**Python Basics Fundamental from commence with Assignments 1**

Contents:

* Introduction to
* Conditionals and Loops
* Pattern 1
* Pattern 2
* More on Loops

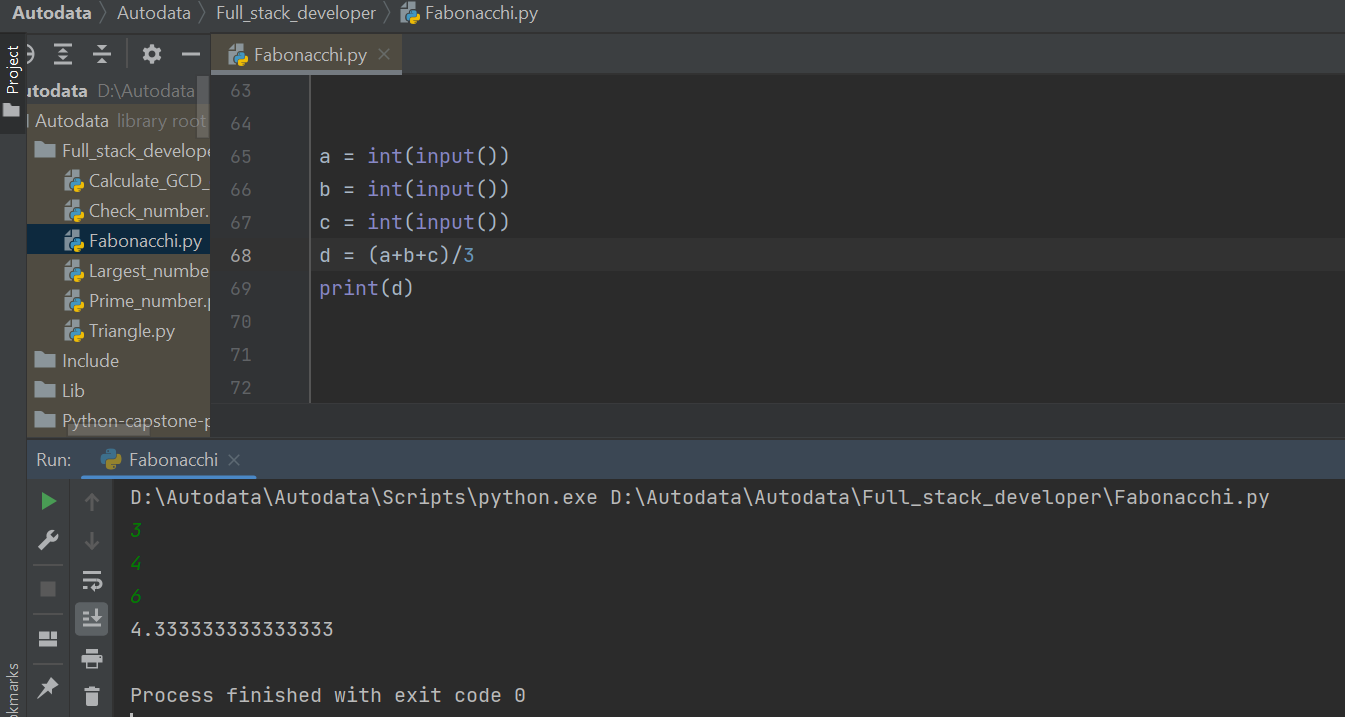
**INTRODUCTION TO PYTHON**

### **Question 1**

Write a program to input marks of three tests of a student (all integers). Then calculate and print the average of all test marks.

Sample Input 1 :  
3  
4  
6

Sample Output 1 :  
4.333333333333333



### **Question 2**

You are given two integers: X and N. You have to calculate X raised to power N and print it.

**Input format:** The first line of input contains an integer X (1 <= X <= 100) The second line of input contains an integer N (1 <= N <= 10)

**Constraints:** Time Limit: 1 second

**Output format:** The first and only line of output contains the result.

Sample Input:  
10  
4

Sample Output:  
10000

### 

### **Question 3**

**Arithmetic Progression**

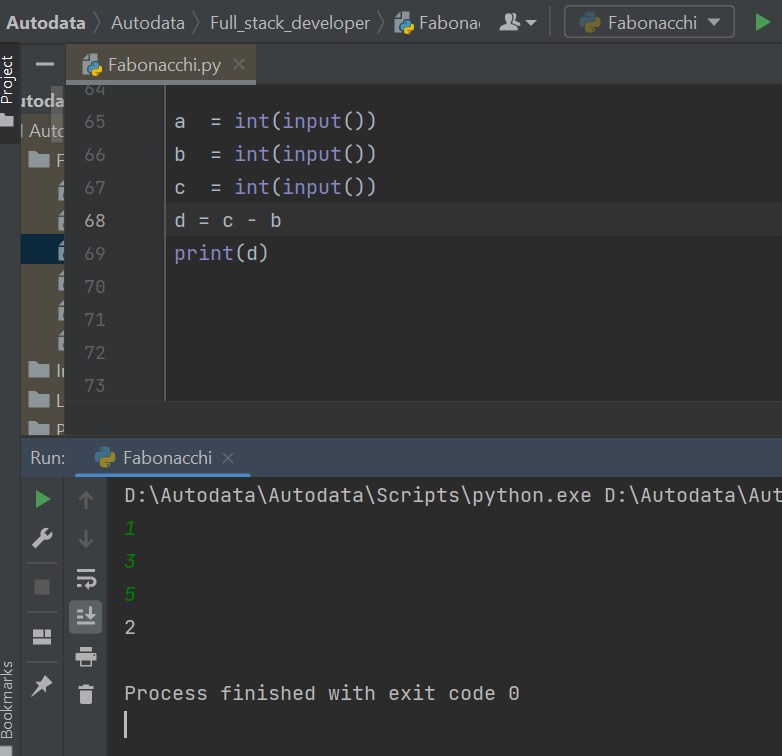
You are given first three entries of an arithmetic progression. You have to calculate the common difference and print it.

**Input format:** The first line of input contains an integer a (1 <= a <= 100)  
The second line of input contains an integer b (1 <= b <= 100)  
The third line of input contains an integer c (1 <= c <= 100)

**Constraints:** Time Limit: 1 second

**Output format:** The first and only line of output contains the result.

Sample Input:  
1  
3  
5  
Sample Output:  
2



### **Question 4**

**Rectangular Area**

You are given a rectangle in a plane. The corner coordinates of this rectangle is provided to you. You have to print the amount of area of the plane covered by this rectangle. The end coordinates are provided as four integral values: x1, y1, x2, y2. It is given that x1 < x2 and y1 < y2.

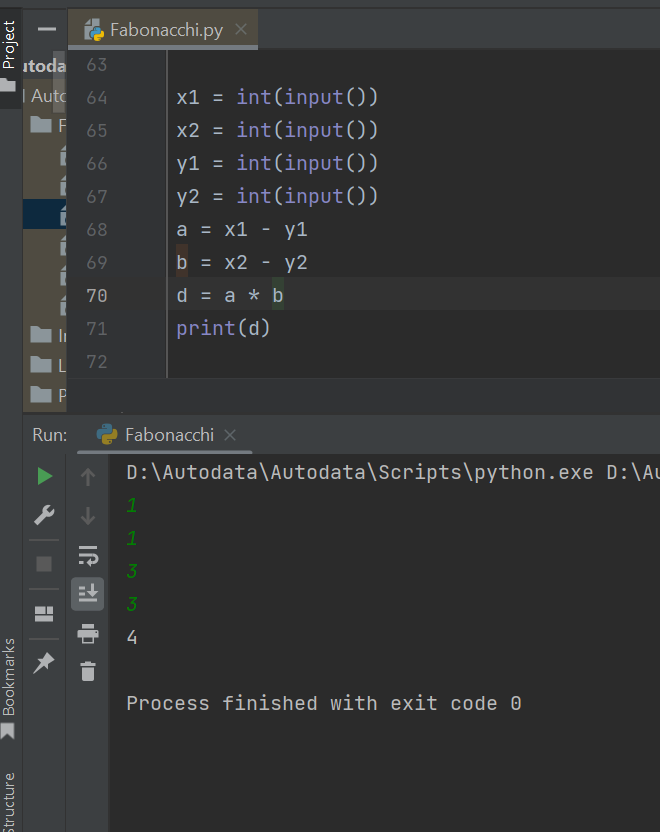
**Input format:**  
The first line of input contains an integer x1 (1 <= x1 <= 10)  
The second line of input contains an integer y1 (1 <= y1 <= 10)  
The third line of input contains an integer x2 (1 <= x2 <= 10)  
The fourth line of input contains an integer y2 (1 <= y2 <= 10)

**Constraints:** Time Limit: 1 second

**Output format:** The first and only line of output contains the result.

Sample Input:  
1  
1  
3  
3

Sample Output:  
4



## **Conditionals and Loops**

### **Question 1**

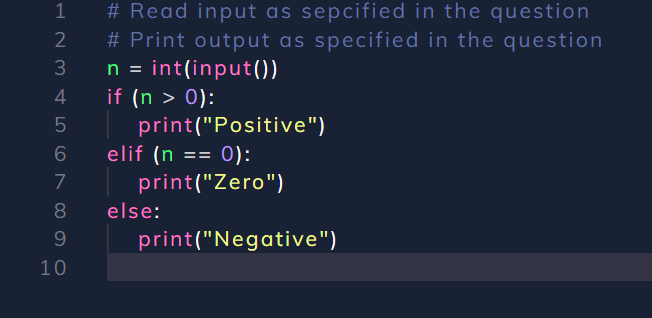
#### **Check Number**

Given an integer n, find if n is positive, negative or 0. If n is positive, print "Positive" If n is negative, print "Negative" And if n is equal to 0, print "Zero".

Input Format:  
Integer n  
Output Format:  
"Positive" or "Negative" or "Zero" (without double quotes)

**Constraints:** 1 <= n <= 100

Sample Input 1:  
10  
Sample Output 1 :  
Positive  
Sample Input 2 :  
-10  
Sample Output 2 :  
Negative



### Question 2

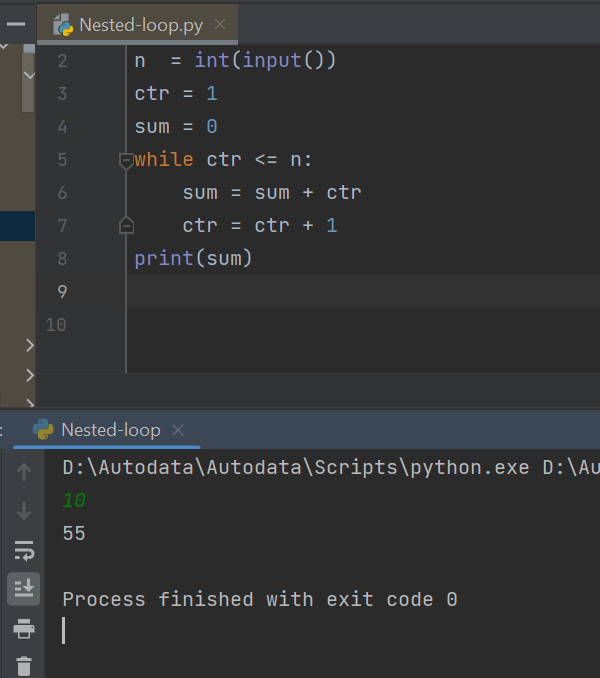
#### **Sum of n numbers**

Given an integer n, find and print the sum of numbers from 1 to n.  
Note : Use while loop only.

Input Format :  
Integer n  
Output Format :  
Sum

Constraints :  
1 <= n <= 100

Sample Input :  
10  
Sample Output :  
55



In [ ]:

### **Question 3**

#### Sum of Even Numbers

Given a number N, print sum of all even numbers from 1 to N.

Input Format :  
Integer N  
Output Format :  
Required Sum

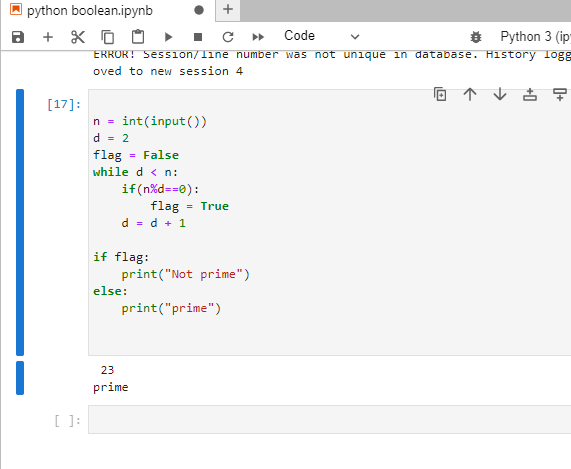
Sample Input 1 :  
6  
Sample Output 1 :  
12

### 

### **Question 4**

#### **Prime Number**

Check to see weather a given number n is prime or not.



### **Question 5**

Check all prime numbers from 2 to n

A screenshot of a computer

Description automatically generated with medium confidence

**Question 6**

#### **Fahrenheit to Celsius**

Given three values - Start Fahrenheit Value (S), End Fahrenheit value (E) and Step Size (W), you need to convert all Fahrenheit values from Start to End at the gap of W, into their corresponding Celsius values and print the table. \

Input Format :  
3 integers - S, E and W respectively

Output Format :  
Fahrenheit to Celsius conversion table. One line for every Fahrenheit and corresponding Celsius value. On Fahrenheit value and its corresponding Celsius value should be separate by tab ("\t")

Constraints :  
0 <= S <= 80  
S <= E <= 900  
0 <= W <= 40

Sample Input 1:  
0  
100  
20

Sample Output 1:  
0 -17  
20 -6  
40 4  
60 15  
80 26  
100 37

Sample Input 2:  
20  
119  
13

Sample Output 2:  
20 -6  
33 0  
46 7  
59 15  
72 22  
85 29  
98 36  
111 43

Explanation For Input 2:  
We need need to start calculating the Celsius values for each of the Fahrenheit Value which starts from 20. So starting from 20 which is the given Fahrenheit start value, we need to compute its corresponding Celsius value which computes to -6. We print this information as a tab space"\t" on each line for each step of 13 we take to get the next value of Fahrenheit and extend this idea till we reach the end that is till 119 in this case. You may or may not exactly land on the end value depending on the steps you are taking.

