Maharaj Vijayaram Gajapathi Raj College of Engineering (AUTONOMOUS)

(Approved by AICTE, New Delhi and Affiliated to JNTU, Kakinada)

(Accredited by NBA & Certified by NAAC)

VIJAYARAM NAGAR CAMPUS, VIZIANAGARAM.

MANSAS

CERTIFICATE

This is to certify that this is the bonafide record of the work done in .... Laboratory by Mr/Ms .... bearing Regd . No. / Roll No .... of .... B.Tech / M.Tech. course during ....

Total Numbers of Experiments held ....

LAB-IN-CHARGE

HEAD OF THE DEPARTMENT

INDEX

WEEK-1

QUESTION 1 :

Write a program that takes three numbers and prints their sum. Every number is given on a separate line.

OBJECTIVE : To learn input, print, numbers.

PROGRAM :

```
a = int(input())

b = int(input())

c = int(input())

Print(a + b + c)
```

```
1. INPUT :                OUTPUT :
     2                    11
     3
     6

2. INPUT :                OUTPUT :
     0                    320
     20
     300

3. INPUT :                OUTPUT :
     -5                   158
     180
     -17
```

INFERENCE :

QUESTION 2 :

Write a program that greets the person by printing the word “Hi” and the name of the Person.

OBJECTIVE : To learn input, print, numbers.

PROGRAM :

```
name = input()
```

```
print('Hi ' + name)
```

1.	**INPUT :** John	**OUTPUT :** Hi John
2.	**INPUT :** Alice	**OUTPUT :** Hi Alice
3.	**INPUT :** Bob	**OUTPUT :** Hi Bob

INFERENCE :

QUESTION 3 :

Write a program that takes a number and print its square.

OBJECTIVE : To learn input, print, numbers.

PROGRAM :

```
a = int (input())

print(a ** 2)
```

1. INPUT : 6	**OUTPUT :** 36
2. INPUT : 5	**OUTPUT :** 25
3. INPUT : 2	**OUTPUT :** 4

INFERENCE :

QUESTION 4 :

Write a program that reads the length of the base and the height of a right-angled triangle and prints the area. Every number is given on a separate line.

OBJECTIVE : To learn input, print, numbers.

PROGRAM :

```
b = int(input())

h = int(input())

Print(0.5 * b * h)
```

```
1. INPUT :          OUTPUT :
     3              7.5
     5

2. INPUT :          OUTPUT :
     10             10.0
     2

3. INPUT :          OUTPUT :
     179            137293.0
     1534
```

INFERENCE :

QUESTION 5 :

Write a program that greets the user by printing the word “Hello”, a comma, the name of the user and an exclamation mark after it.

OBJECTIVE : To learn input, print, numbers.

PROGRAM :

```
name = input()

print(“Hello,” , name + “!”)
```

1. **INPUT :** Harry — **OUTPUT :** Hello, Harry!

2. **INPUT :** Mr. Potter — **OUTPUT :** Hello, Mr. Potter!

3. **INPUT :** Lord Voldemort — **OUTPUT :** Hello, Lord Voldemort!

INFERENCE :

WEEK-2

QUESTION 1 :

Given an integer number, print its last digit.

OBJECTIVE : Integers, Floats.

PROGRAM :

```
a=int(input())

print(a%10)
```

1. **INPUT :** 179 — **OUTPUT :** 9

2. **INPUT :** 40 — **OUTPUT :** 0

3. **INPUT :** 101 — **OUTPUT :** 1

INFERENCE :

QUESTION 2 :

Given a two-digit number, swap its digits.

OBJECTIVE : Integers, Floats.

PROGRAM :

```
n= int(input())

x= n//10

y = n%10

s= (y*10)+x

Print (s)
```

1. INPUT :	**OUTPUT :**
26	62
2. INPUT :	**OUTPUT :**
15	51
3. INPUT :	**OUTPUT :**
55	55

INFERENCE :

QUESTION 3 :

Given a three-digit number. Find the sum of its digits.

OBJECTIVE : Integers, Floats.

PROGRAM :

```
a = int(input())

print((a//100)+((a//10)%10) +(a%10))
```

1. INPUT :	**OUTPUT :**

179	17

2. **INPUT :** 829 | **OUTPUT :** 19

2. INPUT :	**OUTPUT :**
829	19
3. INPUT :	**OUTPUT :**
204	6

INFERENCE :

QUESTION 4 :

Given a positive real number, print its first digit to the right of the decimal point.

OBJECTIVE : Integers, Floats.

PROGRAM :

```
a = float(input())

b =int(a)

c= a – b

d = c*100

 Print (d // 10)
```

1. INPUT :	**OUTPUT :**
1.79	7
2. INPUT :	**OUTPUT :**
10.34	3
3. INPUT :	**OUTPUT :**
0.01	0

INFERENCE :

WEEK-3

QUESTION 1 :

Given an integer, print “YES” if it’s positive and print “NO” otherwise.

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a = int(input())
if a>0:
   print(‘YES’)
else:
   print (‘NO’)
```

1. **INPUT :** 26 — **OUTPUT :** YES
2. **INPUT :** 6 — **OUTPUT :** YES
3. **INPUT :** -9 — **OUTPUT :** NO

INFERENCE :

QUESTION 2 :

Given an integer, print “YES” if it’s odd and print “NO” otherwise.

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a=int(input())
if(a%2= = 1):
    print (‘YES’)
```

```
else :
    print('NO')
```

1. **INPUT :** 7 — **OUTPUT :** YES

2. **INPUT :** 2 — **OUTPUT :** NO

3. **INPUT :** 5 — **OUTPUT :** YES

INFERENCE :

QUESTION 3 :

Given an integer, print "YES" if it's last digit is 7 and print "NO" otherwise.

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a=int(input())
if(a%10==7):
    print ('YES')
else:
    print('NO')
```

1. **INPUT :** 27 — **OUTPUT :** YES

1. **INPUT :** 56 — **OUTPUT :** NO

2. **INPUT :** 77 — **OUTPUT :** YES

INFERENCE :

QUESTION 4 :

Given two integers, print the smaller value.

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a=int(input())
b=int(input())
if a<b:
    print(a)
else:
   print (b)
```

```
      1. INPUT :                    OUTPUT :
            4                       4
            5

      2. INPUT :                    OUTPUT :
            8                        7
            7

      3. INPUT :                    OUTPUT :
            3                       1
            1
```

INFERENCE :

QUESTION 5 :

Given two integers, print "YES" if exactly one of them is odd and print "NO" otherwise.

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a = int(input())

b = int(input())

if(a%2==1 and b%2==0) or (a%2==0 and b%2==1):

    print ('YES')


else:

    print ('NO')
```

1. **INPUT :** **OUTPUT :**

```
25      YES
26
```

2. **INPUT :** **OUTPUT :**

```
12      NO
14
```

3. **INPUT :** **OUTPUT :**

```
57      YES
82
```

INFERENCE :

QUESTION 6 :

Given three different integers, print YES if they're given in ascending order, print NO otherwise.

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a = int(input())

b  =  int(input())

c =int(input())
```

```
if(a<b<c):
    print ('YES')
else :
    print ('NO')
```

1. **INPUT :**   **OUTPUT:**

```
10          YES
20
30
```

2. **INPUT :**   **OUTPUT:**

```
44          NO
22
66
```

3. **INPUT :**   **OUTPUT :**

```
49          YES
65
89
```

INFERENCE :

QUESTION 7 :

A palindrome is a number which reads the same when read forward as it it does when read backward. Given a four-digit integer, print "YES" if it's a palindrome and print "NO" otherwise

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a = int(input())
if (a//100 == a%10) and ((a//100) % 10 == (a//10)%10):
    print ('YES')
else :
    print ('NO')
```

1. INPUT :   **OUTPUT :**

```
4224        YES
```

2. INPUT : **OUTPUT :**

6789 NO

3. INPUT : **OUTPUT :**

2222 YES

INFERENCE :

QUESTION 8 :

Given the year number. You need to check if this year is a leap year. If it is, print LEAP, otherwise print COMMON. The rules in Gregorian calendar are as follows:

• a year is a leap year if its number is exactly divisible by 4 and is not exactly divisible by 100

• a year is always a leap year if its number is exactly divisible by 400

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a=int(input())
if a%4==0 and a%100!=0 or a%400==0:
    print(“LEAP”)
else:
    print(“COMMON”)
```

1. **INPUT :** **OUTPUT :**
   2012 LEAP

2. **INPUT :** **OUTPUT :**
   2011 COMMON

3. **INPUT :** **OUTPUT :**
   1492 LEAP

INFERENCE :

WEEK-4

QUESTION 1 :

Given an integer N, print all the numbers from 1 to N.

OBJECTIVE : Loops (For and While).

PROGRAM :

```
a = int(input())
for i in range(1,a+1):
    print (i)
```

1. **INPUT :** 10 — **OUTPUT :** 1 2 3 4 5 6 7 8 9 10

2. **INPUT :** 3 — **OUTPUT :** 1 2 3

3. **INPUT :** 5 — **OUTPUT :** 1 2 3 4

INFERENCE :

QUESTION 2 :

10 numbers are given in the input. Read them and print their sum. Use as few variables as you can.

OBJECTIVE : Loops (For and While).

PROGRAM :

```
n=10
result=0
for i in range(n):
    result+=int(input())
print(result)
```

1. INPUT : **OUTPUT :**

```
0                45
1
2
3
4
5
6
7
8
9
```

2. INPUT : **OUTPUT :**

```
1                10
1
1
1
1
1
1
1
1
1
```

3. INPUT : **OUTPUT :**

```
758              4899
483
893
393
293
292
292
485
828
182
```

INFERENCE :

QUESTION 3 :

For the given integer n calculate the value n!. Don’t use math module in this exercise.

OBJECTIVE : Loops (For and While).

PROGRAM :

```
n=int(input())

fact=1

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

   fact=fact*i

print(fact)
```

1. **INPUT :** 4 — **OUTPUT :** 24
2. **INPUT :** 1 — **OUTPUT :** 1
3. **INPUT :** 5 — **OUTPUT :** 120

INFERENCE :

QUESTION 4 :

A prime number is a natural number greater than 1 that has no positive divisors other than and itself. Given an integer N > 1, print PRIME if it’s prime and print COMPOSITE otherwise.

OBJECTIVE : Loops (For and While).

