Exercise # 2

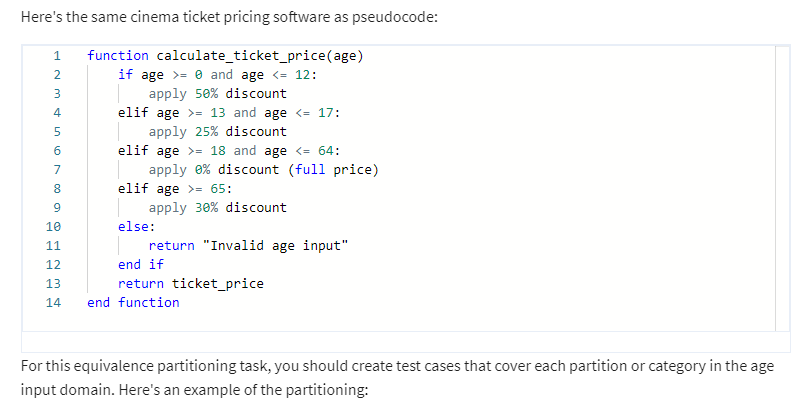
# **Test cases with equivalence partitioning**

Here's the scenario:

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.



1. Children: Age 0-12
2. Teenagers: Age 13-17
3. Adults: Age 18-64
4. Senior citizens: Age 65 and above
5. Invalid age input (negative values)

You should then select one test case from each partition to ensure adequate test coverage.

# 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.

## 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: Teenagers

* Input: Age = 15
* Expected output: 25% discount
* Reasoning: The age 15 falls within the Teenagers category (13-17), so a 25% discount should be applied.

## Test case 3: Adults

* Input: Age = 30
* Expected output: 0% discount (full price)
* Reasoning: The age 30 falls within the Adults category (18-64), so no discount is applied, and the full price is charged.

## Test case 4: Senior Citizens

* Input: Age = 70
* Expected output: 30% discount
* Reasoning: The age 70 falls within the Senior Citizens category (65 and above), so a 30% discount should be applied.

## Test case 5: Invalid Age Input

* Input: Age = -1
* Expected output: Invalid age input
* Reasoning: Negative values are not valid inputs for age. The software should return an error message indicating that the input is invalid.

In this task, you're provided with the same software specification that you used for equivalence partitioning in the previous example.

**Your task this time is to create a test case using boundary value analysis.**

# **Test Cases Using Boundary Value Analysis**

#### **Test Case 1: Age at the Boundary of the Children Category**

* **Input:** Age = 0 (Lower boundary for Children)
* **Expected Output:** 50% discount
* **Reasoning:** Age 0 is the minimum age in the Children category (0-12), so the application should apply a 50% discount.

#### **Test Case 2: Age at the Upper Boundary of the Children Category**

* **Input:** Age = 12 (Upper boundary for Children)
* **Expected Output:** 50% discount
* **Reasoning:** Age 12 is the maximum age in the Children category (0-12), so the application should apply a 50% discount.

#### **Test Case 3: Age at the Lower Boundary of the Teenagers Category**

* **Input:** Age = 13
* **Expected Output:** 25% discount
* **Reasoning:** Age 13 is the minimum age in the Teenagers category (13-17), so the application should apply a 25% discount.

#### **Test Case 4: Age at the Upper Boundary of the Teenagers Category**

* **Input:** Age = 17
* **Expected Output:** 25% discount
* **Reasoning:** Age 17 is the maximum age in the Teenagers category (13-17), so the application should apply a 25% discount.

#### **Test Case 5: Age at the Lower Boundary of the Adults Category**

* **Input:** Age = 18
* **Expected Output:** 0% discount (full price)
* **Reasoning:** Age 18 is the minimum age in the Adults category (18-64), so no discount should be applied, and the full price should be charged.

#### **Test Case 6: Age at the Upper Boundary of the Adults Category**

* **Input:** Age = 64
* **Expected Output:** 0% discount (full price)
* **Reasoning:** Age 64 is the maximum age in the Adults category (18-64), so no discount should be applied, and the full price should be charged.

#### **Test Case 7: Age at the Lower Boundary of the Senior Citizens Category**

* **Input:** Age = 65
* **Expected Output:** 30% discount
* **Reasoning:** Age 65 is the minimum age in the Senior Citizens category (65 and above), so the application should apply a 30% discount.