Problem-1: Breaking the records.

Maria plays college basketball and wants to go pro. Each season she maintains a record of her play. She tabulates the number of times she breaks her season record for most points and least points in a game. Points scored in the first game establish her record for the season, and she begins counting from there.

Example

scores = [12, 24, 10, 24]

Scores are in the same order as the games played. She tabulates her results as follows:

Game	Score	Minimum	Maximum	Count Min	Count Max
0	12	12	12	0	0
1	24	12	24	0	1
2	10	10	24	1	1
3	24	10	24	1	1

Given the scores for a season, determine the number of times Maria breaks her records for most and least points scored during the season.

Function Description

Complete the breakingRecords function in the editor below.

breakingRecords has the following parameter(s):

- n: no of games
- int scores[n]: points scored per game

Returns

- int[2]: An array with the numbers of times she broke her records. Index 0 is for breaking most points records, and index 1 is for breaking least points records.

Input Format

n, the number of games. Integers array osf scores describing the respective values of $score_0, score_1, score_2, ..., score_{n-1}$$score_0, score_1, score_2, ..., score_{n-1}$.

Constraints

- $1 <= n <= 1000$$1 <= n <= 1000$
- $0 <= scores[i] <= 10^8$$0 <= scores[i] <= 10^8$

Sample Input 0

```
10 5 20 20 4 5 2 25 1
```

Sample Output 0

```
2 4
```

Explanation 0

The diagram below depicts the number of times Maria broke her best and worst records throughout the season:

Games	0	1	2	3	4	5	6	7	8
Score	10	5	20	20	4	5	2	25	1
Highest Score	10	10	**20**	20	20	20	20	**25**	25
Lowest Score	10	**5**	5	5	**4**	4	**2**	2	**1**

She broke her best record twice (after games 2 and 7) and her worst record four times (after games 1, 4, 6, and 8), so we print 2 4 as our answer. Note that she did not break her record for best score during game 3, as her score during that game was not strictly greater than her best record at the time.

```
#Solution

def breakingRecords(n, scores):
    # Write your code here
    # Write your code here
    result = [0,0]
    t = scores[0]
    a = t
    b = t
    for i in scores:
        if i > a:
            result[0] = result[0]+1
            a = i
        if i < b:
            result[1] = result[1]+1
            b = i
    return result


scores = [9, 10, 5, 20, 20, 4, 5, 2, 25, 1]
n=len(scores)
result = breakingRecords(n, scores)
```

Problem–2: Student management system in Python

Write a program to build a simple Student Management System using Python which can perform the following operations:

1. **Accept -** This method takes details from the user like name, roll number, and marks for two different subjects.
2. **Display -** This method displays the details of every student.

3. **Search -** This method searches for a particular student from the list of students. This method will ask the user for roll number and then search according to the roll number
4. **Delete -** This method deletes the record of a particular student with a matching roll number.
5. **Update -** This method updates the roll number of the student. This method will ask for the old roll number and new roll number. It will replace the old roll number with a new roll number.

```
class Student:
    # Constructor
    def __init__(self, name, rollno, m1, m2):
        self.name = name
        self.rollno = rollno
        self.m1 = m1
        self.m2 = m2

    # Function to create and append new student
    def accept(self, Name, Rollno, marks1, marks2):

        # use ' int(input()) ' method to take input from user
        ob = Student(Name, Rollno, marks1, marks2)
        ls.append(ob)

    # Function to display student details
    def display(self, ob):
        print("Name : ", ob.name)
        print("RollNo : ", ob.rollno)
        print("Marks1 : ", ob.m1)
        print("Marks2 : ", ob.m2)
        print("\n")

    # Search Function
    def search(self, rn):
        for i in range(ls.__len__()):
            if(ls[i].rollno == rn):
                return i

    # Delete Function
    def delete(self, rn):
        i = obj.search(rn)
        del ls[i]

    # Update Function
    def update(self, rn, No):
        i = obj.search(rn)
        roll = No
        ls[i].rollno = roll


# Create a list to add Students
ls = []
# an object of Student class
obj = Student('', 0, 0, 0)

print("\nOperations used, ")
print("\n1.Accept Student details\n2.Display Student Details\n3.Search Details of

# ch = int(input("Enter choice:"))
# if(ch == 1):
obj.accept("A", 1, 100, 100)
obj.accept("B", 2, 90, 90)
obj.accept("C", 3, 80, 80)
```

```
# elif(ch == 2):
print("\n")
print("\nList of Students\n")
for i in range(ls.__len__()):
    obj.display(ls[i])

# elif(ch == 3):
print("\n Student Found, ")
s = obj.search(2)
obj.display(ls[s])

# elif(ch == 4):
obj.delete(2)
print(ls.__len__())
print("List after deletion")
for i in range(ls.__len__()):
    obj.display(ls[i])

# elif(ch == 5):
obj.update(3, 2)
print(ls.__len__())
print("List after updation")
for i in range(ls.__len__()):
    obj.display(ls[i])

# else:
print("Thank You !")
```

```
    Operations used,

    1.Accept Student details
    2.Display Student Details
    3.Search Details of a Student
    4.Delete Details of Student
    5.Update Student Details
    6.Exit



    List of Students

    Name :  A
    RollNo :  1
    Marks1 :  100
    Marks2 :  100


    Name :  B
    RollNo :  2
    Marks1 :  90
    Marks2 :  90


    Name :  C
    RollNo :  3
    Marks1 :  80
```

```
Marks2 :  80



 Student Found,
Name :  B
RollNo :  2
Marks1 :  90
Marks2 :  90


2
List after deletion
Name :  A
RollNo :  1
Marks1 :  100
Marks2 :  100


Name :  C
RollNo :  3
Marks1 :  80
Marks2 :  80


2
List after updation
Name :  A
RollNo :  1
```