PROGRAM :

```
n = int(input())

for i in range (2,n):

    if(n%i ==0):
```

```
    print ("COMPOSITE")
  else :
    print("PRIME")
```

1. **INPUT :** 7 **OUTPUT :** PRIME

2. **INPUT :** 9 **OUTPUT :** COMPOSITE

3. **INPUT :** 13 **OUTPUT :** PRIME

INFERENCE :

QUESTION 5 :

A prime number is a natural number greater than 1 that has no positive divisors other than and itself. Given two integers A and B, print all prime numbers between them, inclusively

OBJECTIVE : Loops (For and While).

PROGRAM :

```
A= int(input())
B= int(input())
for n in range (A, B+1):
  if(n>1) :
    for i in range (2,n) :
      if(n%i)==0:
        break
    else :
        print(n)
```

1. **INPUT :** **OUTPUT :**

1	2
5	3
	5

2. INPUT :	**OUTPUT :**
10	11
12	

3. INPUT :	**OUTPUT :**
13	13
17	17

INFERENCE :

QUESTION 6 :

The Fibonacci sequence is defined as follows:

Φ0=0, ϕ1=1, ϕn=ϕn−1+ϕn−2.

Given a non-negative integer n, print the nth Fibonacci number ϕn.

OBJECTIVE : Loops (For and While).

PROGRAM :

```
n= int(input())
def Fib(n):
    if n==0:
        teturn 0
     elif n==1:
        return 1
    else:
        return Fib(n-1) + Fib(n-2)
```

```
Print(Fib(n))
```

1. INPUT :	**OUTPUT :**
6	8
2. INPUT :	**OUTPUT :**
2	1
3. INPUT :	**OUTPUT :**
4	3

INFERENCE :

WEEK-5

QUESTION 1 :

For a given integer N, print all the squares of integer numbers where the square is less than or equal to N, in ascending order.

OBJECTIVE : While loop

PROGRAM :

```
n=int(input())

i=1

while(i*i<=n):

    print(i*i,end=' ')

    i=i+1
```

1. **INPUT :** 50 — **OUTPUT :** 1 4 9 16 25 36 49
2. **INPUT :** 10 — **OUTPUT :** 1 4 9
3. **INPUT :** 100 — **OUTPUT :** 1 4 9 16 25 36 49 64 81 100

INFERENCE:

QUESTION 2 :

A sequence consists of integer numbers and ends with the number 0. Determine how many elements of this sequence are greater than their neighbours above.

OBJECTIVE : While loop

PROGRAM :

```
n3=0

n2=int(input())

n1=n2
```

```
while n2!= 0:
    if n2>n1:
       n3+= 1
    n1=n2
    n2=int(input())
print(n3)
```

1. INPUT :	**OUTPUT :**
1	2
7	
9	
0	
2. INPUT :	**OUTPUT :**
1	2
5	
2	
4	
3	
0	
3. INPUT :	**OUTPUT :**
1	1
1	
1	
5	
1	
0	

INFERENCE :

QUESTION 3 :

The Fibonacci sequence is defined as follows:
$\phi 0=0$, $\phi 1=1$, $\phi n=\phi n-1+\phi n-2$
Given a non-negative integer n, print the nth Fibonacci number ϕn.
This problem can also be solved with a for loop.

OBJECTIVE : While loop

PROGRAM :

```
n = int(input())
if n == 0:
    print(0)
else:
    a=0
    b=1
    for i in range(2, n + 1):
        a, b = b, a + b
    print(b)
```

1. INPUT : 6 — **OUTPUT :** 8

2. INPUT : 0 — **OUTPUT :** 0

3. INPUT : 20 — **OUTPUT :** 6765

INFERENCE :

QUESTION 4 :

Given a sequence of integer numbers ending with the number 0. Determine the length of the widest fragment where all the elements are equal to each other.

OBJECTIVE : While loop

PROGRAM :

```
n=1
max=1
```

```
a=int(input())

b=a

while b!=0:

    b,a=int(input()),b

    if b==a:

        n+=1

    else:

        if n> max:

            max=n

        n=1

print(max)
```

1. INPUT : **OUTPUT :**

```
1        2
7
7
9
1
0
```

2. INPUT : **OUTPUT :**

```
1        2
2
2
0
```

3. INPUT : **OUTPUT :**

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

INFERENCE :

WEEK-6

QUESTION 1 :

Write a function capitalize(lower_case_word) that takes the lower case word and returns the word with the first letter capitalized. Eg., print(capitalize('word')) should print the word Word.

Then, given a line of lowercase ASCII words (text separated by a single space), print it with the first letter of each word capitalized using the your own function capitalize().

In Python there is a function ord(character), which returns character code in the ASCII chart, and the function chr(code), which returns the character itself from the ASCII code. For example, ord('a') == 97, chr(97) == 'a'.

OBJECTIVE : Functions and Recursion

PROGRAM :

```
def capitalize(word):
    return chr(ord(word[0]) + ord('A') - ord('a')) + word[1:]
s = [str(s) for s in input().split()]
for i in range(len(s)):
    s[i] = capitalize(s[i])
print(' '.join([str(i) for i in s]))
```

1. **INPUT :** harry potter **OUTPUT :** Harry Potter

2. **INPUT :** all i see turns to brown **OUTPUT :** All I See Turns To Brown

3. **INPUT :** procrastination **OUTPUT :** Procrastination

INFERENCE :

QUESTION 2 :

Given a positive real number aa and a non-negative integer n. Calculate a^n without using loops, ** operator or the built in function math.pow(). Instead, use recursion and the relation $a^n=a \cdot a^{n-1}$. Print the result.

Form the function power(a,n).

OBJECTIVE : Functions and Recursion

PROGRAM :

```
def power(a,n):
    if n==0:
        return 1
    else:
        return a*power(a,n-1)
print(power(float(input()),int(input())))
```

1. INPUT :	**OUTPUT :**
2	8.0
3	
2. INPUT :	**OUTPUT :**
2	4.0
2	
3. INPUT :	**OUTPUT :**
2	32768.0
15	

INFERENCE :

QUESTION 3 :

Given a sequence of integers that end with a 0. Print the sequence in reverse order.

Don't use lists or other data structures. Use the *force* of recursion instead.

OBJECTIVE : Functions and Recursion

PROGRAM :

```
def rev():
   n=int(input())
   if n!=0:
      rev()
   print(n)
rev()
```

```
1. INPUT :             OUTPUT :
     1                 0
     2                 3
     3                 2
     0                 1
2. INPUT :             OUTPUT :
     1                 0
     0                 1
3. INPUT :             OUTPUT :
     8                 0
     7                 5
     2                 4
     3                 1
     1                 3
     4                 2
     5                 7
     0                 8
```

INFERENCE :

QUESTION 4 :

Given a non-negative integer n, print the nth Fibonacci number. Do this by writing a function fib(n) which takes the non-negative integer n and returns the nth Fibonacci number.

Don't use loops, use the *flair* of recursion instead. However, you should think about why the recursive method is much slower than using loops.

OBJECTIVE : Functions and Recursion

PROGRAM :

```
def fib(n):
   if n==1 or n==2:
       return 1
   else:
       return fib(n - 1) +fib(n - 2)
n=int(input())
print(fib(n))
```

1. INPUT : 6 — **OUTPUT :** 8

2. INPUT : 1 — **OUTPUT :** 1

3. INPUT : 8 — **OUTPUT :** 21

INFERENCE :

WEEK-7

QUESTION 1 :

Perform string slicing operations by displaying in various forms of outputs.

OBJECTIVE : Strings

PROGRAM :

```
a=input()
print(a[2])
print(a[-2])
print(a[0:5])

print(a[:-2])

print(a[::2])

print(a[1::2])

print(a[::-1])

print(a[::-2])

print(len(a))
```

1. **INPUT :** Hello

OUTPUT :

```
l
l

Hello

Hel
Hlo
el
olleH
olH
5
```

2. **INPUT :** abc

OUTPUT :

```
c
b
abc
a
ac
b
cba
ca
3
```

3. **INPUT :** program

OUTPUT :

```
o
a
progr
progr
porm
rga
margorp
mrop
7
```

INFERENCE :

QUESTION 2 :

Given a string consisting of words separated by spaces. Determine how many words it has. To solve the problem, use the method count.

OBJECTIVE : Strings

PROGRAM :

```
a=input()

print(a.count(' ')+1)
```

1. **INPUT :** **OUTPUT :**
   Hello world 2

2. **INPUT :**
   In the hole in the ground there lived a hobbit
   OUTPUT :
   10

3. **INPUT :** **OUTPUT :**

   One two three four five 5

INFERENCE :

QUESTION 3 :

Given a string consisting of exactly two words separated by a space. Print a new string with the first and second word positions swapped (the second word is printed first).

This task should not use loops and if.

OBJECTIVE : Strings

PROGRAM :

```
a=input()

y=a.index(' ')

print(a[y:],a[:y])
```

1. **INPUT :** Hello, world! **OUTPUT :** Hello, world!

2. **INPUT :** A B **OUTPUT :** B A

3. **INPUT :** One two **OUTPUT :** two One

INFERENCE :

QUESTION 4 :

Given a string in which the letter h occurs at least twice. Remove from that string the first and the last occurrence of the letter h, as well as all the characters between them. Given a string in which the letter h occurs at least two times, reverse the sequence of characters enclosed between the first and last appearances.

OBJECTIVE : Strings

PROGRAM :

```
a=input()

p=a.find('h')

q=a.rfind('h')

print(a[:p]+a[q+1:])

print(a[:p]+a[q:p:-1]+a[q:])
```

1. **INPUT :**
   In the hole in the ground there lived a hobbit
   OUTPUT :
   In tobbit
   In th a devil ereht dnuorg eht ni eloh ehobbit

2. **INPUT :** qwertyhasdfghzxcvb

   OUTPUT :
   ```
   qwertyzxcvb
   qwertyhgfdsahzxcvb
   ```

3. **INPUT :** asdfghhzxcvb

   OUTPUT :
   ```
   asdfgzxcvb
   asdfghhzxcvb
   ```

INFERENCE :

WEEK-8

QUESTION 1 :

Given a list of numbers, find and print all the list elements with an even index number. (i.e. A[0], A[2], A[4], ...).

