ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO

Sinh viên: Nguyễn Đức Duy
Mã sinh viên: 22028215
Lớp: INT3117_1

I. Mô tả bài toán

1. Bài toán

Hệ thống tính giá vé xem phim cho 1 người dựa vào **Tuổi (Age)** và **Loại ngày (DayType)**

- Nếu Age không hợp lệ (âm hoặc >100) -> báo lỗi
- Ngày chia 2 loại: ngày thường hoặc cuối tuần
- Nhóm tuổi:
  - Child: 0-12
  - Adult: 13-60
  - Senior: 60-100

Bảng giá:

- Child: Ngày thường = 50,000 VND, Cuối tuần = 60,000 VND

- Adult: Ngày thường = 100,000 VND, Cuối tuần = 120,000 VND
- Senior: Ngày thường = 70,000 VND, Cuối tuần = 80,000 VND

Đầu vào: Một số nguyên x, với x ∈ [0..100] và một string là "Weekday" hoặc "Weekend"

Đầu ra: Giá vé hoặc báo lỗi.

2. Input & kiểu dữ liệu

- Age
  - Kiểu: Integer
  - Khoảng hợp lệ: 0 .. 100
  - Không hợp lệ: null, string non-numeric, float.
- DayType
  - Kiểu: string
  - Giá trị hợp lệ: "Weekday" hoặc "Weekend" .
  - Không chấp nhận: null, empty string, hay ngày rơi ngoài hai loại trên.

3. Rules.

- Nếu Age ngoài 0..100 -> trả lỗi InvalidAge
- Phân nhóm tuổi
  - Child: 0..12
  - Adult: 13..60
  - Senior: 60..100
- Bảng giá
  - Child: Ngày thường = 50,000 VND, Cuối tuần = 60,000 VND
  - Adult: Ngày thường = 100,000 VND, Cuối tuần = 120,000 VND
  - Senior: Ngày thường = 70,000 VND, Cuối tuần = 80,000 VND
- Xử lý ngoại lệ
  - Nếu DayType không hợp lệ -> trả lỗi InvalidDayType
  - Nếu Age là non-integer (ví dụ 12.5) -> trả lỗi InvalidAge
- Output mặc định: trả về giá tiền

4. Output

1. Giá vé
2. Báo lỗi

5. Ví dụ input -> output minh họa

- Input: `Age=8`, `DayType="Weekday"` → Output: `50000`
- Input: `Age=12`, `DayType="Weekend"` → Output: `60000`
- Input: `Age=13`, `DayType="Weekday"` → Output: `100000`

II. Phương pháp bảng quyết định

1. Các điều kiện và hành động.
   - Điều kiện 1: Age
     - C1.1: Age ∈ [0..12] -> Child
     - C1.2: Age ∈ [13..60] -> Adult
     - C1.3: Age ∈ [60..100] -> Senior
     - C1.4: Age invalid -> lỗi
   - Điều kiện 2: DayType
     - C2.1: Weekday
     - C2.2: Weekend
     - C2.3: Invalid -> lỗi
   - Số tổ hợp = 4 x 3 = 12
2. Bảng quyết định

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Điều kiện	Age ∈ [0..12]	T	T	T									
	Age ∈ [13..60]	-			T	T	T						
	Age ∈ [60..100]	-						T	T	T			
	Age invalid	-									T	T	T
	Weekday	T			T			T			T		
	Weekend	-	T			T			T			T	
	Invalid daytype	-		T			T			T			T
Hành động	invalid age										X	X	X
	invalid daytype			X			X			X			
	Giá vé = 50000	X											

	Giá vé = 60000		X										
	Giá vé = 70000							X					
	Giá vé = 80000								X				
	Giá vé = 100000				X								
	Giá vé = 120000					X							

3. Các test case.

TC ID	Age	DayType	Expected Output
1	8	Weekday	Vé = 50.000
2	12	Weekend	Vé = 60.000
3	10	“abc”	Lỗi DayType
4	20	Weekday	Vé = 100.000
5	30	Weekend	Vé = 120.000
6	40	“summer”	Lỗi DayType
7	70	Weekday	Vé = 70.000
8	80	Weekend	Vé = 80.000
9	90	“time”	Lỗi DayType
10	-1	Weekday	Lỗi Age
11	150	Weekend	Lỗi Age
12	-5	“age”	Lỗi Age

III. Phương pháp kiểm thử giá trị biên

1. Xác định miền và các giá trị của biến.

   Ta thấy:

- Age: là số nguyên có thứ tự liên tục -> phù hợp với BVA.

- DayType là dạng định tính, không có thứ tự -> không phù hợp với BVA.

  => Ta chỉ áp dụng BVA với Age

- Với mỗi miền giá trị của Age, ta chọn 5 giá trị: min, min+, nom, max-, max.
- Các giá trị biên cho từng miền của Age:
  - Child [0 - 12]: 0, 1, 6, 11, 12.
  - Adult [12 - 60]: 13, 14, 24, 59, 60.
  - Senior [61 - 100]: 61, 62, 81, 99, 100.
- Các giá trị của DayType: “WeekDay” và “Weekend”

  => Số test case = 5 x 3 x 2 = 30

2. Các test case

TC ID	Age	DayType	Expected Output
1	0	WeekDay	Vé = 50.000
2	1	WeekDay	Vé = 50.000
3	6	WeekDay	Vé = 50.000
4	11	WeekDay	Vé = 50.000
5	12	WeekDay	Vé = 50.000
6	13	WeekDay	Vé = 100.000
7	14	WeekDay	Vé = 100.000
8	24	WeekDay	Vé = 100.000
9	59	WeekDay	Vé = 100.000
10	60	WeekDay	Vé = 100.000
11	61	WeekDay	Vé = 70.000
12	62	WeekDay	Vé = 70.000
13	81	WeekDay	Vé = 70.000
14	99	WeekDay	Vé = 70.000
15	100	WeekDay	Vé = 70.000

