Day 4: Logical Operators + Class vs. Instance!

Problem Statement

Welcome to Day 4! Check out a video review of logical operations here, or just jump right into the problem.

Note: This task is focused on Object Oriented concepts, so it is only enabled in a few languages.

You will create a class *Person* and write a constructor that takes an integer, *initial_Age*. In this constructor, you should check that the *initial_Age* is not negative because we can't have people with negative ages.

If the initial_Age is negative, set the instance's *age* equal to zero then print

```
"This person is not valid, setting age to 0."
```

without the quotations..

Inside of this class, you will also create an instance variable called *age* and if initial_Age is not negative, then you will set age to equal the value of initial_Age. In addition, you will write an instance method, *amIOld()*, that prints whether people are old or not to the console.

In **amIOld()**, do the following things:

- If the **age** of the Person instance is less than 13, then print `"You are young."`
- If the **age** of the Person instance is equal or greater than 13, but less 18, print `"You are a teenager."`
- Otherwise, print `"You are old."`

In addition, create an instance function called *yearPasses()* that increases the *age* of the person instance by *one*.

Much of the structure of the code is given to you below, but in the future, you will write this. The code that will create instances of your *Person* class is in the main function. You may not understand it all yet, but take a look just to see what's going on. Do not change any of the variable names or remove any of the code given.

Input Format

First line contains *T*, number of test cases. Each test case contains an integer *age*, representing age of the person.

Constraints
$1 \le T \le 4$
$-5 \le age \le 30$

Output Format

The code that will test your methods is already in the editor. All you have to do is edit the methods given to you in the editor so that they perform correctly as stated above. If your methods are implemented correctly, each testcase will print out either two or three lines.

If the *age* is less than zero, then your program should print out:

```
This person is not valid, setting age to 0.
You are young.
You are young.
```

If the *age* is equal or greater than 0, then your program should print out two lines. The first line that the program prints out should be the output of *amIOld()* on the current *age*. Then, three years pass via *yearPasses()* and the second line the program prints should be the output of *amIOld()* after the time has passed.

Sample Input

```
4
-1
10
16
18
```

Sample Output

```
This person is not valid, setting age to 0.
You are young.
You are young.

You are young.
You are a teenager.

You are a teenager.
You are old.

You are old.
You are old.
```

Explanation

For the first testcase, the age is less than 0 so we set the age to 0.Three years pass and the age is 3. So we print out:

```
This person is not valid, setting age to 0.
You are young.
You are young.
```

For the second testcase, the age is 10, which is considered young according to our program. Three years pass and the age is 13. 13 is considered a 'teenager' age so we print out:

```
You are young.
You are a teenager.
```

For the third testcase, the age is 16, which is the age of a teenager. Three years pass and the age is 19. 19 is considered an 'old' age according to our program so we print out:

```
You are a teenager.
You are old.
```

For the last testcase, the age is 18, which is considered an old age according to our program. Three years pass and the age is 21. 21 is still considered old so we print out:

```
You are old.
You are old.
```