OBJECTIVE : Lists

PROGRAM :

```
n=input().split()

for i in range(0,len(n),2):

    print(n[i],end=' ')
```

1. **INPUT :** 1 2 3 4 5

   OUTPUT :
   ```
   1 3 5
   ```

2. **INPUT :** 9 4 5 2 3

   OUTPUT :
   ```
   9 5 3
   ```

3. **INPUT :** 90 45 3 43

   OUTPUT :
   ```
   90 3
   ```

INFERENCE :

QUESTION 2 :

Given a list of numbers, find and print all the elements that are greater than the previous element.

OBJECTIVE : Lists

PROGRAM :

```
a = input().split()
for i in range(len(a)):
    a[i] = int(a[i])
for i in range(1, len(a)):
    if a[i] > a[i-1]:
        print(a[i], end=' ')
```

1. **INPUT :** 1 5 2 4 3 — **OUTPUT :** 5 4
2. **INPUT :** 1 5 1 5 1 — **OUTPUT :** 5 5
3. **INPUT :** 1 1 1 5 1 — **OUTPUT :** 5

INFERENCE :

QUESTION 3 :

Given a list of numbers, find and print the first adjacent elements which have the same sign. If there is no such pair, leave the output blank.

OBJECTIVE : Lists

PROGRAM :

```
a = input().split()
for i in range(len(a)):
    a[i] = int(a[i])
for i in range(1, len(a)):
    if a[i] * a[i-1] > 0:
        print(a[i-1], a[i], end=' ')
        break
```

1. INPUT : -1 2 3 -1 -2 — **OUTPUT :** 2 3

2. INPUT : 1 2 -3 -4 -5 — **OUTPUT :** 1 2

3. INPUT : 1 2 3 4 — **OUTPUT :** 1 2

INFERENCE :

QUESTION 4 :

Given a list of numbers. Determine the element in the list with the largest value. Print the value of the largest element and then the index number. If the highest element is not unique, print the index of the first instance.

OBJECTIVE : Lists

PROGRAM :

```
a = input().split()
for i in range(len(a)):
    a[i] = int(a[i])
max_ele= a[0]
max_i = 0
```

```
for i in range(1, len(a)):
    if a[i] > max_ele:
        max_i = i
        max_ele= a[i]
print(max_ele, max_i, end=' ')
```

1. **INPUT :** **OUTPUT :**

```
1 2 3 2 1        3 2
```

2. **INPUT :** **OUTPUT :**

```
1 3 2            3 1
```

3. **INPUT :** **OUTPUT :**

```
-1 -2 -3 -4 -5   -1 0
```

INFERENCE :

QUESTION 5 :

Given a list of numbers with all of its elements sorted in ascending order, determine and print the quantity of distinct elements in it.

OBJECTIVE : Lists

PROGRAM :

```
a = input().split()
for i in range(len(a)):
    a[i] = int(a[i])
c = 1
for i in range(1,len(a)):
    if a[i] != a[i-1]:
        c += 1
print(c)
```

1. INPUT :	**OUTPUT :**
1 2 2 3 3 3	3
2. INPUT :	**OUTPUT :**
1 2 3 4 5	5
3. INPUT :	**OUTPUT :**
1 1 2 2 2 3 4 5 6 7	7

INFERENCE :

QUESTION 6 :

Given a list of numbers, find and print the elements that appear in the list only once. The elements must be printed in the order in which they occur in the original list.

OBJECTIVE : Lists

PROGRAM :

```
a = [int(s) for s in input().split()]

for i in range(len(a)):

    for j in range(len(a)):

        if i != j and a[i] == a[j]:

            break

    else:

        print(a[i], end=' ')
```

1. INPUT :	**OUTPUT :**
1 2 2 3 3 3	1
2. INPUT :	**OUTPUT :**
4 2 9 3 2 9 7 8 4	3 7 8

3. **INPUT :** **OUTPUT :**

```
5 2 29 4 28 8                5 2 29 4 28 8
```

INFERENCE :

WEEK -9

QUESTION 1:

Given two numbers n and m. Create a two-dimensional array of size (n×m) and populate it with the characters "." and "*" in a checkerboard pattern. The top left corner should have the character ".".

OBJECTIVE: Two - dimensional lists(arrays)

PROGRAM:

```
n,m=[int(i)for i in input().split()]

a=[[int(j)for j in range(m)]for i in range(n)]

for i in range(n):

    for j in range(m):

        if(i+j)%2 == 0:

            a[i][j]='.'

        else:

            a[i][j]='*'

for i in range(n):

    for j in range(m):

        print(a[i][j],end=' ')

    print()
```

INPUT:

```
3 4
```

OUTPUT:

```
. * . *

* . * .

. * . *
```

INPUT:

```
6 8
```

OUTPUT:

```
. * . * . * . *
* . * . * . * .
. * . * . * . *
* . * . * . * .
. * . * . * . *
* . * . * . * .
```

INPUT:

```
1 6
```

OUTPUT:

```
. * . * . *
```

INFERENCE:

QUESTION 2:

Given two positive integers m and n, m lines of n elements, giving an m×n matrix A, followed by two non-negative integers i and j less than n, swap columns i and j of A and print the result. Write a function swap_columns(a, i, j) and call it to exchange the columns.

OBJECTIVE: Two - dimensional lists(arrays).

PROGRAM:

```
def swap_columns(a,i,j):
    for k in range(len(a)):
        a[k][i],a[k][j] = a[k][j],a[k][i]
n,m=[int(i) for i in input().split()]
a = [[int(j)for j in input().split() ]for i in range(n)]
i,j=[int(i)for i in input().split()]
swap_columns(a,i,j)
for i in range(n):
    for j in range(m):
        print(a[i][j],end = ' ')
    print()
```

INPUT:

```
3 4
11 12 13 14
21 22 23 24
31 32 33 34
0 1
```

OUTPUT:

```
12 11 13 14
22 21 23 24
32 31 33 34
```

INPUT:

```
2 2
1 2
3 4
0 1
```

OUTPUT:

```
2 1
4 3
```

INPUT:

```
1 5
1 2 3 4 5
1 4
```

OUTPUT:

```
1 5 3 4 2
```

INFERENCE:

QUESTION 3:

Given two positive integers m and n, m lines of n elements, giving an m×n matrix A, followed by one integer c, multiply every entry of the matrix by c and print the result.

OBJECTIVE: Two - dimensional lists(arrays)

PROGRAM:

```
m,n=[int(i)for i in input().split()]
a=[[int(j)for j in input().split()]for i in range(m)]
c=int(input())
for i in range(m):
    for j in range(n):
        print(a[i][j]*c,end=' ')
    print()
```

INPUT:

```
3 4
11 12 13 14
21 22 23 24
31 32 33 34
2
```

OUTPUT:

```
22 24 26 28
42 44 46 48
62 64 66 68
```

INPUT:

```
2 2
1 2
3 4
0
```

OUTPUT:

```
0 0
0 0
```

INPUT:

```
4 3
1 2 3
4 5 6
7 8 9
10 11 12
1
```

OUTPUT:

```
1 2 3
4 5 6
7 8 9
10 11 12
```

INFERENCE:

QUESTION 4:

Given three positive integers m, n and r, m lines of n elements, giving an m×n matrix A, and n lines of r elements, giving an n×r matrix B, form the product matrix AB, which is the m×r matrix whose (i,k) entry is the sum

A[i][1]∗B[1][k]+⋯+A[i][n]∗B[n][k]A[i][1]∗B[1][k]+⋯+A[i][n]∗B[n][k]

and print the result.

OBJECTIVE: Two - dimensional lists(arrays).

PROGRAM:

```
m,n,r=[int(i)for i in input().split()]

a=[[int(k)for k in input().split()]for i in range(m)]

b=[[int(k)for k in input().split()]for j in range(n)]

c=[[0]*r for i in range(m)]

for  i  in  range(m):

    for k in range(r):

        for j in range(n):

            c[i][k] += a[i][j] * b[j][k]

for i in range(m):

    for     j     in     range(r):

        print(c[i][j],end = ' ')

    print()
```

INPUT:

3 4 2

0 1 2 3

4 5 6 7

8 9 10 11

2 3

OUTPUT:

13 5

45 33

77 61

INPUT:

2 2 2

1 2

3 4

5 4

3 2

OUTPUT:

11 8

27 20

INPUT:

1 1 1

5

7

OUTPUT:

35

INFERENCE:

WEEK-10

QUESTION 1:

Take two inputs from the user one is to create a tuple and another one is an integer n. Write a program to print the tuple n times. Create another tuple with the user given elements and concatenate both the tuples and print the result.

OBJECTIVE: Tuples.

PROGRAM:

```
a=input().split()

b=tuple(a)

n=int(input())

print(b*n)

c=input().split()

d=tuple(c)

print(b+d)
```

INPUT:

```
apple banana cherry

2

kiwi grapes mango
```

OUTPUT:

```
(‘apple’, ‘banana’, ‘cherry’, ‘apple’, ‘banana’, ‘cherry’)

(‘apple’, ‘banana’, ‘cherry’, ‘kiwi’, ‘grapes’, ‘mango’)
```

INPUT:

```
lilly rose lotus

3

jasmine tulip daffodil
```

OUTPUT:

```
(‘lilly’, ‘rose’, ‘lotus’, ‘lilly’, ‘rose’, ‘lotus’, ‘lilly’, ‘rose’, ‘lotus’)
```

```
('lilly', 'rose', 'lotus', 'jasmine', 'tulip', 'daffodil')
```

INPUT:

```
dog cat bear
2
tiger lion deer
```

OUTPUT:

```
('dog', 'cat', 'bear', 'dog', 'cat', 'bear')
('dog', 'cat', 'bear', 'tiger', 'lion', 'deer')
```

INFERENCE:

QUESTION 2:

Write a program to add an element to a tuple based on the user-given value in a specific index, print the result. If the index is not in the range print the error message.

OBJECTIVE: Tuples.

PROGRAM:

```
a=input().split()
elem=input()
index=int(input())
if index in range(0,len(a)):
    b=a.insert(index,elem)
    c=" ".join([str(i) for i in a])
    print(tuple(a))
else:
    print("Error!! index is out of range")
```

INPUT:

apple banana cherry

kiwi

1

OUTPUT:

('apple', 'kiwi', 'banana', 'cherry')

INPUT:

lilly rose lotus

jasmine

2

OUTPUT:

('lilly', 'rose', 'jasmine', 'lotus')

INPUT:

dog cat bear

lion

1

OUTPUT:

('dog', 'lion', 'cat', 'bear')

INFERENCE:

QUESTION 3:

Create a tuple with the user given elements. Write a program to remove an element from the input tuple, print the result . If the element is not present in the tuple print the error message.

