HACKERRANK CODE

Problem 1

Say “Hello, Wolrd!” with Python

```
if __name__ == '__main__':
    print "Hello, World!"
```

Python If-Else

```
#!/bin/python

import math
import os
import random
import re
import sys

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

if n%2 != 0:
    print("Weird")

if n%2 == 0 and 2 <= n <= 5 :
    print("Not Weird")

if n%2 == 0 and 6 <= n <= 20 :
    print("Weird")

if n%2 == 0 and n > 20 :
    print("Not Weird")
```

Arithmetic Operators

```
if __name__ == '__main__':
    a = int(raw_input())
    b = int(raw_input())
```

```
    print(a+b)
    print(a-b)
    print(a*b)
```

Python : Division

```
from __future__ import division

if __name__ == '__main__':
    a = int(raw_input())
    b = int(raw_input())

print(a//b)
print(a/b)
```

Loops

```
if __name__ == '__main__':
    n = int(raw_input())

    i = 0

    for i in range(n) :
        print(i**2)
```

Write a function

```
def is_leap(year):
    leap = False

    if year % 400 == 0:
        leap = True
    elif year % 100 == 0:
        leap = False
    elif year % 4 == 0:
        leap = True

    return leap
```

Print Function

```
from __future__ import print_function

if __name__ == '__main__':
    n = int(raw_input())

for i in range(n):
    print(i+1, end='')
```

List Comprehensions

```
if __name__ == '__main__':
    x = int(raw_input())
    y = int(raw_input())
    z = int(raw_input())
    n = int(raw_input())


print([[i,j,k] for i in range(x+1) for j in range(y+1) for k in
range(z+1) if (i + j + k) != n])
```

Nested Lists

```
if __name__ == '__main__':
    studentsScore = []
    for _ in range(int(raw_input())):
        name = raw_input()
        score = float(raw_input())
        studentsScore.append([name, score])

secondLowest = sorted(set([i[1] for i in studentsScore]))[1]
print("\n".join(sorted([i[0] for i in studentsScore if i[1] ==
secondLowest])))
```

Finding the percentage

```
if __name__ == '__main__':
    n = int(raw_input())
    student_marks = {}
    for _ in range(n):
        line = raw_input().split()
        name, scores = line[0], line[1:]
        scores = map(float, scores)
        student_marks[name] = scores
    query_name = raw_input()

mean =
float(sum(student_marks[query_name])/len(student_marks[query_name]
))

print("{:.2f}".format(mean))
```

Lists

```
if __name__ == '__main__':
    N = int(raw_input())

myList = []

for n in range(N):
    arg = raw_input().strip().split(" ")

    if arg[0] == "insert":
        myList.insert(int(arg[1]), int(arg[2]))
    elif arg[0] == "remove":
        myList.remove(int(arg[1]))
    elif arg[0] == "append":
        myList.append(int(arg[1]))
    elif arg[0] == "sort":
        myList.sort()
    elif arg[0] == "pop":
        myList.pop()
    elif arg[0] == "reverse":
        myList.reverse()
    elif arg[0] == "print":
        print(myList)
```

Tuples

```
if __name__ == '__main__':
    n = int(raw_input())
    integer_list = map(int, raw_input().split())

print(hash(tuple(integer_list)))
```

sWAP cASE

```
def swap_case(s):
    return s.swapcase()
```

String Split and Join

```
def split_and_join(line):
    return "-".join(line.split())
```

What's Your Name?

```
def print_full_name(first, last):
    welcomePhrase = "Hello "+ first + " " + last + "! You just
delved into python."
    print(welcomePhrase)
```

Mutations

```
def mutate_string(string, position, character):
    s = list(string)
    s[position] = character
    string = "".join(s)
    return string
```

Find a string

```
def count_substring(string, sub_string):
    count = 0
    i = 0

    while i < len(string):
        if sub_string in string[i:i+len(sub_string)]:
            count += 1
        i += 1
    return(count)
```