Problem-3: Get current times of different time zones in Python.

```
from datetime import datetime
import pytz
# get the standard UTC time
UTC = pytz.utc
# it will get the time zone
# of the specified location
IST = pytz.timezone('Asia/Kolkata')
# print the date and time in standard format
print("UTC in Default Format : ", datetime.now(UTC))
print("IST in Default Format : ", datetime.now(IST))
# print the date and time in
# specified format
datetime_utc = datetime.now(UTC)
print("Date & Time in UTC : ", datetime_utc.strftime('%Y:%m:%d %H:%M:%S %Z %z'))
datetime_ist = datetime.now(IST)
print("Date & Time in IST : ", datetime_ist.strftime('%Y:%m:%d %H:%M:%S %Z %z'))
```

```
UTC in Default Format :  2022-12-11 13:09:56.229316+00:00
IST in Default Format :  2022-12-11 18:39:56.231325+05:30
Date & Time in UTC :  2022:12:11 13:09:56 UTC +0000
Date & Time in IST :  2022:12:11 18:39:56 IST +0530
```

Problem-4: Python program to count Even and Odd numbers in a List using list comprehension.

Given a list of numbers, write a Python program to count Even and Odd numbers in a List.

Example:

```
Input: list1 = [2, 7, 5, 64, 14]
Output: Even = 3, odd = 2

Input: list2 = [12, 14, 95, 3]
Output: Even = 2, odd = 2
```

```
list1 = [10, 21, 4, 45, 66, 93, 11]

only_odd = [num for num in list1 if num % 2 == 1]
odd_count = len(only_odd)

print("Even numbers in the list: ", len(list1) - odd_count)
print("Odd numbers in the list: ", odd_count)
```

```
Even numbers in the list:  3
Odd numbers in the list:  4
```

Problem-5: Alphabet Rangoli

You are given an integer, $N$$N$. Your task is to print an alphabet rangoli of size $N$$N$. (Rangoli is a form of Indian folk art based on creation of patterns.)

Different sizes of alphabet rangoli are shown below:

```
# size 3

----c----
--c-b-c--
c-b-a-b-c
--c-b-c--
----c----

# size 5

--------e--------
------e-d-e------
----e-d-c-d-e----
--e-d-c-b-c-d-e--
```

```
e-d-c-b-a-b-c-d-e
--e-d-c-b-c-d-e--
----e-d-c-d-e----
------e-d-e------
--------e--------
```

```
# size 10
```

```
------------------j------------------
----------------j-i-j----------------
--------------j-i-h-i-j--------------
------------j-i-h-g-h-i-j------------
----------j-i-h-g-f-g-h-i-j----------
--------j-i-h-g-f-e-f-g-h-i-j--------
------j-i-h-g-f-e-d-e-f-g-h-i-j------
----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----
--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--
j-i-h-g-f-e-d-c-b-a-b-c-d-e-f-g-h-i-j
--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--
----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----
------j-i-h-g-f-e-d-e-f-g-h-i-j------
--------j-i-h-g-f-e-f-g-h-i-j--------
----------j-i-h-g-f-g-h-i-j----------
------------j-i-h-g-h-i-j------------
--------------j-i-h-i-j--------------
----------------j-i-j----------------
------------------j------------------
```

The center of the rangoli has the first alphabet letter a, and the boundary has the $N^{th}$$N^{th}$alphabet letter (in alphabetical order).

```
def print_rangoli(size):
    # your code goes here
    for i in range(size):
        s = "-".join(chr(ord('a')+size-j-1) for j in range(i+1))
        print((s+s[::-1][1:]).center(4*size-3, "-"))
    for i in range(size-1):
        s = "-".join(chr(ord('a')+size-j-1) for j in range(size-i-1))
        print((s+s[::-1][1:]).center(4*size-3, "-"))

if __name__ == '__main__':
    n = int(input())
    print_rangoli(n)
```

```
7
------------g------------
----------g-f-g----------
--------g-f-e-f-g--------
------g-f-e-d-e-f-g------
```

```
----g-f-e-d-c-d-e-f-g----
--g-f-e-d-c-b-c-d-e-f-g--
g-f-e-d-c-b-a-b-c-d-e-f-g
--g-f-e-d-c-b-c-d-e-f-g--
----g-f-e-d-c-d-e-f-g----
------g-f-e-d-e-f-g------
--------g-f-e-f-g--------
----------g-f-g----------
------------g------------
```

Problem-6: Compress a string!

You are given a string $S$$$S$. Suppose a character '$c$$c$' occurs consecutively $X$$$X$ times in the string. Replace these consecutive occurrences of the character '$c$$c$' with $$(X, c) in the string. For a better understanding of the problem, check the explanation.

Input Format:

A single line of input consisting of the string $S$$$S$.

Output Format:

A single line of output consisting of the modified string.

Sample Input:

```
1222311
```

Sample Output:

```
(1, 1) (3, 2) (1, 3) (2, 1)
```

Explanation:

First, the character occurs only once. It is replaced by (1, 1). Then the character 2 occurs three times, and it is replaced by (3, 2)and so on. Also, note the single space within each compression and between the compressions.

```
s = input()
occurrences = {}
alloccurrences = []
occurrences[s[0]] = 0
for i in s:
    if (i in occurrences):
        occurrences[i] =  occurrences[i] + 1
    else :
        (k, v), = occurrences.items()
        alloccurrences.append(str((v,int(k))))
        occurrences.clear()
        occurrences[i] = 1
(k, v), = occurrences.items()
alloccurrences.append(str((v,int(k))))
print(' '.join(alloccurrences))
```

Problem-7: Word order

You are given words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

*Note: Each input line ends with a "**\n**" character.*

Input Format:

The first line contains the integer, $n$$n$.

The next $n$$n$ lines each contain a word.

Output Format:

Output $2$$2$ lines.

On the first line, output the number of distinct words from the input.

On the second line, output the number of occurrences for each distinct word according to their appearance in the input.

Sample Input:

```
4
bcdef
abcdefg
bcde
bcdef
```

Sample output:

```
3
2 1 1
```

Explanation:

There are $3$$3$ distinct words. Here, "bcdef" appears twice in the input at the first and last positions. The other words appear once each. The order of the first appearances are "bcdef", "abcdefg" and "bcde" which corresponds to the output.

```
from collections import Counter
l = []
for i in range(int(input())):
    l.append(input())
x = Counter(l)
print(len(x))
print(*x.values())
```

```
4
bcdef
abcdefg
bcde
bcdef
3
2 1 1
```

Problem–8: Iterables & Iterators

You are given a list of $N$$N$ lowercase English letters. For a given integer $K$$K$, you can select any $K$$K$ indices (assume 1-based indexing) with a uniform probability from the list. Find the probability that at least one of the $K$$K$ indices selected will contain the letter '$a$$a$'.

Input Format:

The input consists of three lines. The first line contains the integer $N$$N$, enoting the length of the list. The next line consists of &N& space-separated lowercase English letters, denoting the elements of the list.

The third and the last line of input contains the integer $K$$K$, denoting the number of indices to be selected.

Output Format:

Output a single line consisting of the probability that at least one of the $K$$K$ indices selected contains the letter '$a$$a$'.

Note: The answer must be correct up to 3 decimal places.

Sample Input:

```
4
a a c d
```

```
2
```

Sample Output:

```
0.8333
```

Explanation:

All possible unordered tuples of length 2 comprising of indices from 1 to 4 are: (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)

Out of these 6 combinations, 5 of them contain either index 1 or index 2 which are the indices that contain the letter '$a$$a$'. Hence, the answer is $\frac{5}{6}$$\frac{5}{6}$

```
from itertools import combinations as cm
n = int(input())
s = list(input().split(" "))
k = int(input())
l=["".join(word) for word in list(cm(s,k))]
x,y=0,len(l)
for word in l:
    x+=1 if 'a' in word else 0
print(x/y)
```

Problem–9: The Captain's Room

Mr. Anant Asankhya is the manager at the INFINITE hotel. The hotel has an infinite amount of rooms.

One fine day, a finite number of tourists come to stay at the hotel. The tourists consist of:

→ A Captain.

→ An unknown group of families consisting of $K$$K$ members per group where $K ≠ 1$$K ≠ 1$

The Captain was given a separate room, and the rest were given one room per group.

Mr. Anant has an unordered list of randomly arranged room entries. The list consists of the room numbers for all of the tourists. The room numbers will appear $K$$K$ times per group except for the Captain's room.

Mr. Anant needs you to help him find the Captain's room number. The total number of tourists or the total number of groups of families is not known to you. You only know the value of $K$$K$ and the room number list.

Input Format:

The first line consists of an integer, $K$$K$, the size of each group.

The second line contains the unordered elements of the room number list.

Output Format:

Output the Captain's room number.

Sample Input:

```
5
1 2 3 6 5 4 4 2 5 3 6 1 6 5 3 2 4 1 2 5 1 4 3 6 8 4 3 1 5 6 2
```

Sample Output

```
8
```

Explanation:

The list of room numbers contains 31 elements. Since K is 5, there must be groups 6 of families. In the given list, all of the numbers repeat 5 times except for room number 8. Hence, 8 is the Captain's room number.

```
from collections import Counter


N = int(input())
count = Counter(map(int, input().split()))

print(next(k for k, v in count.items() if v != N))
```

Problem-10: Classes: Dealing with Complex Numbers

For this challenge, you are given two complex numbers, and you have to print the result of their addition, subtraction, multiplication, division and modulus operations.

The real and imaginary precision part should be correct up to two decimal places.

Input Format:

One line of input: The real and imaginary part of a number separated by a space.

Output Format:

For two complex numbers $C$$C$ and $D$$D$, the output should be in the following sequence on separate lines:

- $C + D - $$C + D - $C - D
- $C * D$$C * D$
- $C / D$$C / D$
- $mod(C)$$mod(C)$

- $mod(D)$$mod(D)$

For complex numbers with non-zero real ($A$$A$) and complex part ($B), the output should be in the following format:

$A + Bi$$A + Bi$

Replace the plus symbol ($+$$+$) with a minus symbol ($-$$-$) when $B<0$$B<0$.

For complex numbers with a zero complex part i.e. real numbers, the output should be:

$A + 0.00i$$A + 0.00i$

For complex numbers where the real part is zero and the complex part ($B$$B$) is non-zero, the output should be:

$0.00 + Bi$$0.00 + Bi$

Sample Input:

```
2 1
5 6
```

Sample Output:

```
7.00+7.00i
-3.00-5.00i
4.00+17.00i
0.26-0.11i
2.24+0.00i
7.81+0.00i
```

Concept:

Solve this problem using Object Oriented way. Methods with a double underscore before and after their name are considered as built-in methods. They are used by interpreters and are generally used in the implementation of overloaded operators or other built-in functionality.

```
__add__-> Can be overloaded for + operation

__sub__ -> Can be overloaded for - operation

__mul__ -> Can be overloaded for * operation
```

```
import math

class Complex(object):
    def __init__(self, real, imaginary):
        self.real, self.imaginary = real, imaginary

    def __add__(self, no):
        real = self.real + no.real
        imag = self.imaginary + no.imaginary
        return __class__(real, imag)

    def __sub__(self, no):
        real = self.real - no.real
        imag = self.imaginary - no.imaginary
        return __class__(real, imag)

    def __mul__(self, no):
        real = self.real*no.real - self.imaginary*no.imaginary
        imag = self.real*no.imaginary + no.real*self.imaginary
        return __class__(real, imag)

    def __truediv__(self, no):
        real = self.real*no.real + self.imaginary*no.imaginary
        real /= pow(no.real, 2) + pow(no.imaginary, 2)
        imag = self.imaginary*no.real - self.real*no.imaginary
        imag /= pow(no.real, 2) + pow(no.imaginary, 2)
        return __class__(real, imag)

    def mod(self):
        real = math.sqrt(pow(self.real,2) + pow(self.imaginary, 2))
        imag = 0.00
        return __class__(real, imag)

    def __str__(self):
        if self.imaginary == 0:
            result = "%.2f+0.00i" % (self.real)
        elif self.real == 0:
            if self.imaginary >= 0:
                result = "0.00+%.2fi" % (self.imaginary)
            else:
                result = "0.00-%.2fi" % (abs(self.imaginary))
        elif self.imaginary > 0:
            result = "%.2f+%.2fi" % (self.real, self.imaginary)
        else:
            result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))
        return result

if __name__ == '__main__':
    c = map(float, input().split())
    d = map(float, input().split())
    x = Complex(*c)
    y = Complex(*d)
    print(*map(str, [x+y, x-y, x*y, x/y, x.mod(), y.mod()]), sep='\n')
```

Problem–11: Find the Torsional angle

You are given four points $A, B, C$$A, B, C$ and $D$$D$ in a 3-dimensional Carterian coordinate system. You are required to print the angle between the plane made by the points $A, B, C$$A, B, C$ and $B, C, D$$B, C, D$ in degrees (**not radians**). Let the angle be $\phi$$\phi$.

$cos(\phi) = (X.Y)/|X||Y|$$cos(\phi) = (X.Y)/|X||Y|$ where $X = AB * BC$$X = AB * BC$ and $Y = BC * CD$$Y = BC * CD$

Here, $X.Y$$X.Y$ means the dot product of $X$$X$ and $Y$$Y$ and $AB*BC$$AB*BC$ means the cross product of vectors $AB$$AB$ and $BC$$BC$. Also, $AB = B - A$$AB = B - A$

Input Format:

One line of input containing the space separated floating number values of the $X, Y$$X, Y$ and $Z$$Z$ coordinates of a point.

Output Format:

Output the angle correct up to two decimal places.

Sample Input:

```
0 4 5
1 7 6
0 5 9
1 7 2
```

Sample Output:

```
8.19
```

```
from math import sqrt


class Points(object):
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def __sub__(self, no):
        return Points(self.x - no.x,
                      self.y - no.y,
                      self.z - no.z)

    def dot(self, no):
        return self.x * no.x + self.y * no.y + self.z * no.z

    def cross(self, no):
        return Points(self.y * no.z - self.z * no.y,
                      self.z * no.x - self.x * no.z,
                      self.x * no.y - self.y * no.x)

    def absolute(self):
        return sqrt((self.x ** 2 + self.y ** 2 + self.z ** 2))

if __name__ == '__main__':
    points = list()
    for i in range(4):
        a = list(map(float, input().split()))
        points.append(a)

    a, b, c, d = Points(*points[0]), Points(*points[1]), Points(*points[2]), Poin
    x = (b - a).cross(c - b)
    y = (c - b).cross(d - c)
    angle = math.acos(x.dot(y) / (x.absolute() * y.absolute()))

    print("%.2f" % math.degrees(angle))
```

Problem-12: Maximize it

You are given a function $f(X) = X^2$$f(X) = X^2$. You are also given $K$$K$ lists. The $i^{th}$$i^{th}$ list consists of $N_i$$N_i$ elements. You have to pick one element from each list so that the value from the equation below is maximized:

$S = (f(X_1) + f(X_2) + ... + f(N_k)) \% M $$S = (f(X_1) + f(X_2) + ... + f(N_k)) \% M $

$X_i$$X_i$ denotes the element picked from the $i^{th}$$i^{th}$ list. Find the maximized value $S_{max}$$S_{max}$ obtained.

$\%$$\%$ denotes the modulo operator.

Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The

maximum value that you can obtain, will be the answer to the problem.

Input Format:

The first line contains 2 space separated integers $K$$K$ and $M$$M$.

The next $K$$K$ lines each contains an integer $N_i$$N_i$, denoting the number of elements in the $i^{th}$$i^{th}$ list, followed by $N_i$$N_i$ space separated integers denoting the elements in the list.

Output Format:

Output a single integer denoting the value $S_max$$S_max$

Sample Input:

```
3 1000
2 5 4
3 7 8 9
5 5 7 8 9 10
```

Sample Output:

```
206
```

Explanation:

Picking 5 from $1^{st}$$1^{st}$ list, 9 from the $2^{nd}$$2^{nd}$ list and 10 from the $3^{rd}$$3^{rd}$ list gives the maximum $S$$S$ value equal to $(5^2 + 9^2 + 10^2)\%1000 = 206$$(5^2 + 9^2 + 10^2)\%1000 = 206$

```
from itertools import product

x, n = map(int, input().split())

l = [list(map(lambda y: (int(y)**2)%n, input().split()))[1:] for i in range(x)]
bestes = [sum(el)%n for el in product(*l)]
print(max(bestes))
```

Problem–13: Diamond shape

Write a python program to print the diamond shape. Given a number n, write a program to print a diamond shape with 2n rows.

```
Input : 5
Output :
        *
       * *
      * * *
     * * * *
    * * * * *
    * * * * *
     * * * *
      * * *
       * *
        *
```

```
def gotonextLine(k, i, z):
    # base case
    if (k == i):
        return
    print("* ", end="")
    gotonextLine(k + z, i, z)

# print blank space of diamond
def addblankSpaceInDiamond(j,i,z):
    if (j == i):
        return
    print(" ",end=""),
    addblankSpaceInDiamond(j + z, i, z)

def upperDiamond(row,i):
    # base case
    if (i > row):
        return
    addblankSpaceInDiamond(row, i, -1)
    gotonextLine(0, i, 1)
    print("\n",end=""),
    upperDiamond(row, i + 1) # recursive call


def lowerDiamond(row,i):
    # print the next line of diamond
    if (i > row): # base case
        return
    addblankSpaceInDiamond(0, i, 1)
    gotonextLine(row, i, -1)
    print("\n",end=""),
    lowerDiamond(row, i + 1)
# Code
row = 5
upperDiamond(row, 0) # print uper part of triangle
lowerDiamond(row, 1) # print lower part of diamond
```

```
        *
       * *
```

```
   * * *
  * * * *
 * * * * *
  * * * *
   * * *
    * *
     *
```

Problem–14: Write a python function to print the half-diamond star pattern.

Given an integer N, the task is to print half-diamond-star pattern.

```
*
**
***
****
*****
******
*****
****
***
**
*
```

```
def halfDiamondStar(N):
    # Loop to print the upper half
    # diamond pattern
    for i in range(N):
        for j in range(0, i + 1):
            print("*", end = "")
        print()

    # Loop to print the lower half
    # diamond pattern
    for i in range(1, N):
        for j in range(i, N):
            print("*", end = "")
        print()

halfDiamondStar(6)
```

```
*
**
***
****
*****
******
*****
```

```
****
***
**
*
```

Problem–15: Write a Python program to find the power of a given number using recursion.

Given a number N and power P, the task is to find the power of a number (i.e. N^P) using recursion.

```
def power(N, P):
    # if power is 0 then return 1 if the condition is true
    if P == 0:
        return 1

    # recurrence relation
    return (N * power(N, P-1))

if __name__ == "__main__":
    N = 5
    P = 2
    print(power(N, P))
```

```
25
```

Problem–16: Largest number in a list

Write a Python program to find the largest number in a list without using built-in methods.

```
numbers = [3, 8, 1, 7, 2, 9, 5, 4]
def max_list(my_list: list):
    big = my_list[0]
    position = 0

    for index, item in enumerate(my_list):
        if item > big:
            big = item
            position = index

    return position, big

position, big = max_list(numbers)

print(f"The largest element is {big} which is found at position {position}")
```

```
The largest element is 9 which is found at position 5
```

Problem–17: Reverse integer

Write a python function that accepts any length of the integers and prints that integer in a reverse manner. Don't use any python trick or python special syntax.

```
def reverse_int(number: int) ->int:
    rev = 0
    while number != 0:
        digit = number % 10
        rev = (rev * 10) + digit
        number = number // 10
    return rev

number = int(input("Enter a positive integer: "))
print(reverse_int(number))
```

```
Enter a positive integer: 9474158862
2688514749
```

Problem–18: Celsius to Farenhight

Write a Python function that can convert the degree of Celsius to the Farenhight scale.

```
def celsius_to_farenhight(celsius: float) -> float:
    farenhight = (9 * celsius)/5 + 32
    return farenhight

c = float(input("Enter temperature in Centigrade: "))
print(f"In Farenhight scale, the temp is {celsius_to_farenhight(c)}")
```

```
Enter temperature in Centigrade: 17
In Farenhight scale, the temp is 62.6
```

Problem–19: is it right?

You are given two values $a$$a$ and $b$$b$. Perform integer division and print $a/b$$a/b$.

Input Format:

The first line contains $T$$T$, the number of test cases.

The next $T$$T$ lines each contain the space separated values of $a$$a$ and $b$$b$.

Output Format:

Print the value of $a/b$$a/b$.

In the case of *ZeroDivisionError* or *ValueError*, print the error code.

Sample Input:

```
3
1 0
2 5
3 1
```

Sample Output:

```
Error Code: integer division or modulo by zero
Error Code: invalid literal for int() with base 10: '$'
3
```

```
for i in range(int(input())):
    a,b=list(map(str,input().split()))
    try:
        print(int(int(a)/int(b)))
    except:
        if b!="0":
            print(f"Error Code: invalid literal for int() with base 10: '{b if a.
        else:
            print(f"Error Code: integer division or modulo by zero")
```

Problem 20: Piling Up!

There is a horizontal row of $n$$n$ cubes. The length of each cube is given. You need to create a new vertical pile of cubes. The new pile should follow these directions: if $cube[i]$$cube[i]$ is on top of $sube[j]$$sube[j]$ then $sideLength[j] >= sigeLength[i]$$sideLength[j] >= sigeLength[i]$.

When stacling the cubes, you can only pick up either leftmost or the rightmost cube each time. Print Yes if it possible to stack the cubes. Otherwise, print No.

Example:

$blocks = [1, 2, 3, 8, 7]$$blocks = [1, 2, 3, 8, 7]$

Result: No

After choosing the rightmost element 7, choose the leftmost element, 1. After than, the choices are 2 and 8. These are both larger than the top block of size 1.

$blocks = [1, 2, 3, 7, 8]$$blocks = [1, 2, 3, 7, 8]$

Result: Yes

Choose blocks from right to left in order to successfully stack the blocks.

Input Format:

The first line contains a single integer $T$$T$, the number of test cases.

For each test case, there are 2 lines.

The first line of each test case contains $n$$n$, the number of cubes.

The second line contains $n$$n$ space separated integers, denoting the sideLengths of each cube in that order.

Output Format:

For each test case, output a single line containing either `Yes` or `No`.

Sample Input:

```
STDIN       Function
-----       --------
2           T = 2
6           blocks[] size n = 6
4 3 2 1 3 4 blocks = [4, 3, 2, 1, 3, 4]
3           blocks[] size n = 3
1 3 2       blocks = [1, 3, 2]
```

Sample Output:

```
Yes
No
```

Explanation:

In the first test case, pick in this order: *left-4, right-4, left-3, right-3,left-2, right-1*

In the second test case, no order gives an appropriate arrangement of vertical cubes. 3 will always come after either 1 or 2.

```
for _ in range(int(input())):
    size = int(input())
    block = list(map(int, input().split()))
    if block[0] < max(block) > block[-1]:
        print('No')
    else:
        print('Yes')
```

Problem 21 The following Python code represents a Tic-Tac-Toe board as a list of

lists: `[ [ '#' , 'o' , 'x' ] , [ '#' , '#' , 'o' ] , [ 'x' , '#' , 'o' ] ]`

The `#` symbols represent blank squares on the board. Write a function print_board that takes a list of lists as an argument and prints out a Tic-Tac-Toe board in the following format:

```
   | o | x
 ------------
   |   | o
 ------------
  x |   | o
```

```
#Solution
def print_board ( board ) :
    for row in board:
        row_txt=''
        for i in row:
            if i=='#':
                row_txt += '| {} '.format(' ')
            else:
                row_txt += '| {} '.format(i)
        print(row_txt[1:])
        print('------------')

    print() # Print an empty line

print_board([ [ '#' , 'o' , 'x' ] , [ '#' , '#' , 'o' ] , [ 'x' , '#' , 'o' ] ])
```

```
   | o | x
------------
   |   | o
------------
 x |   | o
------------
```

Problem 22: Wobbly Number

A “wobbly” number is one in which the digits alternate between being higher and lower than the preceding one. Here are some wobbly numbers: `19284756242, 90909, 0909`. Write a function that accepts a list of digits to be checked for wobbliness. If the sequence of digits is wobbly, the function should return True, otherwise False.

```
#Soltuion
def isWobbly ( num ) :
    if num == '' :
        return False
    elif not ( len ( num ) > 2 ) :
        return False
    if not ( num[0].isdigit ( ) or num[1].isdigit ( ) ) :
        return False
    isLower = int ( num[0] ) < int ( num[1] )
    curr = int ( num[1] )
    for i in num[2:] :
        if not i.isdigit( ) :
            return False
        d = int ( i )
        if ( curr < d ) == isLower :
            return False
        elif curr == d :
            return False
        isLower = curr < d
        curr = d
    return True

a = ['19284756242', '90909', '0909']
for i in a:
    print(i, isWobbly(i))
```

```
19284756242 True
90909 True
0909 True
```

Problem 23–25

Imagine a fictional land where monetary units aren't based on decimal orders. In this land, we have 3 basic units of currency:

- The Blink. The smallest unit of currency.
- The Hoojim. Worth 12 Blinks.
- The Bung. Worth 20 Hooja (plural of Hoojim) or 240 Blinks.

Problem–23 Write a function called deBung

Write a function called deBung that accepts an integer representing a number of Bungs and displays the number of Hooja and Blink it is worth.

For example,calling the function like this:

```
deBung ( 4 )
```

Will produce the following output: `4 Bungs is worth 80 Hoojim or 960 Blinks.`

```
#Code here
```

Problem–24 Write a function called enBlinkHoojaBung

Write a function called enBlinkHoojaBung that takes a number of Blinks and outputs its equivalent in Blink, Hooja and Bung, using the smallest number of coins. Coins in our imaginary land are made of a very heavy metal, so this is important.

If the function is called with the value `506`, the output should be: `506 Blinks is worth 2 Bung, 1 Hoojim and 6 Blinks`.

```
#Code here
```

Problem–25 Rewrite enBlinkHoojaBung

Rewrite enBlinkHoojaBung so that it returns a tuple of Blinks, Hooja and Bung values instead of writing to the screen. The last example would return `( 2 , 1 , 6 )`.

```
#code here
```

Problem 26–28 Convert the following iterative functions into recursive functions:

P-26:

```
def sum_even ( n : int) :
    total = 0
    for i in range ( 2 , n + 1 , 2 ) :
        total += i
    return total
```

P-27:

```
def min ( l:list ) :
    m = 0
    for i in l :
        if i<m :
            m = i
    return m
```

P-28:

```
def prod ( l : list ) :
    product , i = 1 , 0
    while i < len ( l ) :
        product *= l[i]
        i += 1
    return product
```

```
#Code here for Problem 26-28
```

Problem 29-31 Convert the following iterative functions into recursive functions:

P-29:

```
def sum_odd ( n , total ) :
    if n == 1 :
        return total
    elif n % 2 == 0 :
        return sum_odd ( n - 1 , total )
    else :
        return sum_odd ( n - 2 , total + n )
```

P-30:

```
def max ( l , n ) :
    if l == [ ] :
        return n
    elif l[0] > n :
        return max ( l[1:] , l[0] )
    else :
        return max ( l[1:] , n )
```

P-31:

```
def mylen ( l , n ) :
    if l == [] :
        return n
    else :
        return mylen ( l[1:] , n +1 )
```

```
#Code here for Problem 29-31
```

`Problem 32-35

P-32:

> Use map and lambda to turn a list of integers from `1 to 100` into a list of even numbers from `2 to 200`.

P-33:

> Use filter to generate a list of odd numbers from 0 to 100

P-34:

> Use a list comprehension to generate a list of odd numbers from 0 to 100.

P-35:

> Write a generator function (using the yield keyword) that generates factorial numbers.

```
#Code here for P-32 to P-35
```

Problem-36: Ackermann's Function

Ackermann's Function is defined as:

```
          n+1 if m = 0
  A(m,n) = A(m-1,1) if m > 0 and n = 0
          A(m-1,A(m,n-1)) if m > 0 and n > 0
```

i. Write a recursive Python function to implement Ackermann's Function.

ii. How many recursive calls will be required to evaluate A(2,3)?

```
#Code here
def ackermann ( m , n ) :
    if m == 0 : return n + 1
    elif m > 0 and n == 0 : return ackermann ( m - 1 , 1 )
    else : return ackermann ( m - 1 , ackermann ( m , n - 1 ) )

ackermann(2, 3)
```

```
9
```

Problem–37: Palindrome String

Write a recursive function implementation of `isPalindrome` to test whether or not a string is a palindrome.

`a man, a plan, a canal, panama!` is a palindrome – so we do not make spaces and special characters(! here) significant.

```
#Write your code here
```

Problem–38 Create a module for playing Tic-Tac-Toe.

Hint: You may want to consider the following functions:

1. print_board ( ) – from the programming exercise above, except that you will want to use a global board variable, rather than a function argument.
2. has_won ( ) – check to see whether either player has won. This function should return the string 'o' or 'x' if either the o or x player has won and '#' if neither player has won. A player wins in Tic-Tac-Toe if they have three of their counters in a row, a column or a diagonal.
3. place_counter (sq , counter) – place the counter on a particular square of the board. The first argument should be a number between 0 and 8 and the second argument should be either 'o' or 'x'. You should consider the squares on the board to be numbered as in the diagram below.

   ```
   0 | 1 | 2
   ----------
   3 | 4 | 5
   ----------
   6 | 7 | 8
   ```

   Using a numbering such as this one makes the answer to this challenge simpler!
4. next_play ( ) – This function should ask the user for the next move they want to play. You should make sure that the user knows whether x or o is currently playing. You can assume that the user will enter an integer value. You should still check that the integer the player has provided is between 0 and 8 inclusive.

```
#Think yourself for answers
```

Problem 39: Die class

Create a class to represent a single die that can have any positive integer number of sides. This kind of die might be used when playing role-playing games (RPGs).