Week-2, Graded, Programming

Problem-1

Question

Accept three positive integers as input from the user and check if they form the sides of a right triangle. Print `YES` if they form one, and `NO` if they do not.

Input-Output

Specification

The input will have three lines, with one integer on each line.

The output will be a single line containing one of these two strings: `YES` or `NO`.

Examples

Input-1

```
1
2
3
```

Output-1

```
NO
```

Input-2

```
3
4
5
```

Output-2

```
YES
```

Answer

```python
x = int(input())
y = int(input())
z = int(input())
if x >= y and x >= z:
    z, x = x, z
elif y >= x and y >= z:
    z, y = y, z
if x ** 2 + y ** 2 == z ** 2:
    print('YES')
else:
    print('NO')
```

Test cases

Public

Input-1

```
1
2
3
```

Output-1

```
NO
```

Input-2

```
3
4
5
```

Output-2

```
YES
```

Private

Input-1

```
8
1
5
```

Output-1

```
NO
```

Input-2

```
12
13
5
```

Output-2

```
YES
```

Input-3

```
80
39
89
```

Output-3

```
YES
```

Input-4

```
32
60
68
```

Output-4

```
YES
```

Input-5

```
2
1
8
```

Output-5

```
NO
```

Solution

```
x = int(input())
y = int(input())
z = int(input())
if x >= y and x >= z:
    z, x = x, z
elif y >= x and y >= z:
    z, y = y, z
if x ** 2 + y ** 2 == z ** 2:
    print('YES')
else:
    print('NO')
```

The basic idea of the solution is as follows: in a Pythagorean triplet, the hypotenuse will always be greater than the other two sides. So, we extract the maximum among the three numbers and see if the sum of squares of the other two numbers is equal to the square of this maximum. One way to implement this is to enforce `z` to hold the maximum value.

If `z` is already the maximum, we don't have to do anything. If not, then we need to check which among `x` and `y` is the maximum, and swap `z` with this variable which holds the maximum. For example, in line-4, we are checking if `x` is the maximum. If it is, then we swap `x` and `z`. Likewise, line-6 checks if `y` is the maximum. If it is, then we swap `y` and `z`. If the conditions in lines 4 and 6 both evaluate to `False`, then we know that `z` is the maximum by default.

At this stage, `z` holds the maximum among the three numbers. Finally, in lines 8 to 11, we check if the three numbers form a Pythagorean triplet or not.

Problem-2

Question

Accept an integer between 0 and 23 (both endpoints included) from the user and print what time of the day it is. Use the following table for reference. If the time is outside this range, print the string INVALID.

Input	Output
$T < 0$	INVALID
$0 \leq T \leq 5$	NIGHT
$6 \leq T \leq 11$	MORNING
$12 \leq T \leq 17$	AFTERNOON
$18 \leq T \leq 23$	EVENING
$T \geq 24$	INVALID

Input-Output

Specification

The input will be a single line containing an integer.

The output will be one of these strings: NIGHT, MORNING, AFTERNOON, EVENING

Examples

Input-1

```
3
```

Output-1

```
NIGHT
```

Input-1

```
14
```

Output-2

```
AFTERNOON
```

Answer

```python
t = int(input())
if 0 <= t <= 5:
    print('NIGHT')
elif 6 <= t <= 11:
    print('MORNING')
elif 12 <= t <= 17:
    print('AFTERNOON')
elif 18 <= t <= 23:
    print('EVENING')
else:
    print('INVALID')
```