String Validators

```
if __name__ == '__main__':
    s = raw_input()

myList =list(s)
alnum, alpha, digit, lower, upper = False, False, False, False,
False

for i in myList :
    if i.isalnum():
        alnum = True

    if i.isalpha():
        alpha = True

    if i.isdigit():
        digit = True

    if i.islower():
        lower = True

    if i.isupper():
        upper = True

print(alnum)
print(alpha)
print(digit)
print(lower)
print(upper)
```

Text Alignment

```
width = int(input())
c = 'H'

#Top Cone
for i in range(width):
    print((c*i).rjust(width-1)+c+(c*i).ljust(width-1))

#Top Pillars
for i in range(width+1):
    print((c*width).center(width*2)+(c*width).center(width*6))

#Middle Belt
for i in range((width+1)//2):
    print((c*width*5).center(width*6))

#Bottom Pillars
for i in range(width+1):
    print((c*width).center(width*2)+(c*width).center(width*6))

#Bottom Cone
for i in range(width):

print(((c*(width-i-1)).rjust(width)+c+(c*(width-i-1)).ljust(width)
).rjust(width*6))
```

Text Wrap

```
def wrap(string, max_width):
    return textwrap.fill(text = string, width = max_width)
```

String Formatting

```
def print_formatted(number):
    binaryLenght=len(bin(n)[2:])

    for i in range(1,n+1):
        o = oct(i)[2:]
        h = hex(i)[2:].upper()
        b = bin(i)[2:]

        print
(str(i).rjust(binaryLenght),str(o).rjust(binaryLenght),str(h).rjus
t(binaryLenght),str(b).rjust(binaryLenght))
```

Alphabet Rangoli

```
def print_rangoli(size):
    myString = 'abcdefghijklmnopqrstuvwxyz'[0: size]

    for i in range(size-1, -size, -1):
        x = abs(i)
        line = myString[size:x:-1] + myString[x:size]
        print ("--"*x + '-'.join(line) + "--"*x)
```

Capitalize!

```
def solve(s):
    s = s.split(" ")

    return(" ".join(i.capitalize() for i in s))
```

The Minion Game

```
def minion_game(s):

    vowel = ['A','E','I','O','U','Y']
    Stuart = 0
    Kevin = 0
    for i in range(len(s)):
        if s[i] in vowel:
            Kevin += len(s)-i
        else:
            Stuart += len(s)-i

    if Stuart > Kevin:
        print("Stuart" + " " + "%d" % Stuart)

    elif Kevin > Stuart:
        print("Kevin" + " " + '%d' % Kevin)

    else:
        print("Draw")
```

Designer Door Mat

```
n, m = map(int, input().split())
pattern = [('.|.'*(2*i + 1)).center(m, '-') for i in range(n//2)]

print("\n".join(pattern))
print("WELCOME".center(m, "-"))
print("\n".join(pattern[::-1]))
```

Merge the Tools!

```
def merge_the_tools(string, k):
    t = int(len(string)/k)
    for i in range(0,len(string),k):
        line = ""
        for j in range(k):
            if i+j < len(string) and string[i+j] not in line:
                line += string[i+j]
        print(line)
```

Introduction to Sets

```
def average(arr):

    avg = sum(set(arr))/len(set(arr))

    return avg
```

Symmetric Difference

```


M=int(input())
m=input()
N=int(input())
n=input()

x=list(map(int,m.split()))
y=list(map(int,n.split()))

a=set(x)
b=set(y)

c=a.difference(b)
d=b.difference(a)

e=c.union(d)

result=list(e)

result.sort()

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

No Idea!

```
happiness = 0

a,b = map(int, input().split())
array = list(map(int,input().split()))

A = set(map(int, input().split()))
B = set(map(int, input().split()))

for i in array:
    if i in A:
        happiness += 1
    elif i in B:
        happiness -= 1

print(happiness)
```

Set .add()

```
print(len(set([input() for i in range(int(input()))])))
```

Set .discard(), .remove() & .pop()

