# Day 0: Hello, world

In [ ]:

```
# Read a full line of input from stdin and save it to our dynamically typed variable, inp
ut_string.
input_string = input()

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

# TODO: Write a line of code here that prints the contents of input_string to stdout.
print(input_string)
```

# Day1: Data Types

In [ ]:

```
i = 4
d = 4.0
s = 'HackerRank '
# Declare second integer, double, and String variables.

# Read and save an integer, double, and String to your variables.
i2 = int(input())
d2 = float(input())
s2 = input()

# Print the sum of both integer variables on a new line.
print(i + i2)
# Print the sum of the double variables on a new line.
print(f"{d + d2}")
# Concatenate and print the String variables on a new line
# The 's' variable above should be printed first.
print(s+s2)
```

# Day 2: Operators

In [ ]:

```
#!/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):
    # Write your code here
    tip_cost = (tip_percent / 100) * meal_cost
    tax_cost = (tax_percent / 100) * meal_cost
    total_meal_cost = int(round(meal_cost + tip_cost + tax_cost, 0))
    print(total_meal_cost)
```

```
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)
```

# Day 3: Intro to Conditional Statements

In [ ]:

```
#!/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 N <= 5:
            print("Not Weird")
        elif N <= 20:
            print("Weird")
        else:
            print("Not Weird")
```

# Day 4: Class vs. Instance

In [ ]:

```
class Person:
    def __init__(self,initialAge):
        # Add some more code to run some checks on initialAge
        if initialAge > 0:
            self.age = initialAge
        else:
            self.age = 0
            print("Age is not valid, setting age to 0.")
    def amIOld(self):
        # Do some computations in here and print out the correct statement to the console
        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):
        # Increment the age of the person in here
        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("")
```

# Day 5: Loops

In [ ]:

```
#!/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, 1):
        print(f"{n} x {i} = {n * i}")
```

# Day 6: Let's Review

In [ ]:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
T = int(input())

for i in range(T):

    S = input()

    even_S = ""
    odd_S = ""

    for idx, ch in enumerate(S):
        if idx % 2 == 0:
            even_S += ch
        else:
            odd_S += ch

    print(even_S + " " + odd_S)
```

# Day 7: Arrays

In [ ]:

```
#!/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()))

    arr = arr[::-1]

    for a in arr:
        print(a, end=" ")
```

## Day 8: Dictionaries and Maps

In [ ]:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
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:
    friend, number = entry.split()
    phonebook[friend] = number

for query in queries:
    query = query.rstrip()
    existing_num = phonebook.get(query)

    if existing_num == None:
        print("Not found")
    else:
        print(query + "=" + str(existing_num))
```

## Day 9: Recursion 3

In [ ]:

```
#!/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):
    # Write your code here
    if n <= 1:
        return 1
    else:
        return n * factorial(n - 1)

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

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

    result = factorial(n)

    fptr.write(str(result) + '\n')

    fptr.close()
```

# Day 10: Binary Numbers

In [ ]:

```
#!/bin/python3

import math
import os
import random
import re
import sys

def get_binary(n):
    if n == 0:
        return 0
    else:
        return n % 2 + 10 * (get_binary(n // 2))

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

    count = 0
    count_1 = 0
    count_1_list = []
    while True:
        if str(binary_num)[count] == "1":
            count_1 += 1
        else:
            count_1_list.append(count_1)
            count_1 = 0
        count += 1

        if len(str(binary_num)) == count:
            count_1_list.append(count_1)
            break

    print(max(count_1_list))
```

# Day 11: 2D Arrays

In [ ]:

```
#!/bin/python3

import math
import os
import random
import re
import sys



if __name__ == '__main__':

    arr = []

    for _ in range(6):
        arr.append(list(map(int, input().rstrip().split())))

    base_idx = [[0, 0], [0, 1], [0, 2], [1, 1], [2, 0], [2, 1], [2, 2]]

    sum_ = 0
    sum_list = []
    inc_col = 0
    inc_row = 0

    while True:
        for idx in base_idx:
```

```
            sum_ += arr[idx[0] + inc_row][idx[1] + inc_col]
        sum_list.append(sum_)
        sum_ = 0
        inc_col += 1
        if inc_col == 4:
            inc_col = 0
            inc_row += 1

        if inc_row == 4:
            break

    print(max(sum_list))
```

# Day 12: Inheritance

In [ ]:

```
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


    #   Function Name: calculate
    #   Return: A character denoting the grade.
    #
    # Write your function here
    def __init__(self, firstName, lastName, idNumber, scores):
        self.scores = scores
        Person.__init__(self, firstName, lastName, idNumber)

    def calculate(self):
        sum_ = 0
        for ele in self.scores:
            sum_ += ele

        avg = sum_ // len(self.scores)
        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


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())
```

# Day 13: Abstract Classes

In [ ]:

```
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.price = price
        Book.__init__(self, title, author)

    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()
```

# Day 14: Scope

In [ ]:

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

    # Add your code here
    def computeDifference(self):
        self.maximumDifference = 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)
```

# Day 15: Linked List

In [ ]:

```
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
        temp_node = Node(data)
        if head == None:
            head = temp_node
        elif head.next == None:
            head.next = temp_node
        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);
```

# Day 16: Exceptions - String to Integer

In [ ]:

```
#!/bin/python

import sys


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

# Day 17: More Exceptions

In [ ]:

```
#Write your code here
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 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)
```

# Day 18: Queues and Stacks

In [ ]:

```
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();
```

# Day 19

In [ ]:

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

# Day 20

In [ ]:

```
#!/bin/python

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]) )
```

# Day 22

In [ ]:

```
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
```

# Day 23

In [ ]:

```
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])
```

# Day 24

In [ ]:

```
def removeDuplicates(self,head):
    #Write your code here
    current = head
    while (current.next):
        if (current.data == current.next.data):
            current.next = current.next.next
        else:
            current = current.next

    return head
```

# Day 25

In [ ]:

```
import math

def check_prime(num):
    if num is 1:
        return "Not prime"
    sq = int(math.sqrt(num))
    for x in range(2, sq+1):
        if num % x is 0:
            return "Not prime"
    return "Prime"


t = int(input())
for i in range(t):
    number = int(input())
    print(check_prime(number))
```

# Day 26

In [ ]:

```
da, ma, ya = input().split(' ')
da = int(da)
ma = int(ma)
ya = int(ya)
de, me, ye = input().split(' ')
```

```
de = int(de)
me = int(me)
ye = int(ye)
fine = 0
if(ye==ya):
    if(me < ma):
        fine = (ma - me) * 500
    elif((me == ma) and (de < da)):
        fine = (da - de) * 15
elif(ye < ya):
    fine = 10000

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 [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
```

# Day 28

In [ ]:

```
#!/bin/python

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 )
```

# Day 29

In [ ]:

```
#!/bin/python

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)
```