Arrays

Task: Calculate the hourglass sum for every hourglass in A, then print the maximum hourglass sum.

In [5]:

```
import numpy as np

A = [[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]]

max_sum = -63

for i in range(4):
    for j in range(4):
        upper = A[i][j] + A[i][j + 1] + A[i][j + 2]
        middle = A[i + 1][j + 1]
        lower = A[i + 2][j] + A[i + 2][j + 1] + A[i + 2][j + 2]

        sub_sum = np.sum([upper, middle, lower])

        if sub_sum > max_sum:
            max_sum = sub_sum

print(max_sum)
```

```
19
```

Day 12: Inheritance

Task: You are given two classes, Person and Student, where Person is the base class and Student is the derived class. Completed code for Person and a declaration for Student are provided for you in the editor. Observe that Student inherits all the properties of Person.

In [6]:

```
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):
    #   Class Constructor
    #
    #   Parameters:
    #   firstName - A string denoting the Person's first name.
    #   lastName - A string denoting the Person's last name.
    #   id - An integer denoting the Person's ID number.
    #   scores - An array of integers denoting the Person's test scores.
    #
    # Write your constructor here
    def __init__(self, firstName, lastName, idNumber, scores):
        self.firstName = firstName
        self.lastName = lastName
        self.idNumber = idNumber
        self.scores = scores


    #   Function Name: calculate
    #   Return: A character denoting the grade.
    #
    # Write your function here
    def calculate(self):

        average_score = sum(self.scores) / len(self.scores)

        if 90 <= average_score <= 100:
            return 'O'

        elif 80 <= average_score < 90:
            return 'E'

        elif 70 <= average_score < 80:
            return 'A'

        elif 55 <= average_score < 70:
            return 'P'

        elif 40 <= average_score < 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())
```

```
Name: Memelli, Heraldo
ID: 8135627
Grade: O
```

Day 13: Abstract Classes

Task: Given a Book class and a Solution class, write a MyBook class

In [7]:

```
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):
        self.title = title
        self.author = 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()
```

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

Day 14: Scope

Task: Complete the Difference class by writing the following:

- A class constructor that takes an array of integers as a parameter and saves it to the __elements instance variable.
- A computeDifference method that finds the maximum absolute difference between any 2 numbers in __elements and stores it in the maximumDifference instance variable.

In [10]:

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

    # Add your code here
    def computeDifference(self):
        self.maximumDifference = abs(max(self.__elements) - min(self.__elements
))

# End of Difference class

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

d = Difference(a)
d.computeDifference()

print(d.maximumDifference)
```

```
4
```

Day 15: Linked List

Task: Complete the insert function in your editor so that it creates a new Node (pass data as the Node constructor argument) and inserts it at the tail of the linked list referenced by the head parameter. Once the new node is added, return the reference to the head node.

In [1]:

```
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
        new_Node = Node(data)

        if head == None:
            head = new_Node
            self.tail = head

        else:
            self.tail.next = new_Node
            self.tail = new_Node

        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);
```

```
2 3 4 1
```

Day 16: Exceptions - String to Integer

Task: Read a string, S, and print its integer value; if S cannot be converted to an integer, print Bad String.

In [2]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

S = input()

try:
    S_int = int(S)
    print(S_int)

except ValueError:
    print('Bad String')
```

```
3
```

Day 17: More Exceptions

Task: Write a Calculator class with a single method: int power(int,int). The power method takes two integers, n and p, as parameters and returns the integer result of n^p. If either n or p is negative, then the method must throw an exception with the message: n and p should be non-negative.

In [3]:

```
#Write your code here
class Calculator:
    def power(self, n, p):
        if n < 0 or p < 0:
            raise Exception('n and p should be non-negative')
        else:
            result = int(n ** p)
            return result

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)
```

```
243
16
n and p should be non-negative
n and p should be non-negative
```

Day 18: Queues and Stacks

Task: Determine if a given string, s, is a palindrome

In [4]:

```
import sys

class Solution:
    # Write your code here
    def __init__(self):
        self.stack = list()
        self.queue = list()

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

    def enqueueCharacter(self, ch):
        self.queue.append(ch)

    def popCharacter(self):
        last = self.stack[-1]
        self.stack = self.stack[:-1]
        return last

    def dequeueCharacter(self):
        first = self.queue[0]
        self.queue = self.queue[1:]
        return first

# 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.")
```

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

Day 19: Interfaces

Task: The AdvancedArithmetic interface and the method declaration for the abstract divisorSum(n) method are provided for you in the editor below. Complete the implementation of Calculator class, which implements the AdvancedArithmetic interface. The implementation for the divisorSum(n) method must return the sum of all divisors of n.

In [5]:

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

class Calculator(AdvancedArithmetic):
    def divisorSum(self, n):
        sum_div = 0

        for i in range(1, n + 1):
            if n % i == 0:
                sum_div += i

        return sum_div

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

```
I implemented: AdvancedArithmetic
12
```

Day 20: Sorting

Task: Given an array, a, of size n distinct elements, sort the array in ascending order using the Bubble Sort algorithm above.

In [6]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

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

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

    # Write your code here
    numberOfSwaps = 0

    for i in range(0, n):
        for j in range(0, n - 1):
            if a[j] > a[j + 1]:
                # Swapping
                temp = a[j]
                a[j] = a[j + 1]
                a[j + 1] = temp

                numberOfSwaps += 1

        if numberOfSwaps == 0:
            break

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

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

Day 21: Sorting

Cannot be implemented in Python

Day 22: Binary Search Trees

Task: The height of a binary search tree is the number of edges between the tree's root and its furthest leaf. You are given a pointer, root, pointing to the root of a binary search tree. Complete the getHeight function provided in your editor so that it returns the height of the binary search tree.