### 

Output:

0 -17

20 -6

40 4

60 15

80 26

100 37

### **Questoin 7**

#### **Calculator**

Write a program that performs the tasks of a simple calculator. The program should first take an integer as input and then based on that integer perform the task as given below.

1. If the input is 1, then 2 integers are taken from the user and their sum is printed.
2. If the input is 2, then 2 integers are taken from the user and their difference(1st number - 2nd number) is printed.
3. If the input is 3, then 2 integers are taken from the user and their product is printed.
4. If the input is 4, then 2 integers are taken from the user and the quotient obtained (on dividing 1st number by 2nd number) is printed.
5. If the input is 5, then 2 integers are taken from the user and their remainder(1st number mod 2nd number) is printed.
6. If the input is 6, then the program exits.
7. For any other input, then print "Invalid Operation".

Note: Each answer in next line.

Input format: Take integers as input, in accordance to the description of the question.

Constraints: Time Limit: 1 second

Output format: The output lines must be as prescribed in the description of the question.

Sample Input: 3  
1  
2  
4  
4  
2  
1  
3  
2  
7  
6

Sample Output: 2  
2  
5

Invalid Operation Explanation of the sample input The first number given is 3, so that means two more numbers will be given and we'll have to multiply them and show the result. The two numbers are 1 and 2. Their product is 2, so 2 is displayed first in the output. Similarly, all the numbers are processed in groups of three. The first number tells the operation and the next two numbers tell on which numbers the operation is done. This applies to numbers from 1 to 5. If the input is 6 (like it is at the end), two more numbers will NOT be provided, you simply have to exit the program. Also, if the input is any number except 1 to 6 (like 7 which is at the second last), then you simply have to print "Invalid Operation"

while (1):  
 i = int(input())  
 if(i == 1):  
 a = int(input())  
 b = int(input())  
 c = a+b  
 print(c)  
 elif(i == 2):  
 a = int(input())  
 b = int(input())  
 c = a-b  
 print(c)  
 elif(i == 3):  
 a = int(input())  
 b = int(input())  
 c = a\*b  
 print(c)  
 elif(i == 4):  
 a = int(input())  
 b = int(input())  
 c = a/b  
 print(int(c))  
 elif(i == 5):  
 a = int(input())  
 b = int(input())  
 c = a%b  
 print(int(c))  
 elif(i == 6):   
 break   
 else:  
 print("Invalid Operation")

### **Question 8**

**Reverse of a number**

Write a program to generate the reverse of a given number N. Print the corresponding reverse number.  
Note : If a number has trailing zeros, then its reverse will not include them. For e.g., reverse of 10400 will be 401 instead of 00401.

Input format :  
Integer N

Output format :  
Corresponding reverse number

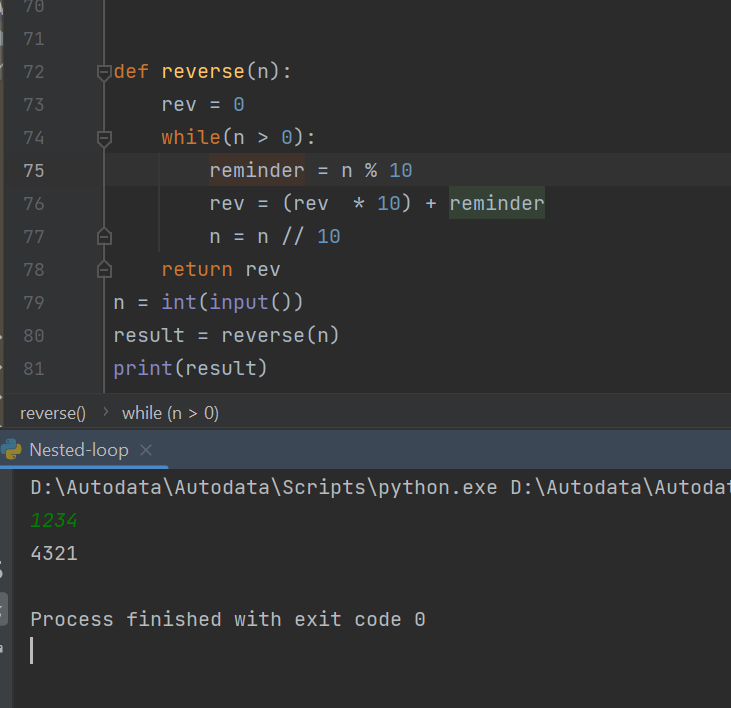
Constraints:  
0 <= N < 10^8

Sample Input 1 :  
1234

Sample Output 1 :  
4321

Sample Input 2 :  
1980

Sample Output 2 :  
891



### **Question 9**

#### **Palindrome number**

Write a program to determine if given number is palindrome or not. Print true if it is palindrome, false otherwise.  
Palindrome are the numbers for which reverse is exactly same as the original one. For eg. 121

Sample Input 1 :  
121  
Sample Output 1 :  
true

Sample Input 2 :  
1032  
Sample Output 2 :  
false

### 

### **Question 10**

#### **Sum of even & odd**

Write a program to input an integer N and print the sum of all its even digits and sum of all its odd digits separately.  
Digits mean numbers, not the places! That is, if the given integer is "13245", even digits are 2 & 4 and odd digits are 1, 3 & 5.

Input format : Integer N

Output format : Sum\_of\_Even\_Digits Sum\_of\_Odd\_Digits (Print first even sum and then odd sum separated by space)

Constraints 0 <= N <= 10^8

Sample Input 1:  
1234  
Sample Output 1:  
6 4

Sample Input 2:  
552245  
Sample Output 2:  
8 15

Explanation for Input 2:  
For the given input, the even digits are 2, 2 and 4 and if we take the sum of these digits it will come out to be 8(2 + 2 + 4) and similarly, if we look at the odd digits, they are, 5, 5 and 5 which makes a sum of 15(5 + 5 + 5). Hence the answer would be, 8(evenSum) 15(oddSum)

### 

### **Question 11**

#### **Nth Fibonacci Number**

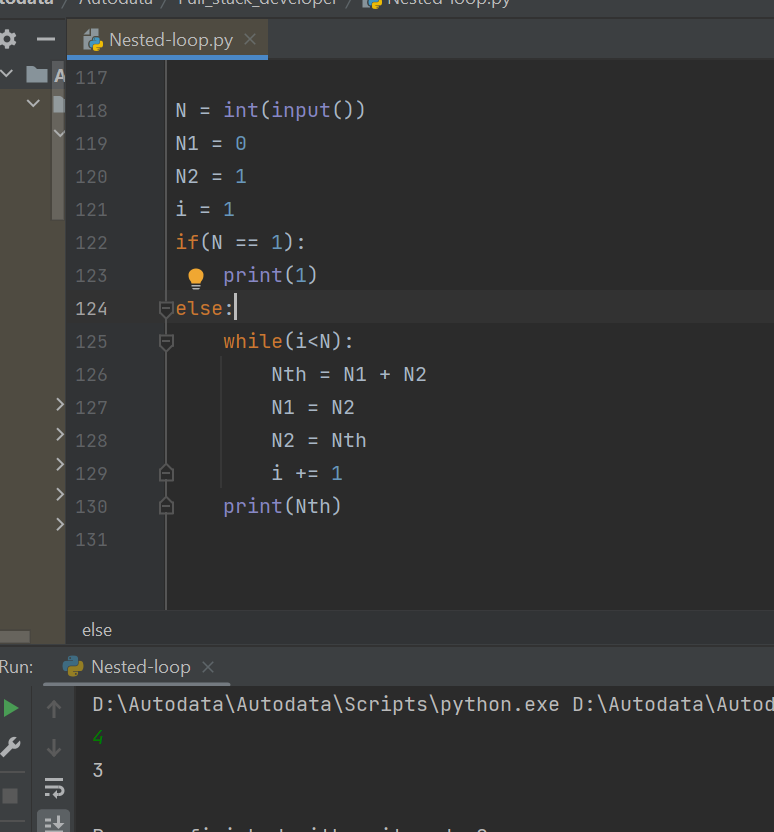
Nth term of Fibonacci series F(n), where F(n) is a function, is calculated using the following formula -  
F(n) = F(n-1) + F(n-2),  
Where, F(1) = F(2) = 1  
Provided N you have to find out the Nth Fibonacci Number.

Input Format :  
The first line of each test case contains a real number ‘N’.  
Output Format :  
For each test case, return its equivalent Fibonacci number.  
Constraints:  
1 <= N <= 10000  
Where ‘N’ represents the number for which we have to find its equivalent Fibonacci number.

Time Limit: 1 second  
Sample Input 1:  
6  
Sample Output 1:  
8

Explanation of Sample Input 1:  
Now the number is ‘6’ so we have to find the “6th” Fibonacci number  
So by using the property of the Fibonacci series i.e

[ 1, 1, 2, 3, 5, 8]  
So the “6th” element is “8” hence we get the output.



A screenshot of a computer

Description automatically generated

## **Patterns 1**

### **Example**

'\*\*\*\*  
'\*\*\*\*  
'\*\*\*\*  
'\*\*\*\*Text

Description automatically generated

### **Questoin 1**

**Code : Square Pattern**

Print the following pattern for the given N number of rows. Pattern for N = 4  
4444  
4444  
4444  
4444

Input format :  
Integer N (Total no. of rows)

Output format : Pattern in N lines

Constraints  
0 <= N <= 50

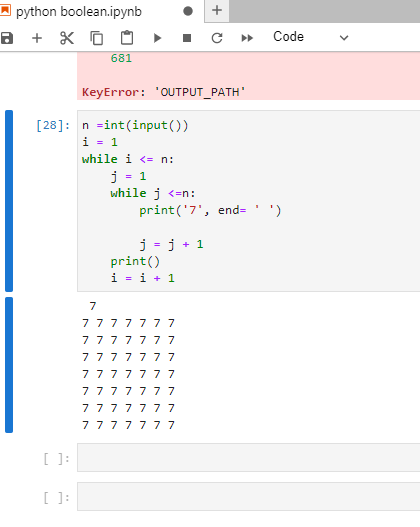
Sample Input 1:  
7  
Sample Output 1:  
7777777  
7777777  
7777777  
7777777  
7777777  
7777777  
7777777

Sample Input 1:  
6

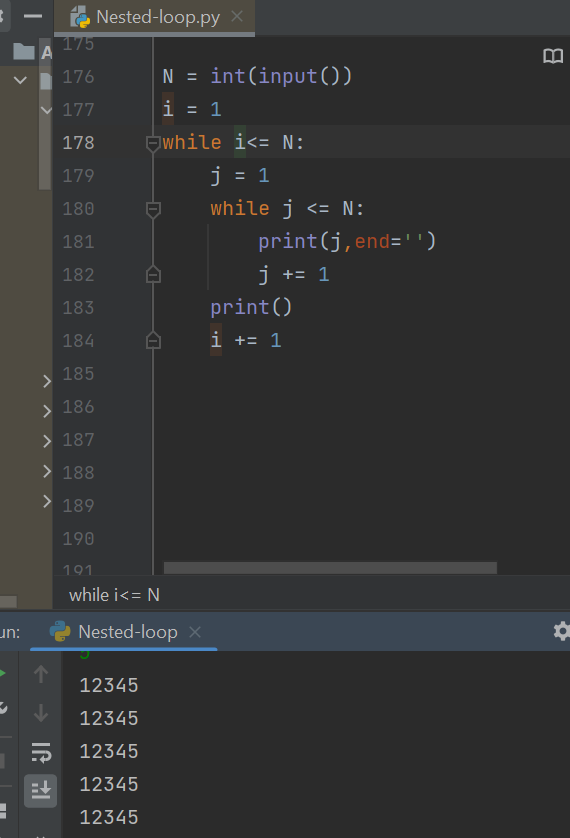
Sample Output 1:  
666666  
666666  
666666  
666666  
666666  
666666Text

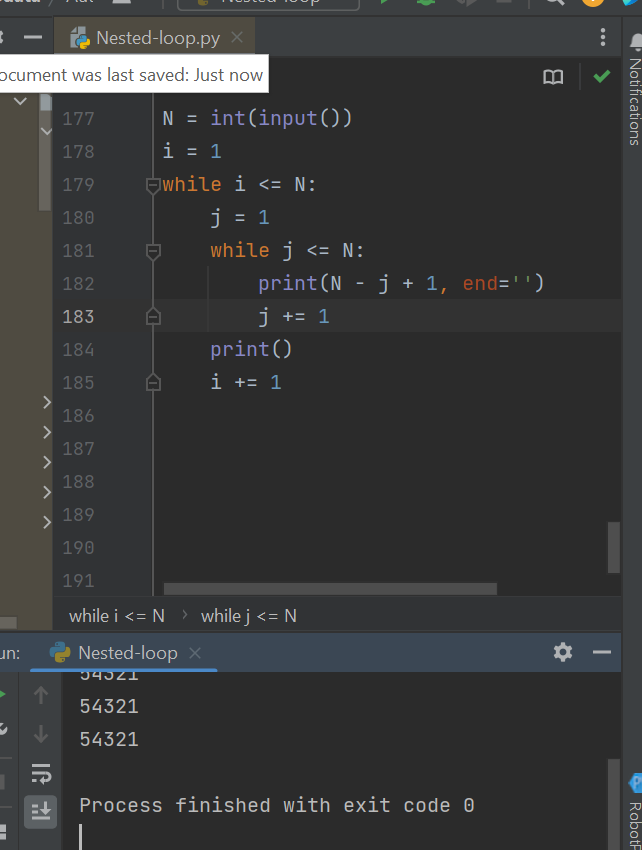
Description automatically generated

### **Square patterns**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### **Triangle Pattern**

A screenshot of a computer

Description automatically generated with medium confidence1

12

123

1234

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A screenshot of a computer

Description automatically generated with medium confidence

1

23

345

4567

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*# In previos question we reset p now we have it globally*

A screenshot of a computer

Description automatically generated with medium confidence

1

23

456

78910

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### **Question 2**

**Code : Triangular Star Pattern**

Print the following pattern for the given N number of rows.  
Pattern for N = 4  
'\*  
'\*\*  
'\*\*\*  
'\*\*\*\*

Note : There are no spaces between the stars (\*).