Test Cases

Public

Input	Output
3	`NIGHT`
14	`AFTERNOON`

Private

Input	Output
9	`MORNING`
24	`INVALID`
12	`AFTERNOON`
-1	`INVALID`
19	`EVENING`

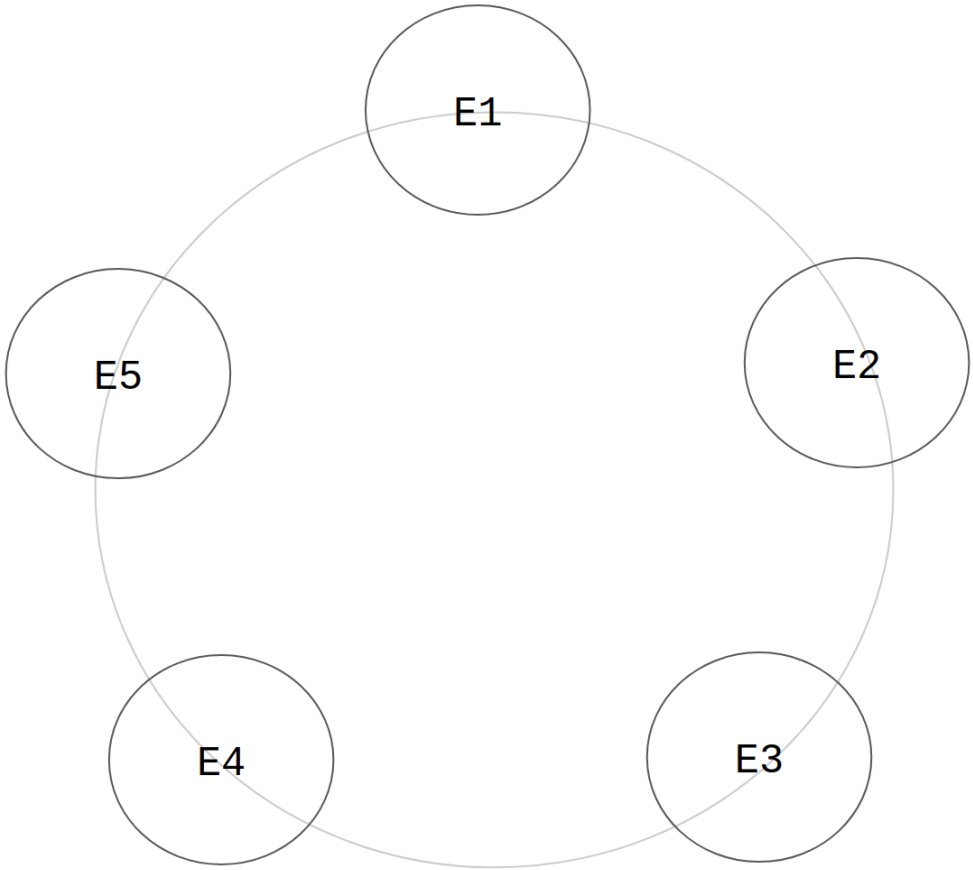
Solution

It is a simple `if-elif-else` ladder that checks for these conditions.

Problem-3

Question

`Even0dd` is a tech startup. Each employee at the startup is given an employee id which is a unique positive integer. On one warm Sunday evening, five employees of the company come together for a meeting and sit at a circular table:



The employees follow a strange convention. They will continue the meeting only if the following condition is satisfied.

> The sum of the employee-ids of any two adjacent employees at the table must be an even number.

They are so lazy that they won't move around to satisfy the above condition. If the current seating plan doesn't satisfy the condition, the meeting will be canceled. You are given the employee-id of all five employees. Your task is to decide if the meeting happened or not.

Input-Output

Specification

The input will be five lines, each line containing an integer. The i^{th} line will have the employee-id of Ei.

The output will be a single line containing one of these two strings:

- `YES`
- `NO`

Examples

Input

```
1
2
3
4
5
```

Output

```
NO
```

Input

```
2
4
6
8
10
```

Output

```
YES
```

Answer

```python
e1 = int(input())
e2 = int(input())
e3 = int(input())
e4 = int(input())
e5 = int(input())
if (e1 + e2) % 2 != 0:
    print('NO')
elif (e2 + e3) % 2 != 0:
    print('NO')
elif (e3 + e4) % 2 != 0:
    print('NO')
elif (e4 + e5) % 2 != 0:
    print('NO')
elif (e5 + e1) % 2 != 0:
    print('NO')
else:
    print('YES')
```

Answer-3

```
e1, e2, e3, e4, e5 = int(input()), int(input()), int(input()), int(input()), int(input())
if (e1 + e2) % 2 == (e2 + e3) % 2 == (e3 + e4) % 2 == (e4 + e5) % 2 == 0:
   print("YES")
else:
   print("NO")
```