```
n = int(input())
s = set(map(int, input().split()))


for i in range(int(input())):
    arg = input().split()

    if arg[0] == "remove":
        s.remove(int(arg[1]))
    elif arg[0] == "discard":
        s.discard(int(arg[1]))
    elif arg[0] == "pop":
        s.pop()

print(sum(s))
```

Set .union() Operation

```
n = int(input())
N = set(map(int, input().split()))
b = int(input())
B = set(map(int, input().split()))

print(len(N.union(B)))
```

Set .intersection() Operation

```
n = input()
N = set(map(int, input().split()))
b = input()
B = set(map(int, input().split()))

print(len(N.intersection(B)))
```

Set .difference() Operation

```
n = input()
N = set(map(int, input().split()))
b = input()
B = set(map(int, input().split()))

print(len(N.difference(B)))
```

Set .symmetric_difference() Operation

```
n = input()
N = set(map(int, input().split()))
b = input()
B = set(map(int, input().split()))

print(len(N.symmetric_difference(B)))
```

Set Mutations

```
n = int(input())
s = set(list(map(int, input().split())))

for i in range (int(input())):
    getattr(s, input().split()[0])(list(map(int,
input().split())))
print (sum(s))
```

The Captain's Room

```
n = int(input())
myList = input().split(' ')

roomSet= set()
repeatSet = set()

for i in myList:
    if i not in roomSet:
        roomSet.add(i)
    else:
        repeatSet.add(i)

print(*(roomSet - repeatSet))
```

Check Subset

```
num_test_case = int(input())

for i in range(num_test_case):
  A = int(input())
  a = set(input().split())
  B = int(input())
  b = set(input().split())

  print(a.issubset(b))
```

Check Strict Superset

```
s = set(map(int, input().split()))
n = int(input())
result = True
for _ in range(n):
    newSet = set(map(int, input().split()))
    if result == True:
        result = s.issuperset(newSet)

print(result)
```

collections.Counter()

```
from collections import *


shoesStock = int(input())
sizes = Counter(map(int, input().split()))

bill = 0

for i in range(int(input())):
    size, price = map(int, input().split())

    if sizes[size] > 0:
        sizes[size]-= 1
        bill += price

print(bill)
```

DefaultDict Tutorial

```
from collections import defaultdict

dic = defaultdict(list)

N, M = map(int, input().split())

for i in range(1, N+1):
    dic[input()].append(i)

myList = []

for i in range(M):
    myList.append(input())

for i in myList:
    if i in dic:
        print(*dic[i])
    else:
        print(-1)
```

Collections.namedtuple()

```
from collections import namedtuple

n = int(input())
f = input().split()
student=namedtuple('student',f)
myList = []
totalGrade = 0

for i in range(n):
    f1, f2, f3, f4 = input().split()
    myList.append(student(f1,f2,f3,f4))
    totalGrade += int(myList[i].MARKS)

print(totalGrade/n)
```

Collections.OrderedDict()

```
from collections import OrderedDict

N = int(input())

myDict = OrderedDict()

for i in range(N):
    fast_food = input().rsplit(' ',1)
    fast_food[-1] = int(fast_food[-1])

    if fast_food[0] in myDict:
        myDict[fast_food[0]] += fast_food[1]

    else:
        myDict[fast_food[0]] = fast_food[1]

for key, value in myDict.items():
    print(key, value)
```

Word Order

```
from collections import Counter

myList = []

for _ in range(int(input())):
    myList.append(input())

count = Counter(myList)

print(len(count))
print(*count.values())
```

Collections.deque()

```
from collections import deque

d = deque()

for i in range(int(input())):
    args = input().split()

    if args[0]=="append":
        d.append(args[1])

    elif args[0]=="appendleft":
        d.appendleft(args[1])

    elif args[0]=="pop":
        d.pop()

    elif args[0]=="popleft":
        d.popleft()

print(*d)
```

Company logo

```
from collections import Counter

if __name__ == '__main__':
    s = sorted(input())

    top3 = Counter(s).most_common(3)

for i in top3:
    print('{} {}'.format(*i))
```

Piling Up!