OBJECTIVE: Tuples.

PROGRAM:

```
a=input().split()

elem=input()

if elem in a:

    b=a.remove(elem)

    c=“ ”.join(str(i) for i in a )

    print(tuple(a))

else:

    print(“Error !! element not found)
```

INPUT:

```
apple banana cherry

banana
```

OUTPUT:

```
(‘apple’, ‘cherry’)
```

INPUT:

```
lilly rose lotus

rose
```

OUTPUT:

```
 (‘lilly’, ‘lotus’)
```

INPUT:

```
dog cat bear lion

bear
```

OUTPUT:

```
 ('dog', 'cat', 'lion')
```

INFERENCE:

QUESTION 4:

Create a tuple with the user given inputs. Take an index n from the user. Write a program to print the tuple element at index n, print the result. If the given index range is not in the tuple print the error message.

OBJECTIVE: Tuples.

PROGRAM:

```
a=input().split()
b=tuple(a)
n=int(input())
if n in range(0,len(a)):
      print(b[n])
else:
     print("Error!! Element not found")
```

INPUT:

```
apple banana cherry
1
```

OUTPUT:

```
Banana
```

INPUT:

```
lilly rose lotus
1
```

OUTPUT:

rose

INPUT:

dog cat bear lion

2

OUTPUT:

bear

INFERENCE:

WEEK-11

QUESTION 1:

The text is given in a single line. For each word of the text count the number of its occurrences before it. A word is a sequence of non-whitespace characters. Two consecutive words are separated by one or more spaces. Punctuation marks are a part of a word, by this definition.

OBJECTIVE: Dictionaries

PROGRAM:

```
A=input().split()
d={}
for key in A:
   d[key]=d.get(key,0)+1
   print(d[key]-1,end=' ')
```

INPUT:

```
one two one tho three
```

OUTPUT:

```
0 0 1 0 0
```

INPUT:

```
aba aba; aba @?"
```

OUTPUT:

```
0 0 1 0
```

INPUT:

```
you will not solve this test. This @problem@ is unsolvable.
```

OUTPUT:

```
0 0 0 0 0 0 0 0 0 0
```

INFERENCE:

QUESTION 2:

You are given a dictionary consisting of word pairs. Every word is a synonym the other word in its pair. All the words in the dictionary are different. After the dictionary there's one more word given. Find a synonym for him.

OBJECTIVE: Dictionaries

PROGRAM:

```
n = int(input())

d = {}

for i in range(n):

    key, value = input().split()

    d[key] = value

    d[value] = key

print(d[input()])
```

INPUT:

```
3

Hello Hi

Bye Goodbye

List Array

Goodbye
```

OUTPUT:

```
Bye
```

INPUT:

```
1

beep Car

Car
```

OUTPUT:

```
beep
```

INPUT:

```
10
95UUSA4P3K 140YAZQF1W2
JIGCEL82Y3BXJ 277WMID9GFAWGC
X36U1924TQ GUDPNJ60NR3A
TR7JOPXQ0JK64CM O6BJSY7MHMAI8S
17HZOQO20CDQ1UG VVIPAVTVY202R
8T1A0XRRKKXEV 7L2UKWCJ917G1Q
WLPOJYRN0G3MC 5QWJPM4FG58
Z8TN35IQAF GXUPQY7I3AJCX3
5F382242HD25Q 3IV9JL29DNP64
KZWNAOSV8H8APVF LDAIALO4YMXE7
WLPOJYRN0G3MC
```

OUTPUT:

```
5QWJPM4FG58
```

INFERENCE:

QUESTION 3:

Given the text: the first line contains the number of lines, then given the lines of words. Print the word in the text that occurs most often. If there are many such words, print the one that is less in the alphabetical order.

OBJECTIVE: Dictionaries.

PROGRAM:

```
n=int(input())
d={}
```

```
max = 0
for i in range(n):
    for word in input().split():
        if word in d:
            d[word] += 1
        else:
            d[word] = 1
        if d[word] > max:
            max = d[word]
for key, val in sorted(d.items()):
    if val == max:
        print(key)
        break
```

INPUT:

```
1
apple orange banana banana orange
```

OUTPUT:

```
banana
```

INPUT:

```
3
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
```

OUTPUT:

```
a
```

INPUT:

```
2
```

```
taia ikm ikm ikm taia taia taia
ikm ikm ikm
```

OUTPUT:

```
ikm
```

INFERENCE:

QUESTION 4:

Given a list of countries and cities of each country. Then given the names of the cities. For each city specify the country in which it is located.

OBJECTIVE: Dictionaries.

PROGRAM:

```
n=int(input())
d={}
for i in range(n):
   x=input().split()
   for key in x[1:]:
      d[key]=x[0]
m=int(input())
for  i in range(m):
   print(d[input()])
```

INPUT:

```
2
USA Boston Pittsburgh Washington Seattle
```

UK London Edinburgh Cardiff Belfast

3

Cardiff

Seattle

London

OUTPUT:

UK

USA

UK

INPUT:

2

Russia Moscow Petersburg Novgorod Kaluga

Ukraine Kiev Donetsk Odessa

3

Odessa

Moscow

Novgorod

OUTPUT:

Ukraine

Russia

Russia

INPUT:

5

A B

C D

E F

G H

```
I J
5
J
H
F
D
B
```

OUTPUT:

```
I
G
E
C
A
```

INFERENCE:

WEEK-12

QUESTION 1:

Write a program to print each line of a file in reverse order**.**

OBJECTIVE: File Handling.

PROGRAM:

```
f=open("c:\\abc\\third.txt","w")
with open("c:\\abc\\first.txt","r") as myfile:
   data=myfile.read()
f.write(data[::-1])
f.close()
```

INPUT: first.txt:

```
HAPPY NEWYEAR

GOOD BYE 2021

WELCOME 2022
```

OUTPUT: third.txt:

```
2202 EMOCLEW

1202 EYB DOOG

RAEYWEN YPPAH
```

INPUT: first.txt:

```
HELLO WORLD!

WELCOME TO PYTHON

INTERPRETER LANGUAGE
```

OUTPUT: third.txt:

```
EGAUGNAL RETERPRETNI

NOHTYP OT EMOCLEW

!DLROW OLLEH
```

INPUT: first.txt:

```
For a better tomorrow,

Plant more trees and make

this planet better

place to live in.
```

OUTPUT: third.txt:

```
.ni evil ot ecalp

retteb tenalp siht

ekam dna seert erom tnalP

,worromot retteb a roF
```

INFERENCE:

QUESTION 2:

Write a program to print the number of lines , words and characters in a file.

OBJECTIVE: File Handling.

PROGRAM:

```
f=open("c:\\abc\\first.txt","r")
l=0
w=0
c=0
for i in f:
    i=i.strip("\n")
    words=i.split()
    l+=1
    w+=len(words)
    c+=len(i)
f.close()
print(l,w,c)
```

INPUT:

first.txt:

```
HAPPY NEWYEAR

GOOD BYE 2021

WELCOME 2022
```

OUTPUT:

```
3 7 38
```

INPUT:

first.txt:

```
HELLO WORLD!

WELCOME TO PYTHON

INTERPRETER LANGUAGE
```

OUTPUT:

```
3 7 49
```

INPUT:

first.txt:

```
For a better tomorrow,

Plant more trees and make

this planet better

place to live in.
```

OUTPUT:

```
4 16 82
```

INFERENCE:

QUESTION 3:

Write a Python program to copy one file to another file.

OBJECTIVE: File Handling.

PROGRAM:

```
with open('C:\\abc\\first.txt','r') as firstfile, open('C:\\abc\\second.txt','w') as secondfile:
    for line in firstfile:
        secondfile.write(line)
```

INPUT:

first.txt:

```
HAPPY NEWYEAR

GOOD BYE 2021

WELCOME 2022
```

OUTPUT:

second.txt:

```
HAPPY NEWYEAR

GOOD BYE 2021

WELCOME 2022
```

INPUT:

first.txt:

```
HELLO WORLD!

WELCOME TO PYTHON

INTERPRETER LANGUAGE
```

OUTPUT:

second.txt:

```
HELLO WORLD!

WELCOME TO PYTHON

INTERPRETER LANGUAGE
```

INPUT:

first.txt:

```
For a better tomorrow,

Plant more trees and make

this planet better

place to live in.
```

OUTPUT:

second.txt:

```
For a better tomorrow,

Plant more trees and make

this planet better

place to live in.
```

INFERENCE:

WEEK-13

QUESTION 1 :

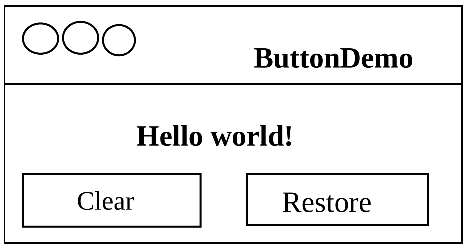
Write a Python GUI program using breezypythongui library, to demonstrate command button events. To clear and restore a message using two command buttons called clear and restore.