In [7]:

```
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 == None or (root.left == None and root.right == None):
            return 0

        else:
            if self.getHeight(root.left) > self.getHeight(root.right):
                return self.getHeight(root.left) + 1

            else:
                return self.getHeight(root.right) + 1


T=int(input())
```

Day 23: BST Level-Order Traversal

Task: A level-order traversal, also known as a breadth-first search, visits each level of a tree's nodes from left to right, top to bottom. You are given a pointer, root, pointing to the root of a binary search tree. Complete the levelOrder function provided in your editor so that it prints the level-order traversal of the binary search tree.

In [10]:

```
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):
        queue = [root]

        while len(queue) != 0:
            element = queue[0]
            queue = queue[1:]

            print(f'{element.data} ', end='')

            if element.left != None:
                queue.append(element.left)

            if element.right != None:
                queue.append(element.right)

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

```
3 2 5 1 4 7
```

Day 24: More Linked Lists

Task: A Node class is provided for you in the editor. A Node object has an integer data field, data, and a Node instance pointer, next, pointing to another node (i.e.: the next node in a list). A removeDuplicates function is declared in your editor, which takes a pointer to the head node of a linked list as a parameter. Complete removeDuplicates so that it deletes any duplicate nodes from the list and returns the head of the updated list.

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

    def removeDuplicates(self,head):
        curr_node = head

        while curr_node:
            if (curr_node.next) and (curr_node.data == curr_node.next.data):
                curr_node.next = curr_node.next.next
                continue

            curr_node = curr_node.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);
```

```
1 2 3 4
```

Day 25: Running Time and Complexity

Task: A prime is a natural number greater than 1 that has no positive divisors other than 1 and itself. Given a number, n, determine and print whether it is Prime or Not prime.

In [9]:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT

def is_prime(n):
    if n == 1:
        return 'Not prime'

    else:
        for i in range(2, int(n / 2) + 1):
            if n % i == 0:
                return 'Not prime'

    return 'Prime'

T = int(input())

for i in range(T):
    n = int(input())

    print(is_prime(n))
```

```
Not prime
Prime
Prime
```

Day 26: Nested Logic

Task: Your local library needs your help! Given the expected and actual return dates for a library book, create a program that calculates the fine (if any).

In [19]:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
actual = input().split(' ')
actual = [int(i) for i in actual]

expected = input().split(' ')
expected = [int(i) for i in expected]

fine = 0

if actual[2] == expected[2]:
    if (actual[0] > expected[0]) and (actual[1] == expected[1]):
        fine = 15 * (actual[0] - expected[0])

    elif actual[1] > expected[1]:
        fine = 500 * (actual[1] - expected[1])

elif actual[2] > expected[2]:
    fine = 10000

else:
    fine = 0

print(fine)
```

```
45
```

Day 27: Testing

Task: This problem is about unit testing.

Your company needs a function that meets the following requirements:

- For a given array of integers, the function returns the index of the element with the minimum value in the array. If there is more than one element with the minimum value, it returns the smallest one.
- If an empty array is passed to the function, it raises an exception. A colleague has written this method. The implementation in Python is listed below. Implementations in other languages can be found in the code template.

In [20]:

```
def minimum_index(seq):
    if len(seq) == 0:
        raise ValueError("Cannot get the minimum value index from an empty seque
nce")
    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 [3, 7, 2, 1, 11]

    @staticmethod
    def get_expected_result():
        return 3

class TestDataExactlyTwoDifferentMinimums(object):
    @staticmethod
    def get_array():
        return [1, 7, 2, 1, 11]

    @staticmethod
    def get_expected_result():
        return 0

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
```

Day 28: RegEx, Patterns, and Intro to Databases

Task: Consider a database table, Emails, which has the attributes First Name and Email ID. Given N rows of data simulating the Emails table, print an alphabetically-ordered list of people whose email address ends in @gmail.com.

In [8]:

```
#!/bin/python3

import math
import os
import random
import re
import sys


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

    names = []

    for N_itr in range(N):
        first_multiple_input = input().rstrip().split()

        firstName = first_multiple_input[0]

        emailID = first_multiple_input[1]

        if '@gmail.com' in emailID:
            names.append(firstName)

    names.sort(reverse=False)

    for i in range(len(names)):
        print(names[i])
```

```
julia
julia
riya
samantha
tanya
```

Day 29: Bitwise AND

Task: Given set S = {1, 2, 3, ..., N}. Find two integers, A and B (where A < B), from set S such that the value of A&B is the maximum possible and also less than a given integer, K. In this case, & represents the bitwise AND operator.

In [24]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'bitwiseAnd' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER N
#  2. INTEGER K
#

def bitwiseAnd(N, K):
    max_val = 0

    for i in range(N):
        for j in range(i + 1, N):
            comb = i & j
            if K > comb and comb > max_val:
                max_val = comb

    return max_val

if __name__ == '__main__':

    t = int(input().strip())

    for t_itr in range(t):
        first_multiple_input = input().rstrip().split()

        count = int(first_multiple_input[0])

        lim = int(first_multiple_input[1])

        res = bitwiseAnd(count, lim)

        print(res)
```

```
1
4
0
```