Test Cases

Public

Input-1

```
1
2
3
4
5
```

Output-1

```
NO
```

Input-2

```
2
4
6
8
10
```

Output-2

```
YES
```

Private

Input-1

```
1
101
3
303
5
```

Output-1

```
YES
```

Input-2

```
60
98
1738
52
48
```

Output-2

```
YES
```

Input-3

```
1
10
3
30
5
```

Output-3

```
NO
```

Input-4

```
1
11
111
1111
11110
```

Output-4

```
NO
```

Solution

If we pick any two adjacent employees, their ids should sum to an even number. For example, if there are five employees, then we have to consider the following pairs:

- 1 and 2
- 2 and 3
- 3 and 4
- 4 and 5
- 5 and 1

If `e_i` and `e_j` are the ids of two adjacent employees, the conditional expression we have to evaluate is:

```
(e_i + e_j) % 2 == 0
```

The use of the term `any` resulted in some confusion. But the first public test case should have given you the idea that every pair of adjacent employee ids should sum to an even number. It is not enough if "some" pair of adjacent employee ids sums to an even number.

Problem-4

Question

Write a program to find which vowels are present in the input string. Print the vowels in lexical / dictionary order.

Input-1

```
"Moon flowers bloom only at night, closing during the day."
```

Output-1

```
aeiou
```

Input-2

```
"1 Yard (yd) = 3 feet (ft)."
```

Output-2

```
ae
```

Answer

```
input_string = input().lower()
vowels = ""
if "a" in input_string:
    vowels += "a"
if "e" in input_string:
    vowels += "e"
if "i" in input_string:
    vowels += "i"
if "o" in input_string:
    vowels += "o"
if "u" in input_string:
    vowels += "u"
print(vowels)
```

Test Cases

Public

Input-1

```
"Moon flowers bloom only at night, closing during the day."
```

Output-1

```
aeiou
```

Input-2

```
"1 Yard (yd) = 3 feet (ft)."
```

Output-2

```
ae
```

Private

Input-1

```
Education
```

Output-1

```
aeiou
```

Input-2

```
Summer is coming
```

Output-2

```
eiou
```

Input-3

```
Python politely said hi to JAVA
```

Output-3

```
aeio
```

Input-4

```
bcd xyz pqrs
```

Output-4

```

```

Solution

```
1  input_string = input().lower()
2  vowels = ""
3  if "a" in input_string:
4      vowels += "a"
5  if "e" in input_string:
6      vowels += "e"
7  if "i" in input_string:
8      vowels += "i"
9  if "o" in input_string:
10     vowels += "o"
11 if "u" in input_string:
12     vowels += "u"
13 print(vowels)
```

In line-1, we convert the string to lower case. We start with an empty string named `vowels` in line-2. Every time a vowel is found, we concatenate it to the end of `vowels`. The order in which the vowels are checked is: `a, e, i, o, u`. Since we are checking them in alphabetical order, the output will also be in alphabetical order.

Problem-5

Question

You are given the date of birth of two persons, not necessarily from the same family. The task is to find the younger of the two. If both of them share the same date of birth, then print the person whose name comes first in alphabetical order. If both the names starts with the same letter then compare the next letter of the names and so on.

Input-Output

Specification

The input will have four lines. The first two lines correspond to the first person, while the second two lines correspond to the second person. The first line for both the persons contains the name and the second line contains the date of birth in "DD-MM-YYYY" format.

The output will be the name of the younger of the two.