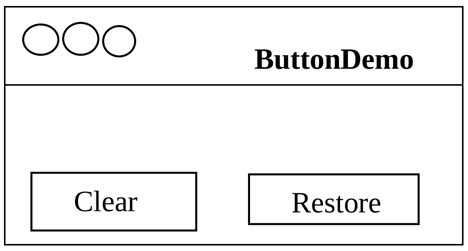
OBJECTIVE: GUI

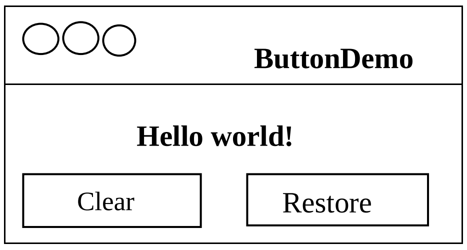
PROGRAM:

```
from breezypythongui import EasyFrame
class ButtonDemo(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self)
        self.label=self.addLabel(text="Hello world!",row=0,column=0,columnspan=2,sticky="NSEW")
        self.clearBtn=self.addButton(text="Clear",row=1,column=0,command=self.clear)
        self.restoreBtn=self.addButton(text="Restore",row=1,column=1,state="disabled",command=self.restore)
    def clear(self):
        self.label["text"]=""
        self.clearBtn["state"]="disabled"
        self.restoreBtn["state"]="normal"
    def restore(self):
        self.label["text"]="Hello world!"
        self.clearBtn["state"]="normal"
        self.restoreBtn["state"]="disabled"
def main():
    ButtonDemo().mainloop()
if __name__ == "__main__":
    main()
```

OUTPUT:







INFERENCE:

QUESTION 2 :

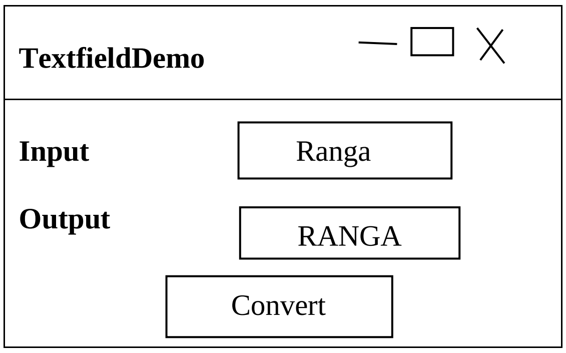
Write a Python GUI program using breezypythongui library, to demonstrate text fields.
Convert the given text to upper case

OBJECTIVE: GUI

PROGRAM:

```
from breezypythongui import EasyFrame
class TextFieldDemo(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self,title="Text Field Demo")
        self.addLabel(text="Input",row=0,column=0)
        self.inputField=self.addTextField(text="",row=0,column=1)
        self.addLabel(text="Output",row=1,column=0)
        self.outputField=self.addTextField(text="",row=1,column=1,state="readonly")
        self.addButton(text="Convert",row=2,column=0,columnspan=2,command=self.convert)
    def convert(self):
        text=self.inputField.getText()
        result=text.upper()
        self.outputField.setText(result)
def main():
    TextFieldDemo().mainloop()
if __name__ == "__main__":
    main()
```

OUTPUT 1:



INFERENCE:

QUESTION 3 :

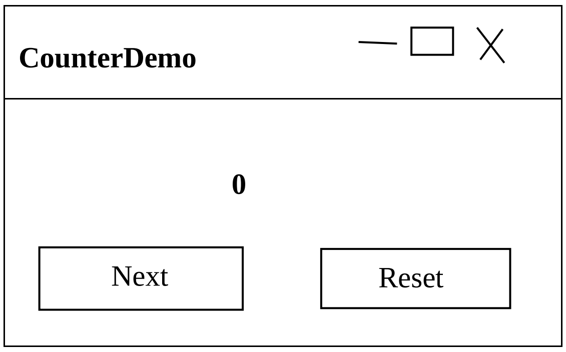
Write a Python GUI program using breezypythongui library, to demonstrate integer and float fields. Calculate the square root of the given number. Handle exceptions and display an error message if the user gives a negative number as input.

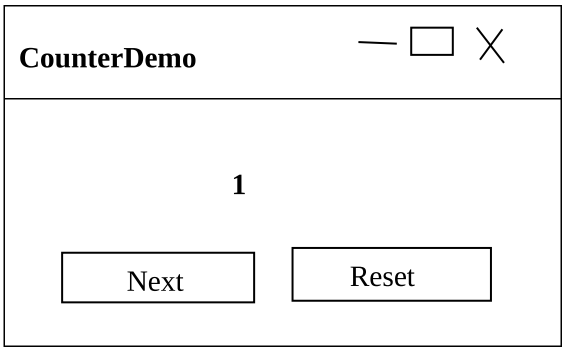
OBJECTIVE: GUI

PROGRAM:

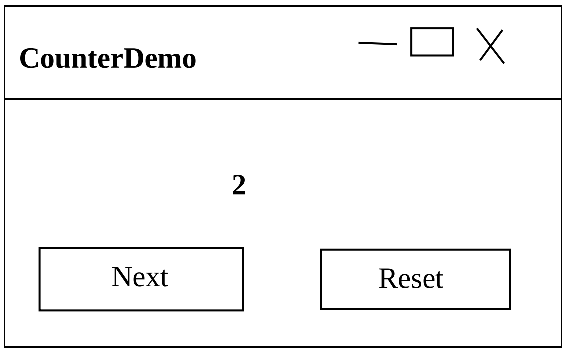
```
from breezypythongui import EasyFrame
import math
class NumberFieldDemo(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self,title="Number Field Demo")
        self.addLabel(text="An integer",row=0,column=0)
        self.inputField=self.addIntegerField(value=0,row=0,column=1,width=10)
        self.addLabel(text="Square root",row=1,column=0)
        self.outputField=self.addFloatField(value=0.0,row=1,column=1,width=8,precision=2,state="readonly")
        self.addButton(text="Compute",row=2,column=0,columnspan=2,command=self.computeSqrt)
    def computeSqrt(self):
        try:
          number=self.inputField.getNumber()
          result=math.sqrt(number)
          self.outputField.setNumber(result)
        except ValueError :
            self.messageBox(title="ERROR",message="Input must be an integer>=0")
def main():
    NumberFieldDemo().mainloop()
if __name__=="__main__":
    main()
```

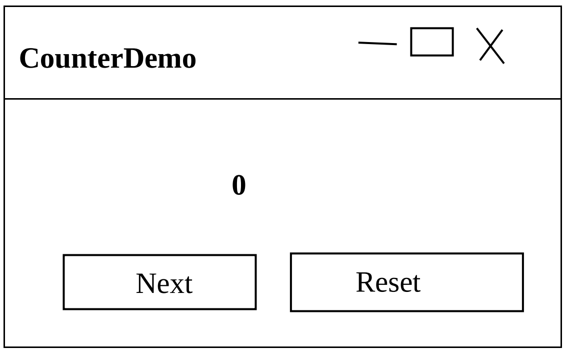
OUTPUT 1 :





Reset:





INFERENCE:

QUESTION 4 :

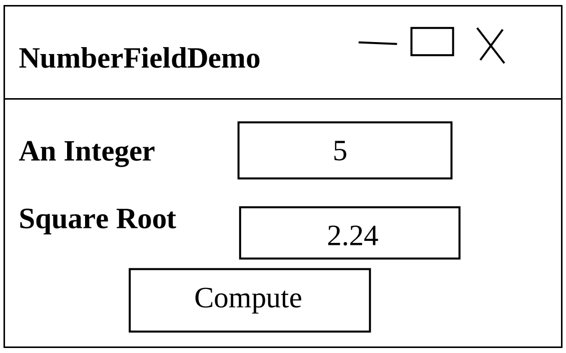
Write a Python GUI program using breezypythongui library, to demonstrate instance variable. Implement a counter increment program on a button event, and reset the counter to 0 when the user presses the reset button.

OBJECTIVE: GUI

PROGRAM:

```
from breezypythongui import EasyFrame
class CounterDemo(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self,title="CounterDemo")
        self.setSize(200,75)
        self.count=0
        self.label=self.addLabel(text="0",row=0,column=0,sticky="NSEW",columnspan=2)
        self.addButton(text="Next",row=1,column=0,command=self.next)
        self.addButton(text="Reset",row=1,column=1,command=self.reset)
    def next(self):
        self.count+=1
        self.label["text"]=str(self.count)
    def reset(self):
        self.count=0
        self.label["text"]=str(self.count)
def main():
    CounterDemo().mainloop()
if __name__=="__main__":
    main()
```

OUTPUT :



INFERENCE :

WEEK-1(PBL)

QUESTION 1 :

Write a program that reads an integer number and prints its previous and next numbers. There shouldn’t be a space before the period.

OBJECTIVE : To learn input, print, numbers.

PROGRAM :

```
a= int(input())

print(‘The next number for the number’ ,a, ‘is’ ,a+1 )

print(‘The previous number for the number’ ,a, ‘is’ ,a-1)
```

1. **INPUT :**
```
179
```
OUTPUT :
```
The next number for the number 179 is 180.
The previous number for the number 179 is 178.
```

2. **INPUT :**
```
0
```
OUTPUT :
```
The next number for the number 0 is 1.
The previous number for the number 0 is -1.
```

3. **INPUT :**
```
 100
```
OUTPUT :
```
The next number for the number 100 is 101.
The previous number for the number 100 is 99.
```

INFERENCE :

QUESTION 2 :

A school decided to replace the desks in three classrooms. Each desk sits two students.

Given the number of students in each class, print the smallest possible number of desks that can be purchased.

The program should read three integers: the number of students in each of the three classes, a, b and c respectively.

OBJECTIVE : To learn input, print, numbers.

PROGRAM :

```
a=int(input())
b=int(input())
c=int(input())
d=(a//2)+(a%2)
e=(b//2)+(b%2)
f=(c//2)+(c%2)
print(d+e+f)
```

1. INPUT : **OUTPUT :**

```
20              32
21
22
```

2. INPUT : **OUTPUT :**

```
1               3
1
1
```

3. INPUT : **OUTPUT :**

```
26              31
20
16
```

INFERENCE :

WEEK-2(PBL)

QUESTION 1 :

Given a four-digit integer number, perform its cyclic rotation by two digits, as shown in the tests below

OBJECTIVE : Integers, Floats.
PROGRAM :

```
n=int(input())

a=(n%100)*100

b=(n//100)

print (a+b)
```

```
1. INPUT :                OUTPUT :
     3456                  5634

2. INPUT :                OUTPUT :
     2615                  1526

3. INPUT :                OUTPUT :
     2513                 1325
```

INFERENCE :

QUESTION 2 :

Let’s count the days of the week as follows: 0 – Sunday, 1 – Monday, 2 – Tuesday, …, 6 –Saturday. Given an integer K in the range 1 to 365, find the number of the day of the week for the K-th day of the year provided that this year’s January 1 is Thursday.

OBJECTIVE : Integers, Floats

PROGRAM :

```
k = int(input())

if(k%7==1):

  print(‘Thursday’)

elif(k%7==2):

print(‘Friday’)
```

```
elif(k%7==3):
  print (‘Saturday’)
elif(k%7==4):
  print(‘Sunday’)
elif(k%7==5):
  print (‘Monday’)
elif(k%7==6):
  print(‘Tuesday’)
else:
  print (‘Wednesday’)
```

1. **INPUT :** 32 — **OUTPUT :** Sunday

2. **INPUT :** 44 — **OUTPUT :** Friday

3. **INPUT :** 62 — **OUTPUT :** Tuesday

INFERENCE :

QUESTION 3 :

Given the integer N – the number of minutes that is passed since midnight – how many hours and minutes are displayed on the 24h digital clock?