Input format : Integer N (Total no. of rows)

Output format : Pattern in N lines

Constraints 0 <= N <= 50

Sample Input 1:  
5  
Sample Output 1:  
'\*  
'\*\*  
'\*\*\*  
'\*\*\*\*  
'\*\*\*\*\*  
Sample Input 2:  
6  
Sample Output 2:  
'\*  
'\*\*  
'\*\*\*  
'\*\*\*\*  
'\*\*\*\*\*  
'\*\*\*\*\*\*

A screenshot of a computer

Description automatically generated with medium confidence

\*

\*\*

\*\*\*

### **Question 3**

**Code : Triangle Number Pattern**

Print the following pattern for the given N number of rows.  
Pattern for N = 4

1  
22  
333  
4444

Input format :  
Integer N (Total no. of rows)

Output format : Pattern in N lines

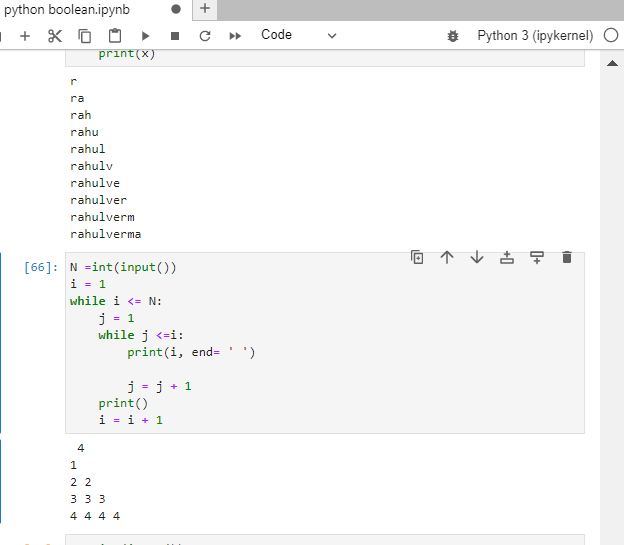
Constraints 0 <= N <= 50

Sample Input 1: 5  
Sample Output 1:

1  
22  
333  
4444  
55555

Sample Input 2: 6  
Sample Output 2:

1  
22  
333  
4444  
55555  
666666

1

22

333

4444

**Code : Reverse Number Pattern**

Print the following pattern for the given N number of rows. Pattern for N = 4

1  
21  
321

4321

Input format :  
Integer N (Total no. of rows)

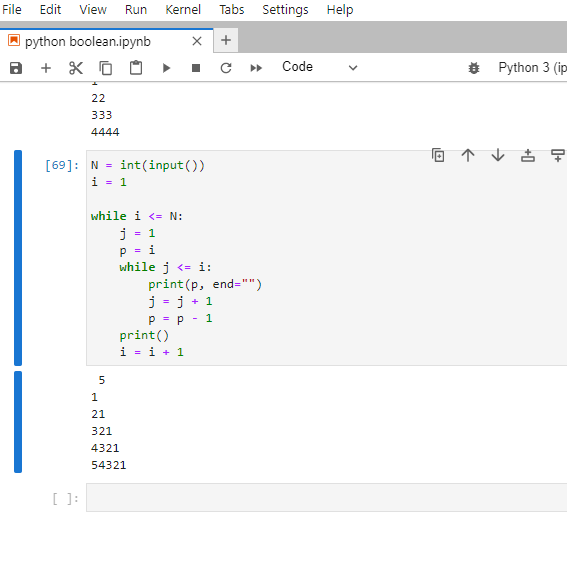
Output format : Pattern in N lines

Constraints  
0 <= N <= 50

Sample Input 1:  
5  
Sample Output 1:

1  
21  
321  
4321  
54321  
Sample Input 2:  
6  
Sample Output 2:  
1  
21  
321  
4321  
54321  
654321

### **Question 4**



1

21

321

4321

54321

654321

### **Printing kth term**

k = int (input())  
# 'A' +k - 1  
x = ord ('A')  
assci\_target = x + k - 1  
target\_char = chr(assci\_target)  
print (target\_char)

ABCD  
ABCD  
ABCD  
ABCD

n = int(input())  
i = 1  
while (i <= n):  
 j = 1  
 while( j <= n):  
 charP = chr(ord('A') + j - 1) # to get the character.  
 print(charP,end='')  
 j += 1  
 print()  
 i += 1

ABCD

ABCD

ABCD

ABCD

ABCD  
BCDE  
CDEF  
DEFG

n = int(input())  
i = 1  
while (i <= n):  
 j = 1  
 start\_char = chr(ord('A') + i - 1)  
 while( j <= n):  
 charP = chr(ord(start\_char) + j - 1) # to get the character.  
 print(charP,end='')  
 j += 1  
 print()  
 i += 1

ABCD

BCDE

CDEF

DEFG

### **Question 5**

#### **Character Pattern**

Print the following pattern for the given N number of rows.  
Pattern for N = 4

A  
BC  
CDE  
DEFG

Input format :  
Integer N (Total no. of rows)

Output format : Pattern in N lines

Constraints 0 <= N <= 13

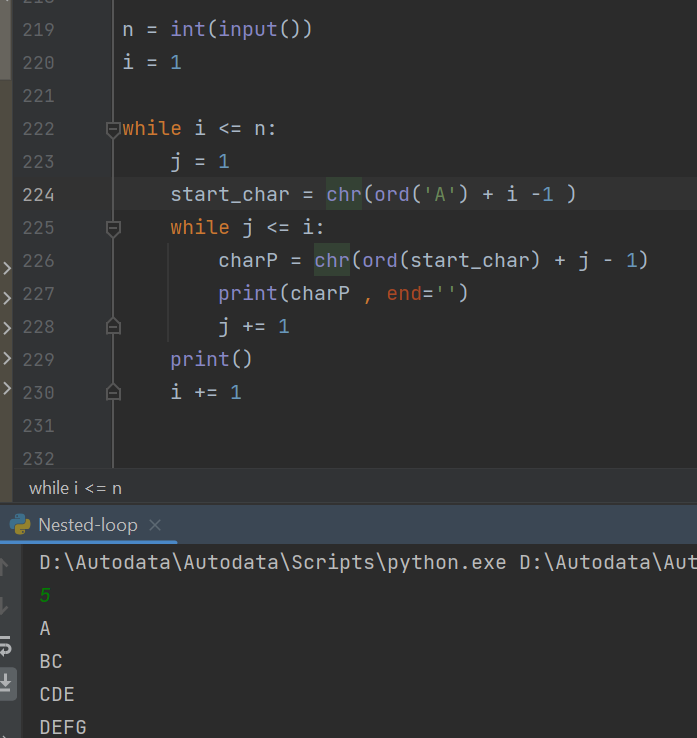
Sample Input 1: 5  
Sample Output 1:

A  
BC  
CDE  
DEFG  
EFGHI

Sample Input 2:  
6

Sample Output 2:

A  
BC  
CDE  
DEFG  
EFGHI  
FGHIJK



A

BC

CDE

DEFG

***# Same solution can also be done by no chr in earlier stages***

n = int(input())   
currRow = 1   
while currRow <= n:  
 currCol = 1   
 ch = ord('A') + currRow - 1   
 while currCol <= currRow:   
 print(chr(ch + currCol - 1), end = "")  
 currCol += 1   
 print()   
 currRow += 1

A

BC

CDE

DEFG

### **Question 6**

#### Pattern for N = 5

E  
DE  
CDE  
BCDE  
ABCDE

n = int(input())  
i = 1  
while i <= n:  
 j = 1  
 start\_char = chr(ord('A') + n - 1)  
  
 while j <= i:  
 charP = chr(ord(start\_char) - i + j)  
 print(charP, end='')  
 j += 1  
 print()  
 i += 1

E

DE

CDE

BCDE

ABCDE

n = int(input())  
i = 1  
while i <= n:  
 j = 1  
 start\_char = chr(ord('A') + n - 1)  
  
 while j <= i:  
 charP = chr(ord(start\_char) - i + j)  
 print(charP, end='')  
 j += 1  
 print()  
 i += 1

E

DE

CDE

BCDE

ABCDE

### **Question 7**

#### **Number Pattern 1**

Print the following pattern for the given N number of rows.  
Pattern for N = 4  
1  
11  
111  
1111

N = int(input())  
i = 1  
while i<=N:  
 j = 1  
 while j<=i:  
 print('1', end='')  
 j = j + 1  
 print()  
 i = i + 1

1

11

111

1111

### **Question 8**

**Number Pattern 2**

Print the following pattern for the given N number of rows.  
Pattern for N = 4

1  
11  
121  
1221

Input format :  
Integer N (Total no. of rows)

Output format : Pattern in N lines

Sample Input :  
5

Sample Output :  
1  
11  
121  
1221  
12221

n = int(input())  
i = 1  
print("1")  
while (i <= n - 1):  
 j = 1  
 print("1", end='')  
 while (j <= i - 1):  
 print("2", end='')  
 j += 1  
 print("1", end='')  
 print()  
 i += 1

1

11

121

1221

12221

122221

N = int(input())  
i = 0  
while (i < N):  
 j = 0  
 while (j < i + 1):  
 if(j == 0 or j == i):  
 print(1, end='')  
 else:  
 print(2, end = '')  
 j = j + 1  
 print()  
 i = i + 1

1

11

121

1221

### **Question 9**

**Number Pattern 3**

Print the following pattern for the given N number of rows.  
Pattern for N = 4  
1  
11  
202  
3003

Input format :  
Integer N (Total no. of rows)

Contraints:  
1 <= n <= 50

Output format :  
Pattern in N lines

Sample Input :  
5

Sample Output :  
1  
11  
202  
3003  
40004

N = int(input())  
i = 1  
print("1")  
while (i <=N-1):  
 j = 1  
 print(i, end = '')  
 while (j <= i - 1):  
 print("0", end='')  
 j = j + 1  
 print(i, end = '')  
  
 print()  
 i = i + 1

1

11

202

3003

40004

N = int(input())  
i = 1  
print("1")  
while (i <=N-1):  
 j = 1  
 print(i, end = '')  
 while (j <= i - 1):  
 print("2", end='')  
 j = j + 1  
 print(i, end = '')  
  
 print()  
 i = i + 1

1

11

202

3003

40004

### **Question 10**

**Number Pattern**

Print the following pattern for the given N number of rows.  
Pattern for N = 4

1234  
123  
12  
1

Input format :  
Integer N (Total no. of rows)

Output format :  
Pattern in N lines

Sample Input :  
5

Sample Output :

12345  
1234  
123  
12  
1

N = int(input())  
i = 1  
  
while i <= N:  
 j = 1  
  
 while j <= (N - i + 1):  
 print(j, end="")  
 j = j + 1  
  
 print()  
 i = i + 1

1234

123

12

1

### **Questoin 11**

**Alpha Pattern**

Print the following pattern for the given N number of rows.  
Pattern for N = 3

A  
BB  
CCC

Input format :  
Integer N (Total no. of rows)

Output format :  
Pattern in N lines

Constraints  
0 <= N <= 26

Sample Input 1:  
7

Sample Output 1:

A  
BB  
CCC  
DDDD  
EEEEE  
FFFFFF  
GGGGGGG

Sample Input 2:  
6

Sample Output 2:

A  
BB  
CCC  
DDDD  
EEEEE  
FFFFFF

n = int(input())  
i = 1  
while i <= n:  
 j = 1  
 start\_char = chr(ord('A') + i - 1)  
  
 while j <= i:  
 print(start\_char, end='')  
 j += 1  
 print()  
 i += 1

A

BB

CCC

DDDD

## Patterns 2

### Question1

Code : Inverted Number Pattern

Print the following pattern for the given N number of rows.

Pattern for N = 4

4444  
333  
22  
1

## Print the required output in given format

N = int(input())

i = 1

while i <= N:

    j = 1

    while j <= N - i + 1:

        print(N-i+1, end="")

        j = j + 1

    print()

    i = i + 1

4444

333

22

1

### **Reversed Triangle**

In [ ]:

'''

\*

\*\*

\*\*\*

\*\*\*\*

'''

N = int(input())  
i = 1  
  
while i <= N:  
 spaces = 1  
  
 while spaces <= N - i:  
 print(' ', end="")  
 spaces = spaces + 1  
 stars = 1  
 while stars <= i:  
 print('\*', end ="")  
 stars = stars + 1  
 print()  
 i = i + 1

\*

\*\*

\*\*\*

\*\*\*\*

### **Question 2**

**Code : Mirror Number Pattern**

Print the following pattern for the given N number of rows. Pattern for N = 4

...1  
..22  
.333  
4444

The dots represent spaces.

n = int(input())  
i = 1  
while (i <= n):  
 spaces = 1  
 while spaces <= (n - i):  
 print(' ', end='')  
 spaces += 1  
  
 stars = 1  
 while stars <= i:  
 print(i, end='')  
 stars += 1  
 print()  
 i += 1

1

22

333

4444

### **Isoscelese Pattern Problems**

*# To bulid this first learn to build this*

'''

1

12

123

1234

'''

*# increasing sequence*

N = int(input())  
i = 1  
  
while i <= N:  
 spaces = 1  
  
 while spaces <= N - i:  
 print(' ', end="")  
 spaces = spaces + 1  
 stars = 1  
 while stars <= i:  
 print(stars, end ="")  
 stars = stars + 1  
 print()  
 i = i + 1

1

12

123

1234

***# decreasing sequence***

n = int(input())  
i = 2  
while (i <= n):  
 p = i - 1  
 while p >= 1:  
 print(p, end='')  
 p -= 1  
 print()  
 i += 1

1

21

321

***# Now for the actual Isoscelese Pattern***

n = int(input())  
i = 1  
while (i <= n):  
 # spaces  
 spaces = 1  
 while spaces <= (n - i):  
 print(' ', end='')  
 spaces += 1  
  
 # increasing sequence  
 j = 1  
 p = 1  
 while j <= i:  
 print(p, end='')  
 j += 1  
 p += 1  
  
 # decreasing sequence  
 p = i - 1  
 while p >= 1:  
 print(p, end='')  
 p -= 1  
  
 print()  
 i += 1

1

121

12321

1234321

### **Queston 3**

'''

Print the following pattern

Pattern for N = 4

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

The dots represent spaces.