```
T=int(input())

for i in range(T):
    N = int(input())
    myArray = list(map(int,input().split()))
    maxValue = max(myArray)


    if myArray[0] >= maxValue or myArray[-1] >= maxValue:
        print("Yes")

    else:
        print("No")
```

Calendar Module

```
(import calendar

MM, DD, YYYY = map(int, input().split())

wd = calendar.weekday(YYYY, MM, DD)

print(calendar.day_name[wd].upper())
```

Time Delta

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

# Complete the time_delta function below.
def time_delta(t1, t2):

    t1 = datetime.datetime.strptime(t1,'%a %d %b %Y %H:%M:%S %z' )
    t2 = datetime.datetime.strptime(t2,'%a %d %b %Y %H:%M:%S %z' )
```

```
    differenceTime = abs(t1 - t2)
    differenceTime = differenceTime.total_seconds()

    return str(int(differenceTime))

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

    t = int(input())

    for t_itr in range(t):
        t1 = input()

        t2 = input()

        delta = time_delta(t1, t2)

        fptr.write(delta + '\n')

    fptr.close()
```

Exceptions

```
for i in range(int(input())):
    a, b = input().split()
    try:
        print(int(a) // int(b))
    except BaseException as err:
        print("Error Code:", err)
```

Zipped!

```
n, x = map(int, input().split())
grades = []
for i in range(x):
    subjetGrade = map(float, input().split())
    grades.append(subjetGrade)
for k in zip(*grades):
    result = [sum(k)/x]
    print(*result, sep='\n')
```

Athlete Sort

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

if __name__ == '__main__':
    nm = input().split()

    n = int(nm[0])

    m = int(nm[1])

    arr = []

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

    k = int(input())

for i in sorted(arr, key = lambda j:j[k]):
    print(*i)
```

ginortS

```
rule =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1357902468"

print(''.join(sorted(input(), key = rule.index)))
```

Map and Lambda Function

```
cube = lambda x: x**3 # complete the lambda function

def fibonacci(n):
    # return a list of fibonacci numbers

    fib_series = [0,1]
    for i in range(2,n):
       fibNumber = fib_series[i-2]+fib_series[i-1]
       fib_series.append(fibNumber)
    return fib_series[0:n];
```

Detect Floating Point Number

```
myList=[]

for i in range(int(input())):
    myList.append(input())

for i in myList:
    try:
        y = float(i)

        if y==0:
            print("False")

        elif type(y)==float:
            print("True")

    except ValueError as v:
        print("False")
```

Re.split()

```
regex_pattern = r"\D"
```

Group(), Groups() & Groupdict()

```
import re

result = re.search(r'([a-zA-Z0-9])\1+', input().strip())

if result == None:
    print(-1)
else:
    print(result.group()[0])
```

Re.findall() & Re.finditer()

```
import re

m =
re.finditer(r"(?<=[^aeiouAEIOU])([aeiouAEIOU]){2,}(?=[^aeiouAEIOU]
)", input())
myList = [*m]

if len(myList) > 0:
    for s in myList:
        print(s.group())
else:
    print(-1)
```

Re.start() & Re.end()

```
import re
s,k=input(),input()
m=list(re.finditer('(?=(%s))'%k,s))

if not m:
    print((-1,-1))

for i in m:
    print((i.start(1), i.end(1)-1))
```

Regex Substitution

```
import re

for i in range(int(input())):
    line = input()

    for i in range(2):
        line = re.sub(' \&\& ',' and ',line)
        line = re.sub(' \|\| ',' or ',line)

    print(line)
```

Validating phone numbers

```
import re

for i in range(int(input())):
    if re.fullmatch(r'^[7-9]{1}[0-9]{9}', input()):
        print("YES")

    else:
        print("NO")
```

Validating and Parsing Email Addresses

```
import re
import email.utils

for i in range(int(input())):
    z = input()
    s=email.utils.parseaddr(z)
    if z.find(",") == -1:
        if
re.fullmatch("[A-Z|a-z]+[-|_|.|\d|a-z]+@[A-Z|a-z]+.[A-Z|a-z]{1,3}"
,s[1]):
            print(email.utils.formataddr(s))
```

Hex Color Code

```
import re

for i in range(int(input())):
    line = input()
    match = re.findall(r'.(#[0-9a-fA-F]{6}|#[0-9a-fA-F]{3})',
line)

    for m in match:
        print(m)
```

HTML Parser - Part 1

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print ('Start :', tag)
        for args in attrs:
            print ('->', args[0], '>', args[1])

    def handle_endtag(self, tag):
        print ('End   :', tag)

    def handle_startendtag(self, tag, attrs):
        print ('Empty :', tag)
        for args in attrs:
            print ('->', args[0], '>', args[1])

parser = MyHTMLParser()
for i in range(int(input())):
    parser.feed(input())
```

HTML Parser - Part 2

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):

    def handle_comment(self, data):
        if '\n' in data:
            print(">>> Multi-line Comment")
        else:
            print(">>> Single-line Comment")
        print(data)

    def handle_data(self, data):
        if data != '\n':
            print(">>> Data", data, sep='\n')

html = ""
for i in range(int(input())):
    html += input().rstrip()
    html += '\n'

parser = MyHTMLParser()
parser.feed(h
```

Detect HTML Tags, Attributes and Attribute Values

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print (tag)
        for args in attrs:
            print ('->', args[0], '>', args[1])

parser = MyHTMLParser()
for _ in range(int(input())):
    parser.feed(input())
```

Validating UID

```
import re

for _ in range(int(input())):
    myInput = ''.join(sorted(input()))
    try:
        assert re.search(r'[A-Z]{2}', myInput)
        assert re.search(r'\d\d\d', myInput)
        assert not re.search(r'[^a-zA-Z0-9]', myInput)
        assert not re.search(r'(.)\1', myInput)
        assert len(myInput) == 10
    except:
        print('Invalid')
    else:
        print('Valid')
```

XML 1 - Find Score

```
def get_attr_number(node):
    # your code goes here

    score = len(node.attrib)

    for child in node:
        score += get_attr_number(child)

    return score
```

XML 2 - Find the maximum

```
maxdepth = 0
def depth(elem, level):
    global maxdepth
    # your code goes here

    for child in elem:
        depth(child, level+1)
    maxdepth = max(level+1, maxdepth)

    return maxdepth
```

Standardize Mobile Number Using Decorators

```
def wrapper(f):
    def fun(l):
        # complete the function
        l =  ["+91 " + i[-10:-5] + " " + i[-5:] for i in l]
        return f(l)

    return fun
```

Decorators 2 - Name Directory

```
def person_lister(f):
    def inner(people):
        # complete the function
        people.sort(key = lambda x: int(x[2]))
        res =[f(i) for i in people]

        return res

    return inner
```

Arrays

```
def arrays(arr):
    return numpy.array(arr[::-1], float)
```

Shape and Reshape

```
import numpy

arr = input().strip().split(' ')
arr = numpy.array(arr,int)
print(numpy.reshape(arr,(3,3)))
```

Transpose and Flatten

```
import numpy

i, j = input().split(" ")
myArray = [ ]
for k in range(int(i)):
    myArray.append(input().split())

print(numpy.transpose(numpy.array(myArray,int)))
print((numpy.array(myArray,int).flatten()))
```

Concatenate

```
import numpy

n, m, p = map(int,input().split())

array1 = numpy.array([input().split() for _ in range(n)],int)
array2 = numpy.array([input().split() for _ in range(m)],int)

print(numpy.concatenate((array1, array2)))
```

Zeros and Ones

```
import numpy

d = tuple(map(int,(input().split())))
array1 = numpy.zeros((d), dtype = numpy.int)
array2 = numpy.ones((d), dtype = numpy.int)

print(array1)
print(array2)
```

Eye and Identity

```
import numpy

N, M = map(int,input().split(" "))
numpy.set_printoptions(sign=' ')
print(numpy.eye(N, M))
```

Array Mathematics

```
import numpy

N, M = map(int, input().split())
A= numpy.array([list(map(int, input().split())) for i in
range(N)], int)
B = numpy.array([list(map(int, input().split())) for i in
range(N)], int)

print(numpy.add(A,B), numpy.subtract(A,B), numpy.multiply(A,B),
numpy.floor_divide(A,B), numpy.mod(A,B), numpy.power(A,B), sep =
"\n")
```

Floor, Ceil and Rint

```
import numpy

numpy.set_printoptions(sign=' ')

myArray = numpy.array(input().split(),float)

print(numpy.floor(myArray), numpy.ceil(myArray),
numpy.rint(myArray), sep='\n')
```