The program should print two numbers: the number of hours (between 0 and 23) and the number of minutes (between 0 and 59).

For example, if N = 150, then 150 minutes have passed since midnight – i.e. now is 2:30 am. So the program should print 2 30.

OBJECTIVE : Integers, Floats.

PROGRAM :

```
n = int( input())

x = n//60

y = n%60

print( x,y )
```

1. **INPUT :**   **OUTPUT :**
```
180                     3 0
```

2. **INPUT :**   **OUTPUT :**
```
444                     7 24
```

3. **INPUT :**   **OUTPUT :**
```
1439                    23 59
```

INFERENCE :

QUESTION 4 :

A cupcake costs A dollars and B cents. Determine, how many dollars and cents should one pay for N cupcakes. A program gets three numbers: A, B, N. It should print two numbers total cost in dollars and cents.

OBJECTIVE : Integers, Floats.

PROGRAM :

```
a = int(input())

b  =  int(input())

n = int(input())

cost=  n  *  (100  *  a  +  b)

print(cost // 100, cost % 100)
```

1. **INPUT :**   **OUTPUT :**
```
2                        10 0
```

50
4

2. **INPUT :** **OUTPUT :**

```
10        20 30
15
2
```

3. **INPUT :** **OUTPUT :**

```
3000      9002970  0
99
3000
```

INFERENCE :

QUESTION 5 :

A snail goes up A feet during the day and falls B feet at night. How long does it take him to go up H feet? Given three integer numbers H, A and B (A > B), the program should output a number of days.

OBJECTIVE : Integers, Floats.

PROGRAM :

```
H=int(input())

A=int(input())

B=int(input())

D=(H-B)/A-B

Print(D)
```

1. **INPUT :** **OUTPUT :**

```
30        28
3
2
```

2. **INPUT :**

```
15
2
1
```

OUTPUT :

```
14
```

3. **INPUT :**

```
65
10
5
```

OUTPUT :

```
12
```

INFERENCE :

<u>WEEK-3(PBL)</u>

QUESTION 1 :

A white chess pawn moves up vertically one square at a time. An exception is a pawn on a row #2: it can move either one or two squares up. In addition, a white chess pawn captures diagonally up one square to the left or right. A white chess pawn can never occur on a row #1. The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two – for the first square, and then the last two – for the second square. The program should print YES if a white pawn can possibly move from the first square to the second square in one move in some game – either by move or by capture. The program should print NO otherwise.

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
a=int(input())
b=int(input())
c=int(input())
d=int(input())
if(b!=1):
  if(b==2):
    if((a==c and d==b+1) or (a==c and d==b+2) or (c==a+1 and d==b+1) or (c==a-1 and d==b+1)) :
      print ('Yes')
    else:
      print('No')
  elif(b>2):
    if((a==c and d==b+1) or (c==a-1 and d==b+1) or (c==a+1 and d==b+1)):
      print ('Yes')
    else:
      print ('NO')
else:
  print ('Pawn cannot be in first row')
```

1. **INPUT :** **OUTPUT :**
```
4          YES
5
4
6
```
2. **INPUT :** **OUTPUT :**
```
2           NO
7
7
2
```
3. **INPUT :** **OUTPUT :**
```
2           YES
3
2
4
```

INFERENCE :

QUESTION 2:

Given a month – an integer from 1 (January) to 12 (December), print the number of days in it in the year 2017 (or any other non-leap year).

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
n=int(input())
if(n==2):
  print('28')
elif(n<8):
  if(n%2==0):
    print('30')
  else:
    print('31')
else:
  if(n%2==0):
    print ('31')
  else:
    print('30')
```

1. INPUT : 1	**OUTPUT :** 31
2. INPUT : 5	**OUTPUT :** 31
3. INPUT : 9	**OUTPUT :** 30

INFERENCE :

QUESTION 3 :

Given the month (an integer from 1 to 12) and the day in it (an integer from 1 to 31) in the year 2017 (or in any other common year), print the month and the day of the next day to it. The first test corresponds to March 30 and March 31. The second test corresponds to March 31 and April

OBJECTIVE : Conditions: if, then, else.

PROGRAM :

```
m=int(input())
if(m==2):
  days=28
elif(m<8):
  if(m%2==0):
    days=30
  else:
    days=31
else:
  if(m%2==0):
    days=31
  else:
    days=30
d=int(input()
if(d<days):
  d+=1
else:
  d=1
  m+=1
print ('The next day is',m,d)
```

1. **INPUT :** **OUTPUT :**

```
1            The next day is 1 26
25
```

2. **INPUT :** **OUTPUT :**

```
6            The next day is 6 14
13
```

3. **INPUT :** **OUTPUT :**

4 30	The next day is 5 1

INFERENCE :

WEEK-4(PBL)

QUESTION 1 :

There was a set of cards with numbers from 1 to N. One of the card is now lost. Determine the number on that lost card given the numbers for the remaining cards.

Given a number N, followed by N − 1 integers - representing the numbers on the remaining cards (distinct integers in the range from 1 to N). Find and print the number on the lost card.

OBJECTIVE : For loop

PROGRAM :

```
n=int(input())

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

for i in range(1,n):

   s-=int(input())

print(s)
```

1. INPUT : **OUTPUT :**

```
5        5
1
2
3
4
```

2. INPUT : **OUTPUT :**

```
4        1
3
2
4
```

3. INPUT : **OUTPUT :**

```
3        2
3
1
```

INFERENCE :

QUESTION 2 :

For given integer n ≤ 9 print a ladder of n steps. The k-th step consists of the integers from 1 to k without spaces between them. To do that, you can use the sep and end arguments for the function print().

OBJECTIVE : For loop

PROGRAM :

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

1. INPUT :	**OUTPUT :**
3	1 12 123
2. INPUT :	**OUTPUT :**
1	1
3. INPUT :	**OUTPUT :**
4	1 12 123 1234

INFERENCE :

WEEK-5(PBL)

QUESTION 1 :

For a given integer N, find the greatest integer x where 2^x is less than or equal to N. Print the exponent value and the result of the expression 2^x

Don't use the operation **

OBJECTIVE : While loop

PROGRAM :

```
n=int(input())

x=0

while(2**x<=n):

    x+=1

x=x-1

print(x,2**x)
```

1. INPUT : 50	**OUTPUT :** 5 32
2. INPUT : 10	**OUTPUT :** 3 8
3. INPUT : 1025	**OUTPUT :** 10 1024

INFERENCE :

QUESTION 2 :

As a future athlete you just started your practice for an upcoming event. Given that on the first day you run *x* miles, and by the event you must be able to run *y* miles.

Calculate the number of days required for you to finally reach the required distance for the event, if you increases your distance each day by 10% from the previous day.

Print one integer representing the number of days to reach the required distance.

OBJECTIVE : While loop

PROGRAM :

```
x = int(input())

y = int(input())

i = 1

while x < y:

    x *= 1.1

    i += 1

print(i)
```

1. **INPUT :** **OUTPUT :**
```
10          9
20
```
2. **INPUT :** **OUTPUT :**
```
10          13
30
```
3. **INPUT :** **OUTPUT :**
```
100         2
101
```

INFERENCE :

QUESTION 3 :

Given a sequence of non-negative integers, where each number is written in a separate line. Determine the length of the sequence, where the sequence ends when the integer is equal to 0. Print the length of the sequence (not counting the integer 0). The numbers following the number 0 should be omitted.

OBJECTIVE : While loop

PROGRAM :

```
i = 0
while int(input()) != 0:
   i += 1
print(i)
```

1. **INPUT :** **OUTPUT :**

```
1          3
7
9
0
5
```

2. **INPUT :** **OUTPUT :**

```
100        1
0
```

3. **INPUT :** **OUTPUT :**

```
1          7
2
3
4
5
6
7
0
1
2
3
```

INFERENCE :

QUESTION 4 :

A sequence consists of integer numbers and ends with the number 0. Determine the index of the largest element of the sequence. If the highest element is not unique, print the index of the first of them.

OBJECTIVE : While loop

PROGRAM :

```
a=int(input())
m=a
i=1
index=i
while a!=0:
    a=int(input())
    i+=1
    if m<a:
        m=a
        index=i
print(index)
```

1. INPUT : **OUTPUT :**

```
1        3
7
9
0
```

2. INPUT : **OUTPUT :**

```
1        3
2
3
0
```

3. INPUT : **OUTPUT :**

```
3        1
2
1
0
```

INFERENCE :

QUESTION 5 :

The Fibonacci sequence is defined as follows:

$\phi_0=0$, $\phi_1=1$, $\phi_n=\phi_{n-1}+\phi_{n-2}$.

Given an integer a, determine its index among the Fibonacci numbers, that is, print the number n such that $\phi_n=a$. If a is not a Fibonacci number, print -1.

OBJECTIVE : While loop

PROGRAM :

```
n = int(input())
f1 = 0
f2 = 1
if n==1:
   print(1)
else:
   i=1
   while n>f2:
      f2,f1=f1+f2,f2
      i += 1
   if n == f2:
      print(i)
   else:
      print(-1)
```

1. **INPUT :**	**OUTPUT :**
8	6

2. **INPUT :** 10 — **OUTPUT :** -1

3. **INPUT :** 55 — **OUTPUT :** 10

INFERENCE :

WEEK-6(PBL)

QUESTION 1 :

Given four real numbers representing cartesian coordinates: (x1,y1),(x2,y2). Write a function distance(x1, y1, x2, y2) to compute the distance between the points (x1,y1) and (x2,y2). Read four real numbers and print the resulting distance calculated by the function.
The formula for distance between two points can be found at Wolfram.

OBJECTIVE : Functions and Recursions

PROGRAM :

```
from math import sqrt

def distance(x1, y1, x2, y2):

   return sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2)

x1=float(input())

x2=float(input())

y1=float(input())

y2=float(input())

print(distance(x1,x2,y1,y2))
```

1. **INPUT :** — **OUTPUT :**

```
0                           1.4142135623730951
0
1
1
```

2. **INPUT :** **OUTPUT :**

```
0                           1
0
1
0
```

3. **INPUT :** **OUTPUT :**

```
3                           9.848857801796104
-2
-1
7
```

INFERENCE :

QUESTION 2:

Given a positive real number a and integer n. Compute a^n. Write a function power(a, n) to calculate the results using the function and print the result of the expression.