Examples

Input-1

```
Harsh
10-03-1990
Sparsh
18-12-1987
```

Output-1

```
Harsh
```

Input-2

```
Harsh
18-01-2000
Sparsh
18-03-2000
```

Output-2

```
Sparsh
```

Answer

```
c1_name = input()   # name of the first person
c1_dob = input()    # dob of the first person
c2_name = input()   # name of the second person
c2_dob = input()    # dob of the second person

# extract the day, month and year from the date of birth
# we can do this by slicing the dob string
#### person-1
```

```
c1_dob_day, c1_dob_month, c1_dob_year = int(c1_dob[: 2]), int(c1_dob[3 : 5]), int(c1_dob[6: ])
#### person-2
c2_dob_day, c2_dob_month, c2_dob_year = int(c2_dob[: 2]), int(c2_dob[3 : 5]), int(c2_dob[6: ])

younger = ''    # variable to store name of younger person
# first check for year
if c1_dob_year > c2_dob_year:
    younger = c1_name   # clearly, c1 was born after c2
elif c2_dob_year > c1_dob_year:
    younger = c2_name   # clearly, c2 was born after c1
# they share the year of birth
else:
    # check for month; same logic applies
    if c1_dob_month > c2_dob_month:
        younger = c1_name
    elif c2_dob_month > c1_dob_month:
        younger = c2_name
    # they share the month of birth; same logic applies
    else:
        # check for day
        if c1_dob_day > c2_dob_day:
            younger = c1_name
        elif c2_dob_day > c1_dob_day:
            younger = c2_name
        # if all the above conditions fail
        # then they were born on the same day
        # this is the moment we have been waiting for
        # check for alphabetical order
        else:
            if c1_name < c2_name:
                younger = c1_name   # c1 comes before c2 in alphabetical order
            else:
                younger = c2_name   # c2 comes before c1 in aplhabetical order
print(younger)
```

Answer-2

```
name1 = input()
dob1 = input()
name2 = input()
dob2 = input()

# changing the dob to YYYY-MM-DD format
dob1 = dob1[6:]+'-'+dob1[3:5]+'-'+dob1[:2]
dob2 = dob2[6:]+'-'+dob2[3:5]+'-'+dob2[:2]

if dob1 > dob2:
    print(name1)
elif dob1 < dob2:
    print(name2)
else:
    if name1 < name2:
        print(name1)
```

```
    else:
        print(name2)

```

Test Cases

Public

Input-1

```
Harsh
10-03-1990
Sparsh
18-12-1987
```

Output-1

```
Harsh
```

Input-2

```
Harsh
18-01-2000
Sparsh
18-03-2000
```

Output-2

```
Sparsh
```

Private

Input-1

```
Neha
01-01-2001
Priya
01-01-2001
```

Output-1

```
Neha
```

Input-2

```
David
29-02-2004
Jerold
28-02-2004
```

Output-2

```
David
```

Input-3

```
Abdul
05-07-2001
Raja
05-09-2001
```

Output-3

```
Raja
```

Input-4

```
Srivatsan
31-01-1932
Guru
19-11-2001
```

Output-4

```
Guru
```

Solution

Please check the comments associated with the code in answer-1.

Problem-6

Question

Write a python to validate the password given by the user based on the following conditions

1. It should be at least 8 and at most 32 characters
2. It should start with an uppercase or lowercase alphabet
3. It can contain numbers
4. It can contain the characters `!`, `@`, `#`, `$`, `%`, `^`, `&`, `_`, `*` and `.`
5. It should not have the characters `/`, `\`, `=`, `'`, `"` and spaces

Answer

```
pw, valid = input(), False  # initialize valid to False
# first condition is being checked here
if 8 <= len(pw) <= 32:
    # second condition is being checked here
    if 'a' <= pw[0] <= 'z' or 'A' <= pw[0] <= 'Z':
        # fifth condition is being checked here
        if not('/' in pw or '\\' in pw or '=' in pw or "'" in pw or '\"' in pw or ' ' in pw):
            valid = True
print(valid)
# Note that we are not checking for conditions 3 and 4
# This is becuase they are positive conditions
# They talk about what the password can contain
# Conditions 1, 2 and 5 are constraints
# They tell us what should not be there or what is not allowed
# So, it is these that we must be concerned with
```

Test Cases

Public

Input	Output
abcd1234	True
pass/word	False

Private

Input	Output
no_password	True
pythonIsSlowerInCompilationThanCPlusPlusandJava	False
rajesh'sBikeisBlack	False
never say never	False
giveA"Quote"	False

Solution

Check comments in answer.