Min and Max

```
import numpy

arr = []
n, m = map(int,input().split())

for row in range(n):
    myArray = list(map(int,input().split()))
    arr.append(myArray)

myArrayay = numpy.array(arr)
result = numpy.min(myArrayay, axis = 1)

print(numpy.max(result))
```

Mean, Var and Std

```
import numpy

arr = []
n, m = map(int,input().split())
for row in range(n):
    myArray = list(map(int,input().split()))
    arr.append(myArray)

print(numpy.mean(arr, axis = 1))
print(numpy.var(arr, axis = 0))
print(numpy.around(numpy.std(arr, axis = None), 11))
```

Dot and Cross

```
import numpy

n = int(input())
arrayA = []
arrayB = []

for i in range(n):
    arr1 = list(map(int,input().split()))
    arrayA.append(arr1)

for i in range(n):
    arr2 = list(map(int,input().split()))
    arrayB.append(arr2)

print(numpy.dot(arrayA, arrayB))
```

Inner and Outer

```
import numpy

arrayA = numpy.array(input().split(),int)
arrayB = numpy.array(input().split(),int)

print(numpy.inner(arrayA,arrayB))
print(numpy.outer(arrayA,arrayB))
```

Polynomials

```
import numpy

myArray = numpy.array(input().split(), float)
n = float(input())

print(numpy.polyval(myArray, n))
```

Linear Algebra

```
import numpy

n = int(input())
arrayA = []

for i in range(n):
    arr1 = list(map(float,input().split()))
    arrayA.append(arr1)

print(numpy.around(numpy.linalg.det(arrayA), 2))
```

Problem 2

Birthday Cake Candles

```python
#!/bin/python3

import math
import os
import random
import re
import sys

def birthdayCakeCandles(candles):
    # Write your code here
    return candles.count(max(candles))

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

    candles_count = int(input().strip())

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

    result = birthdayCakeCandles(candles)

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

    fptr.close()
```

Number Line Jumps

```python
#!/bin/python3

import math
import os
import random
import re
import sys

def kangaroo(x1, v1, x2, v2):
    # Write your code here
    position = "NO"
    x = int(x2-x1)
    v = int(v1-v2)

    if v1 > v2 and x % v == 0:
        position = "YES"

    return position

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

    first_multiple_input = input().rstrip().split()

    x1 = int(first_multiple_input[0])

    v1 = int(first_multiple_input[1])

    x2 = int(first_multiple_input[2])

    v2 = int(first_multiple_input[3])

    result = kangaroo(x1, v1, x2, v2)

    fptr.write(result + '\n')

    fptr.close()
```

Viral Advertising

```python
#!/bin/python3

import math
import os
import random
import re
import sys


def viralAdvertising(n):
    given_value = 5
    likes = 0

    for i in range(n):
        likes += given_value//2
        given_value = given_value//2*3

    return likes

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

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

    result = viralAdvertising(n)

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

    fptr.close()
```

Recursive Digit Sum

```python
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'superDigit' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. STRING n
#  2. INTEGER k
#

def superDigit(n, k):
    # Write your code here
    if len(n) == 1 and k == 1:
        return int(n)

    myList = list(map(int,n))
    x = str(sum(myList*k))
    result = superDigit(x,1)
    return result


if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    first_multiple_input = input().rstrip().split()
    n = first_multiple_input[0]
    k = int(first_multiple_input[1])
    result = superDigit(n, k)

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

    fptr.close()
```

Insertion Sort - Part 1

```
def insertionSort1(n, arr):
    i = n - 1
    value = arr[i]
    while(i > 0 and value < arr[i-1]):
        arr[i] = arr[i-1]
        print(*arr)
        i -= 1
    arr[i] = value
    print(*arr)

if __name__ == '__main__':
    n = int(input().strip())
    arr = list(map(int, input().rstrip().split()))
    insertionSort1(n, arr)
```

Insertion Sort - Part 2

```
def insertionSort2(n, arr):
    for i in range(1,n):
        left = arr[0:i+1]
        left = sorted(left)
        res = left + arr[i+1:n]
        print(*res)

if __name__ == '__main__':
    n = int(input().strip())
    arr = list(map(int, input().rstrip().split()))
    insertionSort2(n, arr)
```