Don't use the same function from the standard library.

OBJECTIVE : Functions and Recursions

PROGRAM :

```
def power(a,n):
    res=1
    for i in range(abs(n)):
        res*=a
    if n>=0:
        return res
    else:
```

```
    return 1 / res
print(power(float(input()), int(input())))
```

1. **INPUT :** **OUTPUT :**

```
2        0.125
-3
```

2. **INPUT :** **OUTPUT :**

```
2        32768
15
```

3. **INPUT :** **OUTPUT :**

```
2        1
0
```

INFERENCE :

WEEK-7(PBL)

QUESTION 1 :

Given a string. Cut it into two "equal" parts (If the length of the string is odd, place the center character in the first string, so that the first string contains one more characther than the second). Now print a new string on a single row with the first and second halfs interchanged (second half first and the first half second)

Don't use the statement if in this task.

OBJECTIVE : Strings

PROGRAM :

```
a=input()
l=int(len(a)/2)
print(a[-l:]+a[:-l])
```

1. **INPUT :** **OUTPUT :**

Hi	iH

2. **INPUT :** Hello — **OUTPUT :** loHel

3. **INPUT :** asdfghj — **OUTPUT :** ghjasdf

INFERENCE :

QUESTION 2 :

Given a string that may or may not contain a letter of interest. Print the index location of the first and last appearance of f. If the letter f occurs only once, then output its index. If the letter f does not occur, then do not print anything.

Don't use loops in this task.

OBJECTIVE : Strings

PROGRAM :

```
a=input()
if a.count('f')==1:
   print(a.find('f'))
if a.count('f')>1:
   print(a.find('f'),a.rfind('f'))
```

1. **INPUT :** comfort — **OUTPUT :** 3

2. **INPUT :** Office — **OUTPUT :** 1 2

3. **INPUT :** — **OUTPUT :**

```
Hello                    -
```

INFERENCE :

QUESTION 3 :

Given a string that may or may not contain the letter of interest. Print the index location of the second occurrence of the letter f. If the string contains the letter f only once, then print -1, and if the string does not contain the letter f, then print -2.

OBJECTIVE : Strings

PROGRAM :

```
a=input()
if(a.count('f')==1):
    print(-1)
elif(a.count('f')==0):
    print(-2)
else:
    print(a.find('f',a.find('f')+1))
```

1. **INPUT :** comfort — **OUTPUT :** -1

```
comfort                  -1
```

2. **INPUT :** Coffee — **OUTPUT :** 3

```
Coffee                    3
```

3. **INPUT :** ofofofofofofofofo — **OUTPUT :** 3

```
ofofofofofofofofo        3
```

INFERENCE :

WEEK-8

QUESTION 1:

Given a list of numbers, find and print all elements that are an even number. In this case use a for-loop that iterates over the list, and not over its indices! That is, don’t use range().

OBJECTIVE: Lists.

PROGRAM:

```
a=input().split()
for i in a:
    i=int(i)
    if(i % 2 == 0):
        print(i,end=' ')
```

INPUT:

```
1 2 2 3 3 3 4
```

OUTPUT:

```
2 2 4
```

INPUT:

```
1 2 3 4 5
```

OUTPUT:

```
2 4
```

INPUT:

```
2 4 6 8
```

OUTPUT:

```
2 4 6 8
```

INFERENCE:

QUESTION 2:

Given a list of numbers, determine and print the quantity of elements that are greater than both of their neighbors.

The first and the last items of the list shouldn’t be considered because they don’t have two neighbors.

OBJECTIVE: Lists.

PROGRAM:

```
a=input().split()
c=0
for i in range(1,len(a)-1):
    if(a[i]>a[i-1] and a[i]>a[i+1]):
        c=c+1
print(c)
```

INPUT:

```
1 2 3 4 5
```

OUTPUT:

```
0
```

INPUT:

```
5 4 3 2 1
```

OUTPUT:

```
0
```

INPUT:

```
1 5 1 5 1
```

OUTPUT:

```
2
```

INFERENCE:

QUESTION 3:

Given a list of numbers, swap adjacent items in pairs (A[0] with A[1], A[2] with A[3], etc.). Print the resulting list. If a list has an odd number of elements, leave the last element in place.

OBJECTIVE: Lists.

PROGRAM:

```
a = [int(i) for i in input().split()]

for i in range(0,len(a)-1,2):

    a[i],a[i+1]=a[i+1],a[i]

print(' '.join([str(i) for i in a]))
```

INPUT:

```
1 2 3 4 5
```

OUTPUT:

```
2 1 4 3 5
```

INPUT:

```
4 5 3 4 2 3
```

OUTPUT:

```
5 4 4 3 3 2
```

INPUT:

```
9 4 5 2 3
```

OUTPUT:

```
4 9 2 5 3
```

INFERENCE:

QUESTION 4:

In chess it is known that it is possible to place 8 queens on an 8×8 chess board such that none of them can attack another. Given a placement of 8 queens on the board, determine if there is a pair of queens that can attach each other on the next move. Print the word NO if no queen can attack another, otherwise print YES. The input consists of eight coordinate pairs, one pair per line, with each pair giving the position of a queen on a standard chess board with rows and columns numbered starting at 1.

OBJECTIVE: Lists.

PROGRAM:

```
N = 8
result = 'NO'
x = [0] * N
y = [0] * N
for i in range(N):
    x[i], y[i] = [int(j) for j in input().split()]
for i in range (N):
    for j in range(i+1,N):
        if x[i] == x[j] or y[i] == y[j] or abs(x[i] - x[j]) == abs(y[i] - y[j]):
            result = 'YES'
print(result)
```

INPUT:

```
1 7
2 4
3 2
4 8
5 6
6 1
```

7 3

8 5

OUTPUT:

NO

INPUT:

1 8

2 7

3 6

4 5

5 4

6 3

7 2

8 1

OUTPUT:

YES

INPUT:

3 4

8 5

4 1

7 3

6 6

1 7

5 8

2 2

OUTPUT:

YES

INFERENCE:

QUESTION 5:

In bowling, the player starts with 10 pins at the far end of a lane. The object is to knock all the pins down. For this exercise, the number of pins and balls will vary. Given the number of pins N and then the number of balls K to be rolled, followed by K pairs of numbers (one for each ball rolled), determine which pins remain standing after all the balls have been rolled. The balls are numbered from 1 to N (inclusive) for this situation. The subsequent number pairs, one for each K represent the start to stop (inclusive) positions of the pins that were knocked down with each role. Print a sequence of N characters, where "I" represents a pin left standing and "." represents a pin knocked down.

OBJECTIVE: Lists.

PROGRAM:

```
a,b=[int(i) for i in input().split()]

c=[]

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

    c.append('I')

for j in range (b):

    x,y=[int(i) for i in input().split()]

    for k in range(x-1,y):

        c[k]='.'

d=''.join(str(i) for i in c)

print(d)
```

INPUT:

```
10 3

8 10

2 5

3 6
```

OUTPUT:

```
I.....I...
```

INPUT:

```
5 2

1 2

4 4
```

OUTPUT:

```
..I.I
```

INPUT:

```
10 3

3 5

4 6

10 10
```

OUTPUT:

```
II....III.
```

INFERENCE:

WEEK-9

QUESTION 1:

Given two integers representing the rows and columns (m×n), and subsequent m rows of n elements, find the index position of the maximum element and print two numbers representing the index (i×j) or the row number and the column number. If there exist multiple such elements in different rows, print the one with smaller row number. If there multiple such elements occur on the same row, output the smallest column number**.**

OBJECTIVE: Two-dimensional lists(arrays).

PROGRAM:

```
m,n=[int(i)for i in input().split()]

a=[[int(j)for j in input().split()]for i in range(m)]

max_i,max_j=0,0

max = a[0][0]

for i in range(m):

    for j in range(n):

        if a[i][j] > max:

            max = a[i][j]

            max_i,max_j=i,j

print(max_i,max_j)
```

INPUT:

```
3 4

0 3 2 4

2 3 5 5

5 1 2 3
```

OUTPUT:

```
1 2
```

INPUT:

```
1 1

1
```

OUTPUT:

```
0 0
```

INPUT:

```
1 6
1 2 3 3 2 1
```

OUTPUT:

```
0 2
```

INFERENCE:

QUESTION 2:

Given an odd number integer n, produce a two-dimensional array of size (n×n). Fill each element with a single character string of "." .Then fill the middle row, the middle column and the diagonals with the single character string of "*"(an image of a snow flake). Print the array elements in (n×n)rows and columns and separate the characters with a single space.

OBJECTIVE: Two-dimensional lists(arrays).

PROGRAM:

```
n=int(input())
a=[['.']*n for i in range(n)]
x=n//2
for i in range(n):
    a[x][i]='*'
    a[i][x]='*'
    a[i][i]='*'
    a[n-i-1][i]='*'
for i in range(n):
    for j in range(n):
        print(a[i][j],end=' ')
    print()
```

INPUT:

```
5
```

OUTPUT:

```
* . * . *
. * * * .
* * * * *
. * * * .
* . * . *
```

INPUT:

```
9
```

OUTPUT:

```
* . . . * . . . *
. * . . * . . * .
. . * . * . * . .
. . . * * * . . .
* * * * * * * * *
. . . * * * . . .
. . * . * . * . .
. * . . * . . * .
* . . . * . . . *
```

INPUT:

```
3
```

OUTPUT:

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

INFERENCE:

QUESTION 3:

Given an integer n, produce a two-dimensional array of size (n×n) and complete it according to the following rules, and print with a single space between characters:

- On the main diagonal write 0 .
- On the diagonals adjacent to the main, write 1 .
- On the next adjacent diagonals write 2 and so forth.

Print the elements of the resulting array.

OBJECTIVE: Two-dimensional lists(arrays).

