# Hackerrank 30 days of Code

## Day 0: Hello, World!

**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 value of your variable on a second line**

In [1]:

```
input_string = input()
print(f'Hello, World!\n{input_string}')
```

```
hi there
Hello, World!
hi there
```

## Day 1: Data Types

**Complete the code in the editor below. The variables l, d, and s are already declared and initialized for you. You must: Declare variables: one of type int, one of type double, and one of type String. Read 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 l plus your int variable on a new line. Print the sum of d plus your double variable to a scale of one decimal place on a new line. Concatenate d with the string you read as input and print the result on a new line.**

In [2]:

```
l, d, s = 1, 1.5, 'Hey'
a = int(input('int :' ))
b = float(input(' double: '))
c = input("String: ")
print( f'{l + a}')
print(f'{round(d + b, 1)}')
print( f'{s + c}')
```

```
int :4
 double: 2.6
String: string
5
4.1
Heystring
```

## Day 2: Operators

**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 [3]:

```
print(round(int(input('Cost: ')) * (int(input('Tip: ')) + int(input('Tax: ')) + 100) / 1
00))
```

```
Cost: 100
Tip: 10
Tax: 2
112
```

## Day 3: Intro to Conditional Statements

**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 [4]:

```
n = int(input())
if n % 2 == 1 or n in range(6, 21, 2):
 print('Weird')
else:
 print('Not Weird')
```

```
3
Weird
```

## Day 4: Class vs. Instance

**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: yearPasses() should increase the instance variable by 1. amIOld() should perform the following conditional actions: If , print You are young.. If and , print You are a teenager.. Otherwise, print You are old.. To help you learn by example and complete this challenge, much of the code is provided for you, but you'll be writing everything in the future. The code that creates each instance of your Person class is in the main method. Don't worry if you don't understand it all quite yet!**

In [5]:

```
class Person:

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

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

    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")
```

In [6]:

```
for t in range(3):
                initial_age = int(input('Enter Age: '))
                person = Person(initial_age=initial_age)
                person.amIOld()
                person.yearPasses()
                person.amIOld()
```

```
Enter Age: 17
You are a teenager
You are old
Enter Age: 14
You are a teenager
You are a teenager
Enter Age: 12
You are young
You are a teenager
```

## Day 7:

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

In [7]:

```
A = [34, 27, 31, 76, 11]
for i in A[::-1]:
 print(i, end= " ")
print()
```

```
11 76 31 27 34
```

## Day 8:

**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 [18]:

```
phonebook = {}
for i in range(int(input('no of contacts:'))):
    inp = input().split()
    phonebook[inp[0]] = inp[1]
q = input('Query: ')
while q != "":
    try:
        print(f'{q}={phonebook[q]}')
    except:
        print('Not Found!')
    q = input('Query: ')
```

```
no of contacts:1
falguni 1280
Query: shruti
Not Found!

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-18-38f0011ffbd5> in <module>
      9     except:
     10         print('Not Found!')
---> 11     q = input('Query: ')

~\Anaconda\lib\site-packages\ipykernel\kernelbase.py in raw_input(self, prompt)
    858                 "raw_input was called, but this frontend does not support input r
equests."
    859             )
--> 860         return self._input_request(str(prompt),
    861             self._parent_ident,
    862             self._parent_header,

~\Anaconda\lib\site-packages\ipykernel\kernelbase.py in _input_request(self, prompt, iden
t, parent, password)
    902             except KeyboardInterrupt:
    903                 # re-raise KeyboardInterrupt, to truncate traceback
--> 904                 raise KeyboardInterrupt("Interrupted by user") from None
    905             except Exception as e:
    906                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

## Day 9:

**Recursive Method for Calculating Factorial**

In [19]:

```
def recursive_factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * recursive_factorial(n - 1)

print( recursive_factorial(int(input('enter number:'))))
```

```
enter number:5
120
```

## Day 10:

**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 [20]:

```
n = int(input('>>>'))
binary = ''
while n > 0:
    binary = str(n % 2) + binary
    n //= 2

print(max([len(i) for i in binary.split('0')]))
```

```
>>>6
2
```

## Day 11:

**2D array**

In [ ]:

```
import math
import os
import random
import re
import sys

if __name__ == '__main__':
    arr = []
    max = -99
    for _ in range(6):
        arr.append(list(map(int, input().rstrip().split())))

    for i in range(6):
        for j in range(6):
            if ((i + 2 < 6) and (j + 2 < 6)):
                temp = arr[i][j] + arr[i + 2][j] + arr[i][j + 1] + arr[i + 1][j + 1] +
arr[i + 2][j + 1] + arr[i][
                    j + 2] + arr[i + 2][j + 2]
                if (temp > max):
                    max = temp

    print(max)
```

## Day 12: Inheritence

**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 [27]:

```
class Student:
    def __init__(self, first_name, last_name, id_number, scores):
        self.first_name = first_name
        self.last_name = last_name
        self.id_number = id_number
        self.scores = scores

    def calculate(self):
        d = {
            'T': 40,
            'D': 55,
            'P': 70,
            'A': 80,
            'E': 90,
            'O': 100,

        }

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

        for grade, score_threshold in d.items():
            if avg < score_threshold:
                 return grade
stu = Student('Falguni', 'Gaikwad', 7001245, [99, 99, 99, 99])
print(f'Name: {stu.first_name} {stu.last_name}')
print(f'ID: {stu.id_number}')
print(f'Grade: {stu.calculate()}')
```

```
Name: Falguni Gaikwad
ID: 7001245
Grade: O
```

## Day 13: Abstract Classes

In [30]:

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

    def display(self):
        print(f'Title: {self.title}\nAuthor: {self.author}\nPrice: {self.price}')

my_book1 = MyBook('Coma', 'Robin Cook', 250)
my_book1.display()
```

```
Title: Coma
Author: Robin Cook
Price: 250
```

## Day 14

In [38]:

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

    def maximum_difference(self):
        return max(self.elements) - min(self.elements)

diff = Difference([20, 35, 10, 34, 15, 45])
diff.maximum_difference()
```

Out[38]:

```
35
```

## Day 15: Linked List

In [39]:

```
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
7
2
8
9
4
7 2 8 9 4
```

## Day 16

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

In [42]:

```
S = input()
try:
    print(int(S))
except:
    print("Bad String")
```

```
hi
Bad String
```

## Day 17

In [54]:

```
class calculator:
    def power(self, n, p):
        if n < 0 or p < 0:
            print("n and p should be non-negative")
        else:
            print(n ** p)
```

## Day 18

**Queue-Stack**

In [61]:

```
class Solution:
    def __init__(self):
        self.stack = []
        self.queue = []

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

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

    def pop_char(self):
        try:
            x = self.stack[-1]
            self.stack = self.stack[:-1]
            return x
        except:
            return None

    def dequeue(self):
        try:
            x = self.queue[0]
            self.queue = self.queue[1:]
            return x
        except:
            return None

sol = Solution()
S = 'Falguni'
for c in S:
    sol.push_char(c)
    sol.enqueue_char(c)
flag = True
while True:
    from_stack = sol.pop_char()
    from_queue = sol.dequeue()

    if from_stack == None:
        break

    if from_stack != from_queue:
        flag = False
        break


if flag:
    print("Palindrome")
else:
    print("Not Palindrome")
```

```
Not Palindrome
```

## Day 19

In [64]:

```
class Calculator(AdvancedArithmetic):
    def divisorSum(self, n):
        val = 0
        for i in range(1,n+1):
            if(n % i == 0):
                val += i
        return val

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

```
4
I implemented: AdvancedArithmetic
7
```

## Day 20

**Sorting Problem**

In [8]:

```
if __name__ == '__main__':
    n = 7
    a = [1, 6, 3, 5, 8, 2, 4]

    num_swaps = 0
    for i in range(n):
        flag = True
        for j in range(n-1):
            if a[j] > a[j+1]:
                a[j], a[j+1] = a[j+1], a[j]
                flag = False
                num_swaps += 1
        # print(a)

        if flag:
             break

    print(f'Array is sorted in {num_swaps} swaps\nFirst Element {a[0]}\nLast Element {a[-
1]}')
```

```
Array is sorted in 9 swaps
First Element 1
Last Element 8
```

## Day 21

**BST height**

In [1]:

```
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:
            return 1 + max(self.getHeight(root.left), self.getHeight(root.right))

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)
```

```
3
23
21
26
1
```

## Day 23

In [6]:

```
def levelOrder(self, root):
    ret = ""
    queue = [root]
    while queue:
        current = queue.pop(0)
        ret += str(current.data) + " "
        if current.left:
            queue.append(current.left)
        if current.right:
            queue.append(current.right)
    print(ret[:-1])
```

## Day 24

In [8]:

```
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 is not None and current.next is not None:
            currData = current.data
            currNext = current.next
            if currData is currNext.data:
                current.next = current.next.next
            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);
```

```
4
1
6
3
1
1 6 3 1
```

## Day 25

In [3]:

```
from math import sqrt
from sys import stdin


def checkPrime(n):
    for i in range(2, int(sqrt(n))+1):
        if n % i == 0:
            return False
    return True


n = int(input())
for line in stdin:
    val = int(line)
    if (val >= 2 and checkPrime(val)):
        print("Prime")
    else:
        print("Not prime")
```

```
7
```

## Day 26

In [ ]:

```
actual = input()
actual = list(map(int, actual.split(" ")))

expected = input()
expected = list(map(int, expected.split(" ")))

actualDate = actual[0]
actualMonth = actual[1]
actualYear = actual[2]
```

```
expectedDate = expected[0]
expectedMonth = expected[1]
expectedYear = expected[2]

fine = 0

if actualYear > expectedYear:
    fine = 10000
elif actualYear == expectedYear: #Same calender year
    if actualMonth > expectedMonth: #checking months
        fine = 500 * (actualMonth - expectedMonth)
    elif actualMonth == expectedMonth: #check dates now
        if actualDate > expectedDate:
            fine = 15 * (actualDate - expectedDate)
print(fine)
```

## Day 27

In [ ]:

```
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 [1, 3, 2, 5]

    @staticmethod
    def get_expected_result():
        return 0
class TestDataExactlyTwoDifferentMinimums(object):
    @staticmethod
    def get_array():
        return [1, 3, 2, 1, 5]

    @staticmethod
    def get_expected_result():
        return 0
```

## Day 28

In [ ]:

```
if __name__ == '__main__':
    N = int(input().strip())
    names = []

    for N_itr in range(N):
        i = input().split()
        if '@gmail.com' in i[1]:
            names.append(i[0])

    for name in sorted(names):
        print(name)
```

# Day 29

In [ ]:

```
import math
import os
import random
import re
import sys



if __name__ == '__main__':
    T = int(input())

    for i in range(T):
        tmp = str(input()).split()
        N = int(tmp[0])
        K = int(tmp[1])

        maximum = 0

        for j in range(1, N):
            for k in range(j + 1, N + 1):
                h = j & k
                if K > h > maximum:
                    maximum = h

        print(maximum)
```

In [ ]:

```

```