'''

'\nPrint the following pattern\nPattern for N = 4\n \*\n \*\*\*\n \*\*\*\*\*\n\*\*\*\*\*\*\*\nThe dots represent spaces.\n'

n = int(input())  
i = 1  
while (i <= n):  
 # spaces  
 spaces = 1  
 while spaces <= (n - i):  
 print(' ', end='')  
 spaces += 1  
  
 # increasing sequence  
 j = 1  
 p = 1  
 while j <= i:  
 print('\*', end='')  
 j += 1  
 p += 1  
  
 # decreasing sequence  
 p = i - 1  
 while p >= 1:  
 print('\*', end='')  
 p -= 1  
  
 print()  
 i += 1

\*

\*\*\*

\*\*\*\*\*

### **Question 4**

'''

Editorial for the problem is present

Code : Triangle of Numbers

Print the following pattern for the given number of rows.

Pattern for N = 4

Sample Input 1:

5

Sample Output 1:

1

232

34543

4567654

567898765

'''

*# important see how p is used in the code*

n = int(input())  
i = 1  
while (i <= n):  
 # spaces  
 spaces = 1  
 while (spaces <= n - i):  
 print(' ', end='')  
 spaces += 1  
  
 # increasing sequence  
 j = 1  
 p = i  
 while (j <= i):  
 print(p, end='')  
 j += 1  
 p += 1

1

232

34543

4567654

n = int(input())  
currRow = 1  
while currRow <= n:  
 spaces = 1  
 while spaces <= (n - currRow):  
 print(" ", end="")  
 spaces += 1  
  
 currCol = 1  
 valToPrint = currRow  
 while currCol <= currRow:  
 print(valToPrint, end="")  
 valToPrint += 1  
 currCol += 1  
  
 currCol = 1  
 valToPrint = 2 \* currRow - 2  
 while currCol <= currRow - 1:  
 print(valToPrint, end="")  
 valToPrint -= 1  
 currCol += 1  
 print()  
 currRow += 1

1

232

34543

4567654

567898765

### **Question 5**

'''

Pyramid Number Pattern

Print the following pattern for the given number of rows.

Pattern for N = 4

1

212

32123

4321234

Input format : N (Total no. of rows)

Output format : Pattern in N lines

'''

n = int(input())  
i = 1  
while (i <= n):  
 spaces = 1  
 while (spaces <= n - i):  
 print(" ", end='')  
 spaces += 1  
  
 j = 1  
 p = i  
 while (j <= i):  
 print(p, end='')  
 p -= 1  
 j += 1  
  
 k = 1  
 while (k <= i - 1):  
 print(k + 1, end='')  
 k += 1  
 print()  
 i += 1

1

212

32123

4321234

### **Question 6**

'''

Code : Diamond of stars

Send Feedback

Print the following pattern for the given number of rows.

Note: N is always odd.

Pattern for N = 5

..\*..

.\*\*\*.

\*\*\*\*\*

.\*\*\*.

..\*..

The dots represent spaces.

'''

n = int(input())  
# top half  
n1 = (n + 1) // 2  
  
i = 1  
while i <= n1:  
 j = 1  
 # top spaces  
 while j <= n1 - i:  
 print(' ', end='')  
 j += 1  
  
 # top stars  
 k = 1  
 while k <= 2 \* i - 1:  
 print('\*', end='')  
 k += 1  
 i += 1  
 print()  
  
n2 = n // 2  
y = 1  
while y <= n2:  
  
 # bottom spaces  
 l = 1  
 while l <= y:  
 print(' ', end='')  
 l += 1  
  
 # bottom star  
 m = 1  
 while m <= n1 + n2 - 2 \* y:  
 print('\*', end='')  
 m += 1  
  
 y += 1  
 print()

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

n = int(input())  
firstHalf = (n + 1) // 2  
secondHalf = n // 2  
  
# First Half  
currRow = 1  
while currRow <= firstHalf:  
 spaces = 1  
 while spaces <= (firstHalf - currRow):  
 print(" ", end="")  
 spaces += 1  
  
 currCol = 1  
 while currCol <= (2 \* currRow) - 1:  
 print("\*", end="")  
 currCol += 1  
  
 print()  
 currRow += 1  
# Second Half  
currRow = secondHalf  
while currRow >= 1:  
 spaces = 1  
 while spaces <= (secondHalf - currRow + 1):  
 print(" ", end="")  
 spaces += 1  
  
 currCol = 1  
 while currCol <= (2 \* currRow) - 1:  
 print("\*", end="")  
 currCol += 1  
  
 print()  
 currRow -= 1

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

### **Question 7**

'''

Arrow pattern

Print the following pattern for the given number of rows.

Assume N is always odd.

Note : There is space after every star.

Pattern for N = 7

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

'''

n = int(input())  
n1 = (n + 1) // 2  
n2 = n // 2  
  
i = 1  
while (i <= n1):  
 # top half spaces  
 j = 1  
 while (j <= i - 1):  
 print(' ', end="")  
 j += 1  
  
 # top half stars  
 k = 1  
 while (k <= i):  
 print('\* ', end="")  
 k += 1  
 print()  
 i += 1  
  
y = 1  
while (y <= n2):  
 # top half spaces  
 j = 1  
 while (j <= n2 - y):  
 print(' ', end="")  
 j += 1  
  
 # top half stars  
 k = 1  
 while (k <= n2 - y + 1):  
 print('\* ', end="")  
 k += 1  
 print()  
 y += 1

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

### **Question 8**

Number Patternn = int(input())

startValue = 1

for row in range(1, n + 1):

# Declaring a string to store our row in the pattern

ans = ""

# Appending value of each column of each row in ans

for col in range(startValue, startValue + n):

ans += str(col) + " "

# 'col' represents our current columnn in our current row

if (row == (n + 1) // 2):

if (n % 2 != 0):

startValue = n \* (n - 2) + 1

else:

startValue = n \* (n - 1) + 1

elif (row > (n + 1) // 2):

startValue = startValue - 2 \* n

else:

startValue = startValue + 2 \* n

print(ans)

Print the following pattern for n number of rows.

For eg. N = 5

1 1

12 21

123 321

1234 4321

N = int(input())  
i = 1  
while (i <= N):  
 # starting numbers  
 j = 1  
 while (j <= i):  
 print(j, end='')  
 j += 1  
  
 # spaces  
 k = 1  
 while (k <= 2 \* N - 2 \* i):  
 print(' ', end='')  
 k += 1  
  
 # ending spaces  
 l = 1  
 p = i  
 while (l <= i):  
 print(p, end='')  
 p -= 1  
 l += 1  
  
 print()  
 i += 1

1 1

12 21

123 321

1234 4321

1234554321

***# Editorial Answer***

n = int(input())  
totalspace = n \* 2 - 2  
row = 1  
while (row <= n):  
 column = 1  
 while (column <= row):  
 print(column, end='')  
 column = column + 1  
  
 space = 1  
 while (space <= totalspace):  
 print(" ", end='')  
 space = space + 1  
  
 totalspace = totalspace - 2  
  
 column = row  
 while (column > 0):  
 print(column, end='')  
 column = column - 1  
 row = row + 1  
 print()

1 1

12 21

123 321

1234 4321

1234554321

***Zeros and Stars Pattern***

Print the following pattern

Pattern for N = 4

\*000\*000\*

0\*00\*00\*0

00\*0\*0\*00

000\*\*\*000

Input Format :

N (Total no. of rows)

Output Format :

Pattern in N lines

Sample Input 1 :

3

Sample Output 1 :

\*00\*00\*

0\*0\*0\*0

00\*\*\*00

Sample Input 2 :

5

Sample Output 2 :

\*0000\*0000\*

0\*000\*000\*0

00\*00\*00\*00

000\*0\*0\*000

0000\*\*\*0000

*# TODO*

N = int(input())  
i=1  
j=1  
while i<=N:  
 j =1  
 while j<=N:  
 if i==j:  
 print("\*", end='')  
 else:  
 print("0", end='')  
 j=j+1  
 j = j-1  
 print("\*", end='')  
 while j >=1:  
 if i ==j:  
 print("\*", end='')  
 else:  
 print("0", end='')  
 j=j-1  
 print("")  
 i =i+1

\*000\*000\*

0\*00\*00\*0

00\*0\*0\*00

000\*\*\*000

## More on loops

### Generic Code for understanding

In [4]:

s **=** 'abcd'

**for** c **in** s:

print(c)

a

b

c

d

In [7]:

*# print number from 1 to n*

n **=** int(input())

**for** i **in** range(1,n **+** 1,1):

print (i)

1

2

3

4

5

6

7

8

n **=** int(input())

**for** i **in** range(n, 0,**-**1):

print (i)

6

5

4

3

2

1

### **Question 1**

print all numbers form a to b which are multiples of 3. Assume a < b

*# approach 1*

a **=** int(input())

b **=** int(input())

**for** i **in** range(a,b**+**1,1):

**if** (i **%3** == 0):

print(i, end **=** ' ')

*# This loop will run n times to execute the result*

102 105 108 111 114 117 120 123 126 129

*#approach 2*

a **=** int(input())

b **=** int(input())

**if** a **%** 3 **==**0:

s **=** a

**if** a **%** 3 **==**1:

s **=** a**+**2

**if** a **%** 3**==**2:

s **=** a**+**1

**for** i **in** range(s,b**+**1,3):

**if** (i **%3** == 0):

print(i, end **=** ' ')

*# Although we have written more code than approach one we are able to see better results as running time is less as it is not running n times.*

*# it is running n/3 times as it has a step of 3 which means it is approximately 3 times faster*

6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54

### **Question 2**

**Check if a number is prime or not**

n = int(input())  
flag = False  
for d in range(2,n,1):  
 if n%d==0:  
 flag = True  
if flag == True:  
 print(n, "Not is prime")  
else:  
 print(n, "prime")

7 prime

**Pyramid Number Pattern**

#### **Print the following pattern for the given number of rows.**

##### Pattern for N = 4

**1**

**212**

**32123**

**4321234**

Input format : N (Total no. of rows)

Output format : Pattern in N lines

##### Sample Input :

5

##### Sample Output :

1

212

32123

4321234

543212345

N = int(input())

i = 1

while i <= N:

    # spaces

    spaces = 1

    while spaces <= N - i:

        print(' ', end='')

        spaces += 1

    # increasing sequence

    j = 1

    p = i

    while j <= i:

        print(p, end='')

        j += 1

        p -= 1

    # decreasing sequence

    p = 2

    while p <= i:

        print(p, end='')

        p += 1

    print()

    i += 1

1

212

34143

4561654

### **Break Statements**

i **=** 1

**while** i **<** 10:

**if** i **==** 5:

**break**

print(i)

i **+=**1

1

2

3

4

n **=** int(input())

d **=** 2

flag **=** **False**

**while**(d **<** n):

**if** (n**%d**==0):

flag **=** **True**

**break**

d **+=** 1

**if** flag:

print('not prime')

**else**:

print('prime')

*# This is better than the code we wrote before as we are able to break out when we see even one number n is divisible bt so th while loop runs less amount of time. Therefore it is optimeised*

not prime

*# break works the same way in for as in while*

n **=** int(input())

flag **=** **False**

**for** d **in** range(2,n,1): *# we are using n as stop because we want to check form 2 to (n - 1)*

**if** n **%** d **==**0:

flag **=** **True**

**break**

**if** flag **==** **True**: *# we could also have written 'if flag' , it would have meant the same thing*

print(n,"not is prime")

**else**:

print(n, "prime")

123488 not is prime

*# if there is inner while loop and a break is inside it it will only break the inner while loop.*

n **=** int(input())

k **=** 2

**while**(k**<=**n):

d **=** 2

flag **=** **False**

**while**(d**<**k):

**if** k **%** d **==** 0:

flag **=** **True**

**break**

d **+=** 1

**if**(**not**(flag)):

print(k)

k **+=**1

2

3

5

### Else in while and for loops

*# else in while loops*

i **=**1

**while**(i **<**10):

print(i)

i **+=** 1

**else**:

print('This will be printed in once in the end')

1

2

3

4

5

6

7

8

9

This will be printed in once in the end

*# else in for loop*

**for** i **in** range(1, 10 ):

print(i)

**else**:

print('This will be printed once in the end')

1

2

3

4

5

6

7

8

9

This will be printed once in the end

*# if the code is broken by the break statement then the code will not go inside the else statement*

i **=**1

**for** i **in** range(1, 10 ):

**if**( i **==** 5):

**break**

print(i)

**else**:

print('This will be printed once in the end')

*# does not print the contents of elso statement*

1

2

3

4

*# To check if the number is prime*

n **=** int(input())

**for** d **in** range(2,n,1):

**if** n **%** d **==**0:

**break**

*# we will only go inside else if nothing could divide n*

**else**:

print(n, 'is a Prime Number')

*# it is better than the previous as we do not have to use if statements and make flags*

13 is a Prime Number

### Continue Statement

**for** i **in** range(1, 11):

**if** i **==** 5:

**continue**

print(i)

1

2

3

4

6

7

8

9

10

*# If we had to do a certain task for only odd number but also wanted to print all the numbers we could do this using continue*

**for** i **in** range(1, 20):

*#print(i)*

**if**(i**%2** ==0):

**continue**

print(i)

1

3

5

7

9

11

13

15

17

19

*# Print all even number exept which are multiple of 7*

n **=** int(input())

**for** i **in** range(2, n**+**1, 2):

**if** i **%** 7 **==** 0:

**continue**

print(i, end**=**' ')

2 4 6 8 10 12 16 18 20 22 24 26 30 32 34 36 38 40

*# continue in case of a while loop*

n **=** int(input())

i **=** 2

**while** i **<=** n:

**if** i **%** 7 **==** 0:

i **+=**2 *# we have to put an increment here as in while continue is also skipping the i +=2 which is written at the bottom, so we ahve to write it inside the intend*

**continue**

print(i)

i **+=** 2

2

4

6

8

10

12

16

18

20

22

24

26

30

32

34

36

38

40

44

46

48

50

### Pass Statement

*# dummy statement if we do not want to write anything inside the loop*

i **=** 1

**if** i **<**7:

**pass**

print(" skipped the if statement")

skipped the if statement

i **=** 1

**while** i**<=** 3:

**pass**

i **+=**1

print('passed the while statement')

passed the while statement

**for** i **in** range(1, 10):

**pass**

print('passed the for loop')

passed the for loop

### Questoin 3

'''

Binary Pattern

Print the following pattern for the given number of rows.