PROGRAM:

```
n=int(input())
a=[[abs(i-j)for i in range(n)]for j in range(n)]
for i in range(n):
    for j in range(n):
        print(a[i][j],end=' ')
    print()
```

INPUT:

```
5
```

OUTPUT:

```
0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
```

```
3 2 1 0 1
4 3 2 1 0
```

INPUT:

```
6
```

OUTPUT:

```
0 1 2 3 4 5
1 0 1 2 3 4
2 1 0 1 2 3
3 2 1 0 1 2
4 3 2 1 0 1
5 4 3 2 1 0
```

INPUT:

```
4
```

OUTPUT:

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

INFERENCE:

QUESTION 4:

Given an integer n, create a two-dimensional array of size (n×n) and populate it as follows, with spaces between each character:

- The positions on the minor diagonal (from the upper right to the lower left corner) receive 1 .
- The positions above this diagonal recieve 0 .
- The positions below the diagonal receive 2 .

Print the elements of the resulting array.

OBJECTIVE: Two-dimensional lists(arrays).

PROGRAM:

```
n=int(input())
a=[[int(j)for j in range(n)]for i in range(n)]
for i in range(n):
    for j in range(n):
        a[i][j]=2
for i in range(n):
    for j in range(n-i-1):
        a[i][j]=0
for i in range(n):
    for j in range(n):
        a[n-i-1][i]=1


for i in range(n):
    for j in range(n):
        print(a[i][j],end=' ')
    print()
```

INPUT:

```
4
```

OUTPUT:

```
0 0 0 1
```

```
0 0 1 2
0 1 2 2
1 2 2 2
```

INPUT:

```
3
```

OUTPUT:

```
0 0 1
0 1 2
1 2 2
```

INPUT:

```
2
```

OUTPUT:

```
0 1
1 2
```

INFERENCE:

WEEK-10

QUESTION 1:

Take two inputs from the user one is to create a tuple and another one is an integer n. Write a program to print the tuple n times. Create another tuple with the user given elements and concatenate both the tuples and print the result.

OBJECTIVE: Tuples.

PROGRAM:

```
a=input().split()

b=tuple(a)

n=int(input())

print(b*n)

c=input().split()

d=tuple(c)

print(b+d)
```

INPUT:

```
apple banana cherry

2

kiwi grapes mango
```

OUTPUT:

```
(‘apple’, ‘banana’, ‘cherry’, ‘apple’, ‘banana’, ‘cherry’)

(‘apple’, ‘banana’, ‘cherry’, ‘kiwi’, ‘grapes’, ‘mango’)
```

INPUT:

```
lilly rose lotus

3

jasmine tulip daffodil
```

OUTPUT:

```
(‘lilly’, ‘rose’, ‘lotus’, ‘lilly’, ‘rose’, ‘lotus’, ‘lilly’, ‘rose’, ‘lotus’)

(‘lilly’, ‘rose’, ‘lotus’, ‘jasmine’, ‘tulip’, ‘daffodil’)
```

INPUT:

```
dog cat bear
2
tiger lion deer
```

OUTPUT:

```
(‘dog’, ‘cat’, ‘bear’, ‘dog’, ‘cat’, ‘bear’)
(‘dog’, ‘cat’, ‘bear’, ‘tiger’, ‘lion’, ‘deer’)
```

INFERENCE:

QUESTION 2:

Take two integers start index and end index as input from the console using input() function. Write a program to find the tuple elements within start index and end index, print the result**.**

OBJECTIVE: Tuples.

PROGRAM:

```
n=input().split()
m=tuple(n)
a=int(input())
b=int(input())
print(m[a:b])
```

INPUT:

```
apple banana cherry grapes papaya kiwi
1
4
```

OUTPUT:

(‘banana’, ‘cherry’, ‘grapes’)

INPUT:

lilly rose tulip daffodil jasmine

0

3

OUTPUT:

(‘lilly’, ‘ rose’, ‘ tulip’)

INPUT:

dog cat bear deer bella koala

1

3

OUTPUT:

(‘cat’, ‘bear’)

INFERENCE:

WEEK-11

QUESTION 1:

As you know, the president of USA is elected not by direct vote, but through a two-step voting. First elections are held in each state and determine the winner of elections in that state. Thereafter, the state election is going: in this election, every state has a certain the number of votes — the number of electors from that state. In practice, all the electors from the state of voted in accordance with the results of the vote within a state. The first line contains the number of records. After that, each entry contains the name of the candidate and the number of votes they got in one of the states. Count the total results of the elections: sum the number of votes for each candidate. Print candidates in the alphabetical order.

OBJECTIVE: Dictionaries.

PROGRAM:

```
n=int(input())

d={}

for i in range(n):

    key,value=input().split()

    if key in d:

        d[key] += int(value)

    else:

        d[key]=int(value)

for key,value in sorted(d.items()):

    print(key,value)
```

INPUT:

```
5

McCain 10

McCain 5

Obama 9

Obama 8

McCain 1
```

OUTPUT:

```
McCain 16
```

Obama 17

INPUT:

1

Obama 1

OUTPUT:

Obama 1

INPUT:

7

ivan 2

gena 1

sergey 100000

ivan 1

ivan 1

ivan 0

gena 100

OUTPUT:

gena 101

ivan 4

sergey 100000

INFERENCE:

QUESTION 2:

The virus attacked the filesystem of the supercomputer and broke the control of access rights to the files. For each file there is a known set of operations which may be applied to it:

- write W,
- read R,
- execute X.

The first line contains the number N — the number of files contained in the filesystem. The following N lines contain the file names and allowed operations with them, separated by spaces. The next line contains an integer M — the number of operations to the files. In the last M lines specify the operations that are requested for files. One file can be requested many times.

You need to recover the control over the access rights to the files. For each request your program should return OK if the requested operation is valid or Access denied if the operation is invalid.

OBJECTIVE: Dictionaries.

PROGRAM:

```
n=int(input())
d={}
for i in range(n):
    m=input().split()
    d[m[0]]=set(m[1:])
for i in range(int(input())):
    key,value=input().split()
    if(key == 'read'):
        key    =    'R'
    if(key == 'write'):
        key = 'W'
    if(key == 'execute'):
        key = 'X'
    if key in d[value]:
        print('OK')
    else:
        print("Access denied")
```

INPUT:

```
4
helloworld.exe R X
pinglog W R
nya R
goodluck X W R
5
read nya
write helloworld.exe
execute nya
read pinglog
write pinglog
```

OUTPUT:

```
OK
Access denied
Access denied
OK
OK
```

INPUT:

```
1
tmp_909925047 W X R
7
execute tmp_909925047
read tmp_909925047
write tmp_909925047
read tmp_909925047
execute tmp_909925047
execute tmp_909925047
read tmp_909925047
```

OUTPUT:

```
OK
OK
OK
OK
OK
OK
OK
```

INPUT:

```
2
tmp_584361681 R X
tmp_70361076 X
3
read tmp_70361076
write tmp_70361076
write tmp_70361076
```

OUTPUT:

```
Access   denied
Access   denied
Access   denied
```

INFERENCE:

QUESTION 3:

Given a number n, followed by n lines of text, print all words encountered in the text, one per line. The words should be sorted in descending order according to their number of occurrences in the text, and all words with the same frequency should be printed in lexicographical order.
Hint. After you create a dictionary of the words and their frequencies, you would like to sort it according to the frequencies. This can be achieved if you create a list whose elements are tuples of two elements: the frequency of occurrence of a word and the word itself. For example, [(2, ‘hi’), (1, ‘what’), (3, ‘is’)]. Then the standard list sort will sort a list of tuples, with the tuples compared by the first element, and if these are equal, by the second element. This is nearly what is required in the problem.

OBJECTIVE: Dictionaries.

PROGRAM:

```
n=int(input())
d={}
for i in range(n):
    m=input().split()
    for p in m:
        d[p]=d.get(p,0)+1
x=[(-d,p) for (p,d) in d.items()]
for key,value in sorted(x):
    print(value)
```

INPUT:

```
9
hi
hi
what is your name
my name is bond
james bond
my name is damme
van damme
claude van damme
```

jean claude van damme

OUTPUT:

damme

is

name

van

bond

claude

hi

my

james

jean

what

your

INPUT:

1

ai ai ai ai ai ai ai ai ai ai

OUTPUT:

ai

INPUT:

2

iovjxotfvt h h iovjxotfvt h iovjxotfvt iovjxotfvt h

h iovjxotfvt

OUTPUT:

h

iovjxotfvt

INFERENCE:

QUESTION 4:

One day, going through old books in the attic, a student Bob found English-Latin dictionary. By that time he spoke English fluently, and his dream was to learn Latin. So finding the dictionary was just in time.

Unfortunately, full-fledged language studying process requires also another type of dictionary: Latin-English. For lack of a better way he decided to make a second dictionary using the first one.

As you know, the dictionary consists of words, each of which contains multiple translations. For each Latin word that appears anywhere in the dictionary, Bob has to find all its translations (that is, all English words, for which our Latin word is among its translations), and take them and only them as the translations of this Latin word. Help him to create a Latin-English.

The first line contains a single integer N — the number of English words in the dictionary. Followed by N dictionary entries. Each entry is contained on a separate line, which contains first the English word, then a hyphen surrounded by spaces and then comma-separated list with the translations of this English word in Latin. All the words consist only of lowercase English letters. The translations are sorted in lexicographical order. The order of English words in the dictionary is also lexicographic. Print the corresponding Latin-English dictionary in the same format. In particular, the first word line should be the lexicographically minimal translation of the Latin word, then second in that order, etc. Inside the line the English words should be sorted also lexicographically.

OBJECTIVE: Dictionaries.

PROGRAM:

```
n=int(input())

d={}

for i in range(n):

    x,y=input().split(' - ')

    z=y.split(', ')

    for key in z:

        if key not in d:
            d[key]=[]
        d[key].append(x)
print(len(d))
for key,value in sorted (d.items()):
    print(key + ' - ' + ', '.join(value))
```

INPUT:

```
3
apple - malum, pomum, popula
fruit - baca, bacca, popum
punishment - malum, multa
```

OUTPUT:

```
7
baca - fruit
bacca - fruit
malum - apple, punishment
multa - punishment
pomum - apple
popula - apple
popum - fruit
```

INPUT:

```
1
school - schola
```

OUTPUT:

```
1
schola - school
```

INPUT:

```
3
greet - empfangen, willkommen
```

```
silicon - silicon
welcome - willkommen
```

OUTPUT:

```
3
empfangen - greet
silicon - silicon
willkommen - greet, welcome
```

INFERENCE: