**Design a test case with equivalence partitioning (EP)**

Imagine a software application that calculates the price of a cinema ticket based on the age of the customer. The app accepts number-only inputs representing the customer's age and provides the ticket price according to the following rules:

1. Children aged 0 to 12 years old receive a 50% discount.
2. Teenagers aged 13 to 17 years old receive a 25% discount.
3. Adults aged 18 to 64 years old pay the full price.
4. Senior citizens aged 65 and above receive a 30% discount.

Using the equivalence partitioning technique, create test cases for the cinema ticket pricing software based on the provided age categories. Divide the input domain (age) into partitions and select one test case from each partition to ensure adequate test coverage.

function calculate\_ticket\_price(age)

    if age >= 0 and age <= 12:

        apply 50% discount

    elif age >= 13 and age <= 17:

        apply 25% discount

    elif age >= 18 and age <= 64:

        apply 0% discount (full price)

    elif age >= 65:

        apply 30% discount

    else:

        return "Invalid age input"

    end if

    return ticket\_price

end function

## **Your task**

1. Identify the equivalence partitions for the age input domain based on the given pseudocode.
2. Select one test case from each partition.
3. Describe the input (age), expected output (discount), and reasoning for each test case.

### **Here's an example to get you started**

Test case 1: Children

* Input: Age = 5
* Expected output: 50% discount
* Reasoning: Since the age falls within the children category (0-12), a 50% discount should be applied according to the pseudocode.

Test case 2: Age = 14

Expected output: 25 % discount

Reasoning : since the age falls within the teenagers category(13-17) , a 25% discount should be applied according to the pseudocode

Test case 3: Adults

* Input: Age = 30
* Expected output: 0% discount
* Reasoning: Since the age falls within the adult category (18-64), a 0% discount should be applied according to the pseudocode.

Test case 4: old

* Input: Age = 80
* Expected output: 30% discount
* Reasoning: Since the age falls within the old category (>65), a 30% discount should be applied according to the pseudocode.

Test case 5: negative test case

Input : -1

Expected output: Invalid age input

Reasoning: since the age without categories , invalid age input should be applied according to the pseudocode