Pattern for N = 4

1111

000

11

0

Input format : N (Total no. of rows)

Output format : Pattern in N lines

'''

n**=**int(input())

**for** i **in** range(1,n**+**1):

**for** k **in** range(1,n**-**i**+**2):

**if** i **%2**!=0:

print(1,end**=**"")

**else**:

print(0,end**=**"")

print()

1111

000

11

0

n **=** int(input())

i **=** 1

**while** i **<=**n:

k **=** 1

**while** k **<=** n**-**i**+**1:

**if** i **%2**!=0:

print(1,end**=**"")

**else**:

print(0,end**=**"")

k **+=**1

print()

i **+=**1

1111

000

11

0

### Question 4

'''

Print Number Pyramid

Print the following pattern for a given n.

For eg. N = 6

123456

23456

3456

456

56

6

56

456

3456

23456

123456

'''

n **=** int(input())

n1 **=** n

n2 **=** n **-** 1

*# Top half*

i **=** 1

**while** i**<=**n1:

*# top spaces*

j **=** 1

**while** j**<=** i **-**1:

print(' ',end **=**'')

j **+=**1

*# top num*

k **=** 1

p **=** i

**while** k **<=**n1 **-**i**+**1:

print(p,end**=**'')

k**+=**1

p **+=**1

print()

i**+=**1

y **=** 1

**while** y**<=**n2:

*# top spaces*

j **=** 1

**while** j**<=** n2 **-**y:

print(' ',end **=**'')

j **+=**1

*# top num*

k **=** 1

p **=** n2 **-**y **+**1

**while** k **<=**y **+**1:

print(p,end**=**'')

k**+=**1

p**+=**1

print()

y **+=**1

1234

234

34

4

34

234

1234

### Question 5

Rectangular numbers

Print the following pattern for the given number of rows.  
Pattern for N = 4  
4444444  
4333334  
4322234  
4321234  
4322234  
4333334  
4444444

*## unable to do copy from the interner change later after understanding*

n**=** int(input())

answer**=**[[1]]

**for** i **in** range(2, n**+**1):

t**=**[i]**\***((2**\***i)**-**3)

answer**.**insert(0, t)

answer**.**append(t**.**copy())

**for** a **in** answer:

a**.**insert(0,i)

a**.**append(i)

answerfinal**=**[]

**for** a **in** answer:

answerfinal**.**append(str(a)**.**replace(' ','')**.**replace(',','')**.**replace(']','')**.**replace('[',''))

**for** a **in** answerfinal:

print(a)

555555555

544444445

543333345

543222345

543212345

543222345

543333345

544444445

555555555

*# solution seen*

n **=** int(input())

**for** i **in** range(1,n**+**1):

temp **=** n

**for** j **in** range(1,i):

print(temp,end**=**"")

temp **=** temp **-**1

**for** j **in** range(1,(2**\***n) **-** (2**\***i) **+** 2):

print(n**-**i**+**1,end**=**"")

**for** j **in** range(1,i):

temp **=** temp**+**1

print(temp,end**=**"")

print()

**for** i **in** range(n**-**1,0,**-**1):

temp **=** n

**for** j **in** range(1,i):

print(temp,end**=**"")

temp **=** temp **-** 1

**for** j **in** range(1,(2**\***n) **-** (2**\***i) **+** 2):

print(n**-**i**+**1,end**=**"")

**for** j **in** range(1,i):

temp **=** temp**+**1

print(temp,end**=**"")

print()

555555555

544444445

543333345

543222345

543212345

543222345

543333345

544444445

555555555

### Question 6

'''

Time Complexity: O(N ^ 2)

Space Complexity: O(N ^ 2)

Where N is the given number of rows.

'''

**def** printPattern(n):

*# Declaring a vector of string for storing the pattern*

result **=** []

*# Keeping track of the start value of each row*

startValue **=** 1

**for** row **in** range(1, n **+** 1):

*# Declaring a string to store our row in the pattern*

ans **=** ""

*# Appending value of each column of each row in ans*

**for** col **in** range(startValue, startValue **+** n):

ans **+=** str(col) **+** " "

*# 'col' represents our current columnn in our current row*

**if** (row **==** (n **+** 1) **//** 2):

**if** (n **%** 2 **!=** 0):

startValue **=** n **\*** (n **-** 2) **+** 1

**else**:

startValue **=** n **\*** (n **-** 1) **+** 1

**elif** (row **>** (n **+** 1) **//** 2):

startValue **=** startValue **-** 2 **\*** n

**else**:

startValue **=** startValue **+** 2 **\*** n

result**.**append(ans)

**return** result

n **=** int(input())

5 6 7 8

### Question 7

'''

Code : Diamond of stars

Send Feedback

Print the following pattern for the given number of rows.

Note: N is always odd.

Pattern for N = 5

..\*..

.\*\*\*.

\*\*\*\*\*

.\*\*\*.

..\*..

The dots represent spaces.

'''

*# conver the code into for loops TODO*

n **=** int(input())

*# top half*

n1 **=** (n**+**1)**//**2

i **=** 1

**while** i **<=** n1:

j **=** 1

*#top spaces*

**while** j **<=** n1 **-** i:

print(' ', end **=** '')

j **+=**1

*#top stars*

k **=** 1

**while** k **<=** 2 **\*** i **-** 1:

print('\*',end **=** '')

k **+=**1

i **+=**1

print()

n2 **=** n**//**2

y **=** 1

**while** y **<=** n2:

*#bottom spaces*

l **=** 1

**while** l **<=** y:

print(' ',end **=** '')

l **+=**1

*#bottom star*

m **=** 1

**while** m **<=** n1**+**n2**-**2 **\*** y :

print('\*',end **=** '')

m **+=**1

y **+=**1

print()

### Question 8

Print the pattern

Print the following pattern for the given number of rows.

Pattern for N = 5

1 2 3 4 5

11 12 13 14 15

21 22 23 24 25

16 17 18 19 20

6 7 8 9 10

Input format : N (Total no. of rows)

Output format : Pattern in N lines

Sample Input :

4

Sample Output :

1 2 3 4

9 10 11 12

13 14 15 16

5 6 7 8

**Arrow pattern**

#### **Print the following pattern for the given number of rows.**

#### **Assume N is always odd.**

##### Note : There is space after every star.

##### Pattern for N = 7

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

##### Input format :

Integer N (Total no. of rows)

##### Output format :

Pattern in N lines

##### Sample Input :

11

##### Sample Output :

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

N = int(input())  
N1 = (N + 1) // 2  
N2 = N // 2  
  
i = 1  
while (i <= N1):  
 # top half spaces  
 j = 1  
 while (j <= i - 1):  
 print(' ', end="")  
 j += 1  
  
 # top half stars  
 k = 1  
 while (k <= i):  
 print('\* ', end="")  
 k += 1  
 print()  
 i += 1  
  
y = 1  
while (y <= N2):  
 # top half spaces  
 j = 1  
 while (j <= N2 - y):  
 print(' ', end="")  
 j += 1  
  
 # top half stars  
 k = 1  
 while (k <= N2 - y + 1):  
 print('\* ', end="")  
 k += 1  
 print()  
 y += 1

11

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**Print Multiple of 3**

a = int(input())  
b = int(input())  
if a % 3==0:  
 s=a   
elif a%3==1:  
 s = a +2  
else:  
 s = a+1  
for i in range(s, b+1, 3):  
 print(i)

**Checking Prime number**

n = int(input())  
flag = False  
for d in range(2, n, 1):  
 if n % d ==0:  
 flag = True  
if flag:  
 print("not prime")  
else:  
 print("Prime")

**Pattern**

n = int(input())  
for i in range(1, n+1, 1):  
 for s in range(n-i):  
 print("", end="")  
 for j in range(i, 2\*i, 1):  
 print(j, end="")  
 for j in range(2\*i - 2, i-1,-1):  
 print(j, end="")  
 print()

**Print Number Pyramid**

#### **Print the following pattern for a given n.**

##### For eg. N = 6

123456

23456

3456

456

56

6

56

456

3456

23456

123456

##### Sample Input 1 :

4

##### Sample Output 1 :

1234

234

34

4

34

234

123

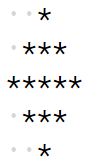
N = 6  
for i in range(1, N + 1):  
 for j in range(i - 1):  
 print(' ', end='')  
 for j in range(i, N + 1):  
 print(j, end='')  
 print('')  
for i in range(1, N):  
 for s in range(N-i-1):  
 print(' ', end='')  
 for j in range(N-i, N + 1):  
 print(j, end='')  
 print('')

**Diamond of Stars**

#### **Print the following pattern for the given number of rows.**

##### Note: N is always odd.

##### Pattern for N = 5



#### **The dots represent spaces.**

##### Input format :

N (Total no. of rows and can only be odd)

##### Output format :

Pattern in N lines

##### Constraints :

1 <= N <= 49

##### Sample Input 1:

5

##### Sample Output 1:

\*

\*\*\*

\*\*\*\*\*

\*\*\*

\*

##### Sample Input 2:

3

##### Sample Output 2:

\*

\*\*\*

\*

N = int(input())  
firstHalf = (N + 1) // 2  
secondHalf = N // 2  
# First Half  
currRow = 1  
while currRow <= firstHalf:  
 spaces = 1  
 while spaces <= (firstHalf - currRow):  
 print(" ", end="")  
 spaces += 1  
 currCol = 1  
 while currCol <= (2 \* currRow) - 1:  
 print("\*", end="")  
 currCol += 1  
  
 print()  
 currRow += 1  
# Second Half  
currRow = secondHalf  
while currRow >= 1:  
 spaces = 1  
 while spaces <= (secondHalf - currRow + 1):  
 print(" ", end="")  
 spaces += 1  
 currCol = 1  
 while currCol <= (2 \* currRow) - 1:  
 print("\*", end="")  
 currCol += 1  
  
 print()  
 currRow -= 1

**Rectangular numbers**

#### **Print the following pattern for the given number of rows.**

##### Pattern for N = 4

4444444

4333334

4322234

4321234

4322234

4333334

4444444

Input format : N (Total no. of rows)

Output format : Pattern in N lines

##### Sample Input :

3

##### Sample Output :

33333

32223

32123

32223

33333

N = int(input())  
  
for i in range(1, N + 1, 1):  
 for up in range(1, i, 1):  
 print(N - up + 1, end='')  
  
 for j in range(i, 2 \* N - i + 1, 1):  
 print(N - i + 1, end='')  
  
 for lp in range(2 \* N - i, 2 \* N - 1, 1):  
 print(lp - N + 2, end='')  
  
 print()  
  
for i in range(N - 1, 0, -1):  
 for up in range(1, i, 1):  
 print(N - up + 1, end='')  
  
 for j in range(i, 2 \* N - i + 1, 1):  
 print(N - i + 1, end='')  
  
 for lp in range(2 \* N - i, 2 \* N - 1, 1):  
 print(lp - N + 2, end='')  
  
 print()

**Print the pattern**

#### **Print the following pattern for the given number of rows.**

##### Pattern for N = 5

1 2 3 4 5

11 12 13 14 15

21 22 23 24 25

16 17 18 19 20

6 7 8 9 10

Input format : N (Total no. of rows)

Output format : Pattern in N lines

##### Sample Input :

4

##### Sample Output :

1 2 3 4

9 10 11 12

13 14 15 16

5 6 7 8

N=int(input())  
  
for i in range(1,N\*\*2,(2\*N)):  
 for j in range(1,N+1):  
 print(i+j-1,end=" ")  
 print()  
if N%2==0:  
 p=N-1  
else:  
 p=N-2  
  
  
for i1 in range(((N\*p)+1),N,-(2\*N)):  
 for j1 in range(1,N+1):  
 print(i1+j1-1,end=" ")  
 print()

**Functions and how to uses this:**

def fact(a):  
 a\_fact = 1  
 for i in range(1, a + 1):  
 a\_fact = a\_fact \* i  
 return a\_fact  
a=fact(4)  
a

n = int(input())  
r = int(input())  
n\_fact =fact(n)  
r\_fact = fact(r)  
n\_r\_fact = fact(n - r)  
ans = n\_fact//(r\_fact\*n\_r\_fact)  
print(ans)

**Fahrenheit to Celsius Function**

**Send Feedback**

#### **Given three values - Start Fahrenheit Value (S), End Fahrenheit value (E) and Step Size (W), you need to convert all Fahrenheit values from Start to End at the gap of W, into their corresponding Celsius values and print the table.**

##### Input Format :

3 integers - S, E and W respectively

##### Output Format :

Fahrenheit to Celsius conversion table. One line for every Fahrenheit and Celsius Fahrenheit value. Fahrenheit value and its corresponding Celsius value should be separate by tab ("\t")

##### Constraints :

0 <= S <= 1000

0 <= E <= 1000

0 <= W <= 1000

##### Sample Input 1:

0

100

20

##### Sample Output 1:

0 -17

20 -6

40 4

60 15

80 26

100 37

##### Sample Input 2:

120

200

40

##### Sample Output 2:

120 48

160 71

200 93

##### Explanation for Sample Output 2 :

Start value is 120, end value is 200 and step size is 40. Therefore, the values we need to convert are 120, 120 + 40 = 160, and 160 + 40 = 200.

The formula for converting Fahrenheit to Celsius is:

Celsius Value = (5/9)\*(Fahrenheit Value - 32)

Plugging 120 into the formula, the celsius value will be (5 / 9)\*(120 - 32) => (5 / 9) \* 88 => (5 \* 88) / 9 => 440 / 9 => 48.88

But we'll only print 48 because we are only interested in the integral part of the value.

def printTable(s,e,w):  
 while True:  
 c =0  
 if s <=e:  
 c = (s - 32) \* 5 / 9  
 print(s, int(c))  
 s = s + w  
 else:  
 break  
s = int(input())  
e = int(input())  
step = int(input())  
printTable(s, e, step)

**INPUT**

0

100

20

**OUTPUT**

0 -17

20 -6

40 4

60 15

80 26

100 37s

**Fibonacci Member**

**Send Feedback**

#### **Given a number N, figure out if it is a member of fibonacci series or not. Return true if the number is member of fibonacci series else false.**

#### **Fibonacci Series is defined by the recurrence**

F(n) = F(n-1) + F(n-2)

#### **where F(0) = 0 and F(1) = 1**

##### Input Format :

Integer N

##### Output Format :

true or false

##### Constraints:

0 <= n <= 10^4

##### Sample Input 1 :

5

##### Sample Output 1 :

true

##### Sample Input 2 :

14

##### Sample Output 2 :

false

OUTPUT: -

def checkMember(n):  
 if (n ==0):  
 return 0  
 elif (n== 1):  
 return 1  
 else:  
 return checkMember(n -1 ) + checkMember(n -2 )  
  
  
n = int(input())  
if (checkMember(n%2)):  
 print("true")  
else:  
 print("false")

**Palindrome number**

#### **Write a program to determine if given number is palindrome or not. Print true if it is palindrome, false otherwise.**

##### Palindrome are the numbers for which reverse is exactly same as the original one. For eg. 121

##### Sample Input 1 :

121

##### Sample Output 1 :

true

##### Sample Input 2 :

1032

##### Sample Output 2 :

false

def checkPalindrome(num):

    n = num

    rn = 0

    while (n > 0):

        r = n % 10

        rn = (rn \* 10) + r

        n = n // 10

    if rn == num:

        return True

    else:

        return False

num = int(input())

isPalindrome = checkPalindrome(num)

if(isPalindrome):

    print('true')

else:

    print('false')

**Check Armstrong**

#### **Write a Program to determine if the given number is Armstrong number or not. Print true if number is armstrong, otherwise print false.**

##### An Armstrong number is a number (with digits n) such that the sum of its digits raised to nth power is equal to the number itself.

#### **For example,**

371, as 3^3 + 7^3 + 1^3 = 371

1634, as 1^4 + 6^4 + 3^4 + 4^4 = 1634

##### Input Format :

Integer n

##### Output Format :

true or false

##### Sample Input 1 :

1

##### Sample Output 1 :

true

##### Sample Input 2 :

103

##### Sample Output 2 :

false

n = int(input())  # or n=int(input()) -> taking input from user

s = n  # assigning input value to the s variable

b = len(str(n))

sum1 = 0

while n != 0:

    r = n % 10

    sum1 = sum1+(r\*\*b)

    n = n//10

if s == sum1:

    print("true")

else:

    print("false")

**TEST -BASE**

**Number Star pattern 1**

#### **Print the following pattern for given number of rows.**

##### Input format :

Integer N (Total number of rows)

##### Output Format :

Pattern in N lines

##### Sample Input :

5

##### Sample Output :

5432\*

543\*1

54\*21

5\*321

\*4321

from os import \*

from sys import \*

from collections import \*

from math import \*

lines = int(input())

i = 1

while(i<=lines):

    j = lines

    while(j>=1):

        if j!=i:

            print(j, end="", flush=True)

        else:

            print('\*', end="", flush=True)

        j = j -1

    print()

    i = i + 1

**Zeros and Stars Pattern**

#### **Print the following pattern**

#### **Pattern for N = 4**

\*000\*000\*

0\*00\*00\*0

00\*0\*0\*00

000\*\*\*000

##### Input Format :

N (Total no. of rows)

##### Output Format :

Pattern in N lines

##### Sample Input 1 :

3

##### Sample Output 1 :

\*00\*00\*

0\*0\*0\*0

00\*\*\*00

##### Sample Input 2 :

5

##### Sample Output 2 :

\*0000\*0000\*

0\*000\*000\*0

00\*00\*00\*00

000\*0\*0\*000

0000\*\*\*0000

Algorithm  
STEP 1: START  
STEP 2: SET lines=4  
STEP 3: Initialize i and j  
STEP 4: SET i =1  
STEP 5: REPEAT STEP 6 to 15 UNTIL i less than or equals to line  
STEP 6: SET j=1  
STEP 7: REPET STEP 8 and 9 UNTIL j is less than or equals to lines  
STEP 8: IF i is equal to j PRINT \* ELSE PRINT 0  
STEP 9: SET j = j + 1  
STEP 10: DECREMENT j by 1 and PRINT \*  
STEP 11: REPEAT STEP 12 and 13 UNTIL j is greater than 0  
STEP 12: IF i is equals to j PRINT \* ELSE PRINT 0  
STEP 13: SET j = j - 1  
STEP 14: PRINT a new line  
STEP 15: SET i = i + 1  
STEP 16: EXIT

***Solution:-***

from os import \*

from sys import \*

from collections import \*

from math import \*

lines = int(input())

i = 1

j = 1

while i <= lines:

    j = 1

    while j <= lines:

        if i == j:

            print('\*', end="", flush=True)

        else:

            print('0', end="", flush=True)

        j = j + 1

    j = j - 1

    print('\*', end="", flush=True)

    while j >= 1:

        if i == j:

            print('\*', end="", flush=True)

        else:

            print('0', end="", flush=True)

        j -= 1

    print()

    i = i + 1

4

\*000\*000\*

0\*00\*00\*0

00\*0\*0\*00

000\*\*\*000

**Check Armstrong**

#### **Write a Program to determine if the given number is Armstrong number or not. Print true if number is Armstrong, otherwise print false.**

##### An Armstrong number is a number (with digits n) such that the sum of its digits raised to nth power is equal to the number itself.

#### **For example,**

371, as 3^3 + 7^3 + 1^3 = 371

1634, as 1^4 + 6^4 + 3^4 + 4^4 = 1634

##### Input Format :

Integer n

##### Output Format :

true or false

##### Sample Input 1 :

1

##### Sample Output 1 :

true

##### Sample Input 2 :

103

##### Sample Output 2 :

false

from os import \*

from sys import \*

from collections import \*

from math import \*

n = int(input())  # or n=int(input()) -> taking input from user

s = n  # assigning input value to the s variable

b = len(str(n))

sum1 = 0

while n != 0:

    r = n % 10

    sum1 = sum1+(r\*\*b)

    n = n//10

if s == sum1:

    print("true")

else:

    print("false")

**Array Sum**

#### **Given an array of length N, you need to find and print the sum of all elements of the array.**

##### Input Format :

Line 1 : An Integer N i.e. size of array

Line 2 : N integers which are elements of the array, separated by spaces

##### Output Format :

Sum

##### Constraints :

#### **1 <= N <= 10^6**

##### Sample Input :

3

9 8 9

##### Sample Output :

26

n = int(input())  
pr = input()  
a = pr.split()  
sum = 0  
for i in a:  
 sum = sum + int(i)  
print(sum)

**Swap Alternate**

#### **You have been given an array/list(ARR) of size N. You need to swap every pair of alternate elements in the array/list.**

#### **You don't need to print or return anything, just change in the input array itself.**

##### Input Format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output Format :

For each test case, print the elements of the resulting array in a single row separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^5

Time Limit: 1sec

##### Sample Input 1:

1

6

9 3 6 12 4 32

##### Sample Output 1 :

3 9 12 6 32 4

##### Sample Input 2:

2

9

9 3 6 12 4 32 5 11 19

4

1 2 3 4

##### Sample Output 2 :

3 9 12 6 32 4 11 5 19

2 1 4 3

from sys import stdin

def swapAlternate(li, n) :

    # Your code

    l = len(li)

    if l%2 == 0:

        for i in range(0,l,2):

            li[i],li[i+1] = li[i+1],li[i]

    else:

        for i in range(0,l-1,2):

            li[i],li[i+1] = li[i+1],li[i]

**Find Unique**

#### **You have been given an integer array/list(ARR) of size N. Where N is equal to [2M + 1].**

#### **Now, in the given array/list, 'M' numbers are present twice and one number is present only once.**

#### **You need to find and return that number which is unique in the array/list.**

##### Note:

Unique element is always present in the array/list according to the given condition.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output Format :

For each test case, print the unique element present in the array.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

Time Limit: 1 sec

##### Sample Input 1:

1

7

2 3 1 6 3 6 2

##### Sample Output 1:

1

##### Sample Input 2:

2

5

2 4 7 2 7

9

1 3 1 3 6 6 7 10 7

##### Sample Output 2:

4

10

    def find\_unique(li, n):

    for i in range(n):

        j = 0

        while(j < n):

            if i != j:

                if li[i] == li[j]:

                    break

            j = j + 1

        if j == n:

            return li[i]

**Find Duplicate**

#### **You have been given an integer array/list(ARR) of size N which contains numbers from 0 to (N - 2). Each number is present at least once. That is, if N = 5, the array/list constitutes values ranging from 0 to 3 and among these, there is a single integer value that is present twice. You need to find and return that duplicate number present in the array.**

##### Note :

Duplicate number is always present in the given array/list.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output Format :

For each test case, print the duplicate element in the array/list.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

Time Limit: 1 sec

##### Sample Input 1:

1

9

0 7 2 5 4 7 1 3 6

##### Sample Output 1:

7

##### Sample Input 2:

2

5

0 2 1 3 1

7

0 3 1 5 4 3 2

##### Sample Output 2:

1

3

def duplicateNumber(arr, n) :

#Your code goes here

    for i in range(len(arr)):

        for j in range(i + 1, n):

            if arr[i] == arr[j]:

                return arr[i]

            # else:

            #     return arr[j]

**Intersection of Two Arrays II**

#### **You have been given two integer arrays/list(ARR1 and ARR2) of size N and M, respectively. You need to print their intersection; An intersection for this problem can be defined when both the arrays/lists contain a particular value or to put it in other words, when there is a common value that exists in both the arrays/lists.**

##### Note :

Input arrays/lists can contain duplicate elements.

The intersection elements printed would be in the order they appear in the first array/list(ARR1)

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the first array/list.

Second line contains 'N' single space separated integers representing the elements of the first the array/list.

Third line contains an integer 'M' representing the size of the second array/list.

Fourth line contains 'M' single space separated integers representing the elements of the second array/list.

##### Output format :

For each test case, print the intersection elements in a row, separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

0 <= M <= 10^3

Time Limit: 1 sec

##### Sample Input 1 :

2

6

2 6 8 5 4 3

4

2 3 4 7

2

10 10

1

10

##### Sample Output 1 :

2 4 3

10

##### Sample Input 2 :

1

4

2 6 1 2

5

1 2 3 4 2

##### Sample Output 2 :

2 1 2

##### Explanation for Sample Output 2 :

Since, both input arrays have two '2's, the intersection of the arrays also have two '2's. The first '2' of first array matches with the first '2' of the second array. Similarly, the second '2' of the first array matches with the second '2' if the second array.

import sys

def intersections(arr1, n, arr2, m) :

    for i in range(n):

        for j in range(m):

            if arr1[i] == arr2[j]:

                print(arr1[i], end=" ")

                arr2[j] = sys.maxsize

                break

**Pair Sum**

#### **You have been given an integer array/list(ARR) and a number X. Find and return the total number of pairs in the array/list which sum to X.**

##### Note:

Given array/list can contain duplicate elements.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the first array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

Third line contains an integer 'X'.

##### Output format :

For each test case, print the total number of pairs present in the array/list.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

0 <= X <= 10^9

Time Limit: 1 sec

##### Sample Input 1:

1

9

1 3 6 2 5 4 3 2 4

7

##### Sample Output 1:

7

##### Sample Input 2:

2

9

1 3 6 2 5 4 3 2 4

12

6

2 8 10 5 -2 5

10

##### Sample Output 2:

0

2

##### Explanation for Input 2:

Since there doesn't exist any pair with sum equal to 12 for the first query, we print 0.

For the second query, we have 2 pairs in total that sum up to 10. They are, (2, 8) and (5, 5).

from sys import stdin

def pairSum(arr, n, x) :

    #Your code goes here

    count = 0

    for i in range(0, n):

        for j in range(i + 1, n):

            if arr[i] + arr[j] == x:

                count += 1

    return count

**Triplet Sum**

#### **You have been given a random integer array/list(ARR) and a number X. Find and return the number of triplets in the array/list which sum to X.**

##### Note :

Given array/list can contain duplicate elements.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the first array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

Third line contains an integer 'X'.

##### Output format :

For each test case, print the total number of triplets present in the array/list.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 50

0 <= N <= 10^2

0 <= X <= 10^9

Time Limit: 1 sec

##### Sample Input 1:

1

7

1 2 3 4 5 6 7

12

##### Sample Output 1:

5

##### Sample Input 2:

2

7

1 2 3 4 5 6 7

19

9

2 -5 8 -6 0 5 10 11 -3

10

##### Sample Output 2:

0

5

##### Explanation for Input 2:

Since there doesn't exist any triplet with sum equal to 19 for the first query, we print 0.

For the second query, we have 5 triplets in total that sum up to 10. They are, (2, 8, 0), (2, 11, -3), (-5, 5, 10), (8, 5, -3) and (-6, 5, 11)

def findTriplet(arr, n, x):

    #Your code goes here

    #return your answer

    count = 0

    for i in range(0, n):

        for j in range(i+1, n):

            for k in range(j+1, n):

                if x == arr[i]+arr[j]+arr[k]:

                    count = count+1

    return count

**Sort 0 1**

#### **You have been given an integer array/list(ARR) of size N that contains only integers, 0 and 1. Write a function to sort this array/list. Think of a solution which scans the array/list only once and don't require use of an extra array/list.**

##### Note:

You need to change in the given array/list itself. Hence, no need to return or print anything.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers(all 0s and 1s) representing the elements in the array/list.

##### Output format :

For each test case, print the sorted array/list elements in a row separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^5

Time Limit: 1 sec

##### Sample Input 1:

1

7

0 1 1 0 1 0 1

##### Sample Output 1:

0 0 0 1 1 1 1

##### Sample Input 2:

2

8

1 0 1 1 0 1 0 1

5

0 1 0 1 0

##### Sample Output 2:

0 0 0 1 1 1 1 1

0 0 0 1 1

def sortZeroesAndOne(arr, n) :

    #our code goes here

    lo = 0

    hi = n - 1

    mid = 0

    # Iterate till all the elements

    # are sorted

    while mid <= hi:

        # If the element is 0

        if arr[mid] == 0:

            arr[lo], arr[mid] = arr[mid], arr[lo]

            lo = lo + 1

            mid = mid + 1

        # If the element is 1

        elif arr[mid] == 1:

            mid = mid + 1

        # If the element is 2

        else:

            arr[mid], arr[hi] = arr[hi], arr[mid]

            hi = hi - 1

    return arr

**Code Binary Search**

#### **You have been given a sorted(in ascending order) integer array/list(ARR) of size N and an element X. Write a function to search this element in the given input array/list using 'Binary Search'. Return the index of the element in the input array/list. If the element is not present in the array/list, then return -1.**

##### Input format :

The first line contains an Integer 'N', which denotes the size of the array/list.

The second line contains 'N' single space-separated integers representing the elements in the array/list.

The third line contains the value of X to be searched for in the array/list.

##### Output Format :

Print the index at which X is present for each test case, -1 otherwise.

##### Constraints :

0 <= N <= 10^6

0 <= X <= 10^9

Time Limit: 1 sec

##### Sample Input 1:

7

1 3 7 9 11 12 45

3

##### Sample Output 1:

1

##### Sample Input 2:

7

1 2 3 4 5 6 7

9

##### Sample Output 2:

-1

def binarySearch(arr : List[int], n : int, x : int) :

    #Your code goes here

    #Your code goes here

    lower = 0

    upper = n-1

    for i in range(0, n):

        if lower > upper:

            # print(-1)

            return -1

        middle = int((lower+upper)/2)

        if arr[middle] < x:

            lower = middle+1

        elif arr[middle] > x:

            upper = middle-1

        else:

            return middle

**Code Selection Sort**

#### **Provided with a random integer array/list(ARR) of size N, you have been required to sort this array using 'Selection Sort'.**

##### Note:

Change in the input array/list itself. You don't need to return or print the elements.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output format :

For each test case, print the elements of the array/list in sorted order separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

Time Limit: 1 sec

##### Sample Input 1:

1

7

2 13 4 1 3 6 28

##### Sample Output 1:

1 2 3 4 6 13 28

##### Sample Input 2:

2

5

9 3 6 2 0

4

4 3 2 1

##### Sample Output 2:

0 2 3 6 9

1 2 3 4

from sys import stdin

def selectionSort(arr, n) :

 for i in range(n):

        minpos = i

        for j in range(i,n):

            if arr[j] < arr[minpos]:

                minpos = j

        temp = arr[i]

        arr[i] = arr[minpos]

        arr[minpos] = temp

**Code Bubble Sort**

#### **Provided with a random integer array/list(ARR) of size N, you have been required to sort this array using 'Bubble Sort'.**

##### Note:

Change in the input array/list itself. You don't need to return or print the elements.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output format :

For each test case, print the elements of the array/list in sorted order separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

Time Limit: 1 sec

##### Sample Input 1:

1

7

2 13 4 1 3 6 28

##### Sample Output 1:

1 2 3 4 6 13 28

##### Sample Input 2:

2

5

9 3 6 2 0

4

4 3 2 1

##### Sample Output 2:

0 2 3 6 9

1 2 3 4

from sys import stdin

def bubbleSort(arr, n) :

    for i in range(len(arr)):

        for j in range(0, len(arr) - 1):

            if arr[j] > arr[j + 1]:

               arr[j], arr[j+1] = arr[j + 1], arr[j]

    return arr

**Code Insertion Sort**

#### **Provided with a random integer array/list(ARR) of size N, you have been required to sort this array using 'Insertion Sort'.**

##### Note:

Change in the input array/list itself. You don't need to return or print the elements.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output Format :

For each test case, print the elements of the array/list in sorted order separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

Time Limit: 1 sec

##### Sample Input 1:

1

7

2 13 4 1 3 6 28

##### Sample Output 1:

1 2 3 4 6 13 28

from sys import stdin

def insertionSort(arr, n) :

    #Your code goes here

    for i in range(1,len(arr)):

        n = arr[i]

        j = i -1

        while j >= 0 and n < arr[j]:

            arr[j + 1] = arr[j]

            j = j - 1

        arr[j + 1] = n

**Code Merge Two Sorted Arrays**

#### **You have been given two sorted arrays/lists(ARR1 and ARR2) of size N and M respectively, merge them into a third array/list such that the third array is also sorted.**

##### Input Format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the first array/list.

Second line contains 'N' single space separated integers representing the elements of the first array/list.

Third line contains an integer 'M' representing the size of the second array/list.

Fourth line contains 'M' single space separated integers representing the elements of the second array/list.

##### Output Format :

For each test case, print the sorted array/list(of size N + M) in a single row, separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^5

0 <= M <= 10^5

Time Limit: 1 sec

##### Sample Input 1 :

1

5

1 3 4 7 11

4

2 4 6 13

##### Sample Output 1 :

1 2 3 4 4 6 7 11 13

##### Sample Input 2 :

2

3

10 100 500

7

4 7 9 25 30 300 450

4

7 45 89 90

0

##### Sample Output 2 :

4 7 9 10 25 30 100 300 450 500

7 45 89 90

from sys import stdin

def merge(arr1, n, arr2, m) :

    #Your code goes here

    # r = len(arr1)//2, len(arr2)//2

    # n = arr1[:r]

    # m = arr2[r:]

    len1, len2 = len(arr1), len(arr2)

    new\_array = []

    n=0;m=0

    while (n < len1 and m < len2):

        if arr1[n] < arr2[m]:

            new\_array.append(arr1[n])

            n = n + 1

        else:

            new\_array.append(arr2[m])

            m = m+1

    while (n < len1):

        new\_array.append(arr1[n])

        n = n+1

    while (m < len2):

        new\_array.append(arr2[m])

        m = m+1

    return new\_array

#Taking Input Using Fast I/O

def takeInput() :

    n = int(stdin.readline().rstrip())

    if n != 0:

        arr = list(map(int, stdin.readline().rstrip().split(" ")))

        return arr, n

    return list(), 0

#to print the array/list

def printList(arr, n) :

    for i in range(n) :

        print(arr[i], end = " ")

    print()

#main

t = int(stdin.readline().rstrip())

while t > 0 :

    arr1, n = takeInput()

    arr2, m = takeInput()

    ans = merge(arr1, n, arr2, m)

    printList(ans, (n + m))

    t -= 1

**Push Zeros to end**

#### **You have been given a random integer array/list(ARR) of size N. You have been required to push all the zeros that are present in the array/list to the end of it. Also, make sure to maintain the relative order of the non-zero elements.**

##### Note:

Change in the input array/list itself. You don't need to return or print the elements.

You need to do this in one scan of array only. Don't use extra space.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output Format :

For each test case, print the elements of the array/list in the desired order separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^5

Time Limit: 1 sec

##### Sample Input 1:

1

7

2 0 0 1 3 0 0

##### Sample Output 1:

2 1 3 0 0 0 0

##### Explanation for the Sample Input 1 :

All the zeros have been pushed towards the end of the array/list. Another important fact is that the order of the non-zero elements have been maintained as they appear in the input array/list.

##### Sample Input 2:

2

5

0 3 0 2 0

5

9 0 0 8 2

##### Sample Output 2:

3 2 0 0 0

9 8 2 0 0

from sys import stdin

def pushZerosAtEnd(arr, n) :

    #Your code goes here

    count = 0

    for i in range(n):

        if arr[i] != 0:

            arr[count] = arr[i]

            count +=1

    while count < n:

            arr[count] = 0

            count +=1

**Rotate array**

#### **You have been given a random integer array/list(ARR) of size N. Write a function that rotates the given array/list by D elements(towards the left).**

##### Note:

Change in the input array/list itself. You don't need to return or print the elements.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

Third line contains the value of 'D' by which the array/list needs to be rotated.

##### Output Format :

For each test case, print the rotated array/list in a row separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^4

0 <= N <= 10^6

0 <= D <= N

Time Limit: 1 sec

##### Sample Input 1:

1

7

1 2 3 4 5 6 7

2

##### Sample Output 1:

3 4 5 6 7 1 2

##### Sample Input 2:

2

7

1 2 3 4 5 6 7

def rotate(arr, n, d):

    #Your code goes here

    n = len(arr)

    arr[:] = arr[d:n]+arr[0:d]

    return arr

**Second Largest in array**

#### **You have been given a random integer array/list(ARR) of size N. You are required to find and return the second largest element present in the array/list.**

#### **If N <= 1 or all the elements are same in the array/list then return -2147483648 or -2 ^ 31(It is the smallest value for the range of Integer)**

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

The first line of each test case or query contains an integer 'N' representing the size of the array/list.

The second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output Format :

For each test case, print the second largest in the array/list if exists, -2147483648 otherwise.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^5

Time Limit: 1 sec

##### Sample Input 1:

1

7

2 13 4 1 3 6 28

##### Sample Output 1:

13

##### Sample Input 2:

1

5

9 3 6 2 9

##### Sample Output 2:

6

##### Sample Input 3:

2

2

6 6

4

90 8 90 5

##### Sample Output 3:

-2147483648

8

def secondHighestElement(arr,n):

    unique\_elements = set(arr)

    # taking care of some base cases before initiating

    if n <= 1:

        return -2147483648

    # same here

    elif len(unique\_elements) == 1:

        return -2147483648

    # if the length of an array is greater than 1

    else:

        list\_element = list(unique\_elements)

        # we simply sort that array with sort() function

        list\_element.sort()

        # because after sorting the second largest will be present at [-2] index

        # we return that index [-2]

        return list\_element[-2]

t = int(input())

for index in range(t):

    n = int(input())

    if n:

        arr = [int(x)for x in input().split()]

        result = secondHighestElement(arr,n)

        print(result)

    else:

        arr = []

        result = secondHighestElement(arr,n)

        print(result)

**Sort 0 1 2**

#### **You are given an integer array/list(ARR) of size N. It contains only 0s, 1s and 2s. Write a solution to sort this array/list in a 'single scan'.**

#### **'Single Scan' refers to iterating over the array/list just once or to put it in other words, you will be visiting each element in the array/list just once.**

##### Note:

You need to change in the given array/list itself. Hence, no need to return or print anything.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers(all 0s, 1s and 2s) representing the elements in the array/list.

##### Output Format :

For each test case, print the sorted array/list elements in a row separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^5

Time Limit: 1 sec

##### Sample Input 1:

1

7

0 1 2 0 2 0 1

##### Sample Output 1:

0 0 0 1 1 2 2

##### Sample Input 2:

2

5

2 2 0 1 1

7

0 1 2 0 1 2 0

##### Sample Output 2:

0 1 1 2 2

0 0 0 1 1 2 2

from sys import stdin

def sort012(arr, n) :

    #Your code goes here

    count0 = 0

    count1 = 0

    count2 = 0

    # Count the number of 0s, 1s, and 2s in the array

    for i in range(n):

        if arr[i] == 0:

            count0 += 1

        elif arr[i] == 1:

            count1 += 1

        else:

            count2 += 1

    # Overwrite the original array with the sorted values

    i = 0

    while count0 > 0:

        arr[i] = 0

        i += 1

        count0 -= 1

    while count1 > 0:

        arr[i] = 1

        i += 1

        count1 -= 1

    while count2 > 0:

        arr[i] = 2

        i += 1

        count2 -= 1

**Check Array Rotation**

#### **You have been given an integer array/list(ARR) of size N. It has been sorted(in increasing order) and then rotated by some number 'K' (K is greater than 0) in the right hand direction.**

#### **Your task is to write a function that returns the value of 'K', that means, the index from which the array/list has been rotated.**

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

##### Output Format :

For each test case, print the value of 'K' or the index from which which the array/list has been rotated.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

2 <= N <= 10^5

Time Limit: 1 sec

##### Sample Input 1:

1

6

5 6 1 2 3 4

##### Sample Output 1:

2

##### Sample Input 2:

2

5

3 6 8 9 10

4

10 20 30 1

##### Sample Output 2:

0

3

from sys import stdin

def arrayRotateCheck(arr, n):

    #Your code goes here

    low = 0

    high = n - 1

    # Loop until the pointers meet or cross over

    while low <= high:

        mid = (low + high) // 2  # Calculate the middle index

        # Check if the middle element is the minimum element

        if arr[mid] < arr[(mid - 1) % n] and arr[mid] < arr[(mid + 1) % n]:

            return mid

        if arr[low] <= arr[mid]:

            if arr[mid] > arr[high]:

                low = mid + 1

            else:

                high = mid - 1

        else:

            if arr[mid] < arr[low]:

                high = mid - 1

            else:

                low = mid + 1

    return 0

#Taking Input Using Fast I/O

def takeInput() :

    n = int(stdin.readline().rstrip())

    if n == 0:

        return list(), 0

    arr = list(map(int, stdin.readline().rstrip().split(" ")))

    return arr, n

#main

t = int(stdin.readline().rstrip())

while t > 0 :

    arr, n = takeInput()

    print(arrayRotateCheck(arr, n))

    t -= 1

**Sum of Two Arrays**

#### **Two random integer arrays/lists have been given as ARR1 and ARR2 of size N and M respectively. Both the arrays/lists contain numbers from 0 to 9(i.e. single digit integer is present at every index). The idea here is to represent each array/list as an integer in itself of digits N and M.**

#### **You need to find the sum of both the input arrays/list treating them as two integers and put the result in another array/list i.e. output array/list will also contain only single digit at every index.**

##### Note:

The sizes N and M can be different.

Output array/list(of all 0s) has been provided as a function argument. Its size will always be one more than the size of the bigger array/list. Place 0 at the 0th index if there is no carry.

No need to print the elements of the output array/list.

Using the function "sumOfTwoArrays", write the solution to the problem and store the answer inside this output array/list. The main code will handle the printing of the output on its own.

##### Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the first array/list.

Second line contains 'N' single space separated integers representing the elements of the first array/list.

Third line contains an integer 'M' representing the size of the second array/list.

Fourth line contains 'M' single space separated integers representing the elements of the second array/list.

##### Output Format :

For each test case, print the required sum of the arrays/list in a row, separated by a single space.

Output for every test case will be printed in a separate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^5

0 <= M <= 10^5

Time Limit: 1 sec

##### Sample Input 1:

1

3

6 2 4

3

7 5 6

##### Sample Output 1:

1 3 8 0

##### Sample Input 2:

2

3

8 5 2

2

1 3

4

9 7 6 1

3

4 5 9

##### Sample Output 2:

0 8 6 5

1 0 2 2 0

from sys import stdin

def sumOfTwoArrays(arr1, n, arr2, m, output) :

    #Your code goes here

    i = n-1

    j = m-1

    carry = 0

    l = max(n, m)+1

    while i >= 0 and j >= 0:

        num = arr1[i]+arr2[j]+carry

        s = num % 10

        carry = num//10

        output[l-1] = s

        l = l-1

        i = i-1

        j = j-1

    while i >= 0:

        num = arr1[i]+carry

        s = num % 10

        carry = num//10

        output[l-1] = s

        l = l-1

        i = i-1

    while j >= 0:

        num = arr2[j]+carry

        s = num % 10

        carry = num//10

        output[l-1] = s

        l = l-1

        j = j-1

    if carry != 0:

        output[0] = carry

#Taking Input Using Fast I/O

def takeInput() :

    n = int(stdin.readline().rstrip())

    if n == 0 :

        return list(), 0

    arr = list(map(int, stdin.readline().rstrip().split(" ")))

    return arr, n

#to print the array/list

def printList(arr, n) :

    for i in range(n) :

        print(arr[i], end = " ")

    print()

#main

t = int(stdin.readline().rstrip())

while t > 0 :

    arr1, n = takeInput()

    arr2, m = takeInput()

    outputSize = (1 + max(n, m))

    output = outputSize \* [0]

    sumOfTwoArrays(arr1, n, arr2, m, output)

    printList(output, outputSize)

    t -= 1

**Check Palindrome**

#### **Given a string, determine if it is a palindrome, considering only alphanumeric characters.**

##### Palindrome

A palindrome is a word, number, phrase, or other sequences of characters which read the same backwards and forwards.

##### Example:

If the input string happens to be, "malayalam" then as we see that this word can be read the same as forward and backwards, it is said to be a valid palindrome.

The expected output for this example will print, 'true'.

#### **From that being said, you are required to return a boolean value from the function that has been asked to implement.**

##### Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

##### Output Format:

The only line of output prints either 'true' or 'false'.

##### Note:

You are not required to print anything. It has already been taken care of.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

##### Sample Input 1 :

abcdcba

##### Sample Output 1 :

true

##### Sample Input 2:

coding

##### Sample Output 2:

false

count = ""

string = str(input())

for i in string:

    count = i + count

if (string == count):

    print("true")

else:

    print("false")

Second method

n=input()  
if n[::-1]==n:  
 print("true")  
else:  
 print("false")

**Check Permutation**

#### **For a given two strings, 'str1' and 'str2', check whether they are a permutation of each other or not.**

##### Permutations of each other

Two strings are said to be a permutation of each other when either of the string's characters can be rearranged so that it becomes identical to the other one.

Example:

str1= "sinrtg"

str2 = "string"

The character of the first string(str1) can be rearranged to form str2 and hence we can say that the given strings are a permutation of each other.

##### Input Format:

The first line of input contains a string without any leading and trailing spaces, representing the first string 'str1'.

The second line of input contains a string without any leading and trailing spaces, representing the second string 'str2'.

##### Note:

All the characters in the input strings would be in lower case.

##### Output Format:

The only line of output prints either 'true' or 'false', denoting whether the two strings are a permutation of each other or not.

You are not required to print anything. It has already been taken care of. Just implement the function.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

##### Sample Input 1:

abcde

baedc

##### Sample Output 1:

true

##### Sample Input 2:

abc

cbd

##### Sample Output 2:

fal

from sys import stdin

def isPermutation(string1, string2) :

    #Your code goes here

    if len(string1) != len(string2):

        return False

    # Create a dictionary to keep track of the frequency of characters in string1

    char\_frequency = {}

    # Populate the dictionary

    for char in string1:

        if char in char\_frequency:

            char\_frequency[char] += 1

        else:

            char\_frequency[char] = 1

    # Check if each character in string2 has the same frequency as in string1

    for char in string2:

        if char in char\_frequency:

            char\_frequency[char] -= 1

            if char\_frequency[char] < 0:

                return False

        else:

            return False

    return True

**Remove Consecutive Duplicates**

#### **For a given string(str), remove all the consecutive duplicate characters.**

##### Example:

Input String: "aaaa"

Expected Output: "a"

Input String: "aabbbcc"

Expected Output: "abc"

###### Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

##### Output Format:

The only line of output prints the updated string.

##### Note:

You are not required to print anything. It has already been taken care of.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

##### Sample Input 1:

aabccbaa

##### Sample Output 1:

abcba

##### Sample Input 2:

xxyyzxx

##### Sample Output 2:

xyzx

from sys import stdin

def removeConsecutiveDuplicates(string) :

    result = ""

    for i in range(len(string)):

        if i == len(string) - 1 or string[i] != string[i+1]:

            result += string[i]

    return result

**Reverse Each Word**

#### **Aadil has been provided with a sentence in the form of a string as a function parameter. The task is to implement a function so as to print the sentence such that each word in the sentence is reversed.**

##### Example:

Input Sentence: "Hello, I am Aadil!"

The expected output will print, ",olleH I ma !lidaA".

##### Input Format:

The first and only line of input contains a string without any leading and trailing spaces. The input string represents the sentence given to Aadil.

##### Output Format:

The only line of output prints the sentence(string) such that each word in the sentence is reversed.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

##### Sample Input 1:

Welcome to Coding Ninjas

##### Sample Output 1:

emocleW ot gnidoC sajniN

##### Sample Input 2:

Always indent your code

##### Sample Output 2:

syawlA tnedni ruoy edoc

from sys import stdin

def reverseEachWord(string):

    words = string.split()  # split string into words

    reversed\_words = []     # create empty list for reversed words

    for word in words:

        reversed\_word = word[::-1]  # reverse each word

        reversed\_words.append(reversed\_word)  # add reversed word to list

    return ' '.join(reversed\_words)

#main

string = stdin.readline().strip()

ans = reverseEachWord(string)

print(ans)

**Remove character**

#### **For a given a string(str) and a character X, write a function to remove all the occurrences of X from the given string.**

#### **The input string will remain unchanged if the given character(X) doesn't exist in the input string.**

##### Input Format:

The first line of input contains a string without any leading and trailing spaces.

The second line of input contains a character(X) without any leading and trailing spaces.

##### Output Format:

The only line of output prints the updated string.

##### Note:

You are not required to print anything explicitly. It has already been taken care of.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

##### Sample Input 1:

aabccbaa

a

##### Sample Output 1:

bccb

##### Sample Input 2:

xxyyzxx

y

##### Sample Output 2:

xxzxx

from sys import stdin

def removeAllOccurrencesOfChar(string, ch):

    # Use a list comprehension to filter out the characters that match ch

    filtered\_chars = [char for char in string if char != ch]

    # Join the remaining characters back into a string and return it

    return ''.join(filtered\_chars)

#main

string = stdin.readline().strip()

ch = stdin.readline().strip()[0]

ans = removeAllOccurrencesOfChar(string, ch)

print(ans)

**Highest Occurring Character**

#### **For a given a string(str), find and return the highest occurring character.**

##### Example:

Input String: "abcdeapapqarr"

Expected Output: 'a'

Since 'a' has appeared four times in the string which happens to be the highest frequency character, the answer would be 'a'.

#### **If there are two characters in the input string with the same frequency, return the character which comes first.**

##### Consider:

Assume all the characters in the given string to be in lowercase always.

##### Input Format:

The first and only line of input contains a string without any leading and trailing spaces.

##### Output Format:

The only line of output prints the updated string.

##### Note:

You are not required to print anything explicitly. It has already been taken care of.

##### Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

##### Sample Input 1:

abdefgbabfba

##### Sample Output 1:

b

##### Sample Input 2:

xy

##### Sample Output 2:

x

from sys import stdin,setrecursionlimit

setrecursionlimit(10\*\*7)

def getCompressedString(a):

    i=0

    x=''

    while(i<len(a)):

        j=i+1

        c=1

        while j<len(a) and (a[i]==a[j]):

            j+=1

            c+=1

        if c==1:

            x+=a[i]

        else:

            x+=a[i]+str(c)

        i=j

    return x

# Main.

string = stdin.readline().strip();

ans = getCompressedString(string)

print(ans)

**Row Wise Sum**

#### **For a given two-dimensional integer array/list of size (N x M), find and print the sum of each of the row elements in a single line, separated by a single space.**

##### Input Format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains two integer values, 'N' and 'M', separated by a single space. They represent the 'rows' and 'columns' respectively, for the two-dimensional array/list.

Second line onwards, the next 'N' lines or rows represent the ith row values.

Each of the ith row constitutes 'M' column values separated by a single space.

##### Output Format :

For each test case, print the sum of every ith row elements in a single line separated by a single space.

Output for every test case will be printed in a seperate line.

##### Constraints :

1 <= t <= 10^2

0 <= N <= 10^3

0 <= M <= 10^3

Time Limit: 1sec

##### Sample Input 1:

1

4 2

1 2

3 4

5 6

7 8

##### Sample Output 1:

3 7 11 15

##### Sample Input 2:

2

2 5

4 5 3 2 6

7 5 3 8 9

4 4

1 2 3 4

9 8 7 6

3 4 5 6

-1 1 -10 5

##### Sample Output 2:

20 32

10 30 18 -5

from sys import stdin

def rowWiseSum(mat, nRows, mCols):

    for row in mat:

        sum=0

        for ele in row:

            sum=sum+ele

        print(sum,end=' ')

    #Your code goes here

#Taking Input Using Fast I/O

def take2DInput() :

    li = stdin.readline().rstrip().split(" ")

    nRows = int(li[0])

    mCols = int(li[1])

    if nRows == 0 :

        return list(), 0, 0

    mat = [list(map(int, input().strip().split(" "))) for row in range(nRows)]

    return mat, nRows, mCols

#main

t = int(stdin.readline().rstrip())

while t > 0 :

    mat, nRows, mCols = take2DInput()

    rowWiseSum(mat, nRows, mCols)

    print()

    t -= 1

**Largest Row or Column**

#### **For a given two-dimensional integer array/list of size (N x M), you need to find out which row or column has the largest sum(sum of all the elements in a row/column) amongst all the rows and columns.**

##### Note :

If there are more than one rows/columns with maximum sum, consider the row/column that comes first. And if ith row and jth column has the same largest sum, consider the ith row as answer.

##### Input Format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains two integer values, 'N' and 'M', separated by a single space. They represent the 'rows' and 'columns' respectively, for the two-dimensional array/list.

Second line onwards, the next 'N' lines or rows represent the ith row values.

Each of the ith row constitutes 'M' column values separated by a single space.

##### Output Format :

For each test case, If row sum is maximum, then print: "row" <row\_index> <row\_sum>

OR

If column sum is maximum, then print: "column" <col\_index> <col\_sum>

It will be printed in a single line separated by a single space between each piece of information.

Output for every test case will be printed in a seperate line.

##### Consider :

If there doesn't exist a sum at all then print "row 0 -2147483648", where -2147483648 or -2^31 is the smallest value for the range of Integer.

##### Constraints :

1 <= t <= 10^2

1 <= N <= 10^3

1 <= M <= 10^3

Time Limit: 1sec

'''

    In order to print two or more integers in a line separated by a single

    space then you may consider printing it with the statement,

    print(str(num1) + " " + str(num2))

    Take Minimum value as MIN\_VALUE = -2147483648

'''

from sys import stdin

def findLargest(arr, nRows, mCols):

    #Your code goes here

    rowIndex = 0

    colIndex = 0

    rowSum = -2147483648

    colSum = -2147483648

    for i in range(nRows):

        tempSum = sum(arr[i])

        if(tempSum>rowSum):

            rowSum = tempSum

            rowIndex = i

    for j in range(mCols):

        tempSum = 0

        for i in range(nRows):

            tempSum += arr[i][j]

        if(tempSum>colSum):

            colSum = tempSum

            colIndex = j

    if(rowSum>=colSum):

        print("row " + str(rowIndex) + " " + str(rowSum))

    else:

        print("column " + str(colIndex) + " " + str(colSum))

    # if nRows==0 and mCols==0:

    #     #print("row"+" "+str(0)+" "+str(-2147483648))

    #     #return

    # col\_max = -1

    # col\_index = -1

    # row\_max=-1

    # row\_index=-1

    # for i in range(mCols):

    #     col\_sum = 0

    #     for j in range(nRows):

    #         col\_sum+=arr[j][i]

    #     if col\_max<col\_sum:

    #         col\_max=col\_sum

    #         col\_index = i

    # for i in range(nRows):

    #     row\_sum=0

    #     for j in range(mCols):

    #         row\_sum+=arr[i][j]

    #     if row\_max<row\_sum:

    #         row\_max=row\_sum

    #         row\_index=i

    # if col\_max>row\_max:

    #     print("column"+" "+str(col\_index)+" "+str(col\_max))

    # elif row\_max>=col\_max:

    #     print("row"+" "+str(row\_index)+" "+str(row\_max))

#Taking Input Using Fast I/O

def take2DInput() :

    li = stdin.readline().rstrip().split(" ")

    nRows = int(li[0])

    mCols = int(li[1])

    if nRows == 0 :

        return list(), 0, 0

    mat = [list(map(int, input().strip().split(" "))) for row in range(nRows)]

    return mat, nRows, mCols

#main

t = int(stdin.readline().rstrip())

while t > 0 :

    mat, nRows, mCols = take2DInput()

    findLargest(mat, nRows, mCols)

    t -= 1