16	0	Weekend	Vé = 60.000
17	1	Weekend	Vé = 60.000
18	6	Weekend	Vé = 60.000
19	11	Weekend	Vé = 60.000
20	12	Weekend	Vé = 60.000
21	13	Weekend	Vé = 80.000
22	14	Weekend	Vé = 80.000
23	24	Weekend	Vé = 80.000
24	59	Weekend	Vé = 80.000
25	60	Weekend	Vé = 80.000
26	61	Weekend	Vé = 120.000
27	62	Weekend	Vé = 120.000
28	81	Weekend	Vé = 120.000
29	99	Weekend	Vé = 120.000
30	100	Weekend	Vé = 120.000

IV. Code

Link mã nguồn: Testing

1. Cài đặt chương trình

```python
def ticket_cost(age: int, daytype: str):
    if age < 0 or age > 100:
        return "Lỗi Age"
    if daytype not in ["Weekday", "Weekend"]:
        return "Lỗi DayType"

    if age <= 12: #Child
        return 50000 if daytype == "Weekday" else 60000
    elif age <= 60: #Adult
        return 100000 if daytype == "Weekday" else 120000
    else: #Senior
        return 70000 if daytype == "Weekday" else 80000

```

2. Phương pháp bảng quyết định

```python
import pytest
from ticket import ticket_cost


test_cases = [
    (8, "Weekday", 50000),   # Rule 1
    (12, "Weekend", 60000),  # Rule 2
    (10, "abc", "Lỗi DayType"),  # Rule 3
    (20, "Weekday", 100000),  # Rule 4
    (30, "Weekend", 120000),  # Rule 5
    (40, "summer", "Lỗi DayType"),  # Rule 6
    (70, "Weekday", 70000), #Rule 7
    (80, "Weekend", 80000), #Rule 8
    (90, "time", "Lỗi DayType"), #Rule 9
    (-1, "Weekday", "Lỗi Age"), #Rule 10
    (150, "Weekend", "Lỗi Age"), #Rule 11
    (-5, "age", "Lỗi Age"), #Rule 12
]

@pytest.mark.parametrize("age,daytype,expected", test_cases)
def test_decision_table(age, daytype, expected):
    assert ticket_cost(age, daytype) == expected
```

TC ID	Age	DayType	Expected Output	Result
1	8	Weekday	Vé = 50.000	Pass
2	12	Weekend	Vé = 60.000	Pass
3	10	“abc”	Lỗi DayType	Pass
4	20	Weekday	Vé = 100.000	Pass
5	30	Weekend	Vé = 120.000	Pass

6	40	“summer”	Lỗi DayType	Pass
7	70	Weekday	Vé = 70.000	Pass
8	80	Weekend	Vé = 80.000	Pass
9	90	“time”	Lỗi DayType	Pass
10	-1	Weekday	Lỗi Age	Pass
11	150	Weekend	Lỗi Age	Pass
12	-5	“age”	Lỗi Age	Pass

3. Phương pháp kiểm thử giá trị biên

```python
import pytest
from ticket import ticket_cost

#Test Child
@pytest.mark.parametrize("age", [0, 1, 6, 11, 12])
@pytest.mark.parametrize("daytype,expected", [
    ("Weekday", 50000),
    ("Weekend", 60000),
])
def test_child_bva(age, daytype, expected):
    assert ticket_cost(age, daytype) == expected

#Test Adult
@pytest.mark.parametrize("age", [13, 14, 24, 59, 60])
@pytest.mark.parametrize("daytype,expected", [
    ("Weekday", 100000),
    ("Weekend", 120000),
])
def test_adult_bva(age, daytype, expected):
    assert ticket_cost(age, daytype) == expected

#Test Senior
@pytest.mark.parametrize("age", [61, 62, 81, 99, 100])
@pytest.mark.parametrize("daytype,expected", [
    ("Weekday", 70000),
    ("Weekend", 80000),
])
def test_senior_bva(age, daytype, expected):
    assert ticket_cost(age, daytype) == expected
```

TC ID	Age	DayType	Expected Output	Result
1	0	WeekDay	Vé = 50.000	Pass
2	1	WeekDay	Vé = 50.000	Pass
3	6	WeekDay	Vé = 50.000	Pass
4	11	WeekDay	Vé = 50.000	Pass
5	12	WeekDay	Vé = 50.000	Pass
6	13	WeekDay	Vé = 100.000	Pass
7	14	WeekDay	Vé = 100.000	Pass
8	24	WeekDay	Vé = 100.000	Pass
9	59	WeekDay	Vé = 100.000	Pass
10	60	WeekDay	Vé = 100.000	Pass
11	61	WeekDay	Vé = 70.000	Pass
12	62	WeekDay	Vé = 70.000	Pass
13	81	WeekDay	Vé = 70.000	Pass
14	99	WeekDay	Vé = 70.000	Pass
15	100	WeekDay	Vé = 70.000	Pass
16	0	Weekend	Vé = 60.000	Pass
17	1	Weekend	Vé = 60.000	Pass
18	6	Weekend	Vé = 60.000	Pass
19	11	Weekend	Vé = 60.000	Pass

20	12	Weekend	Vé = 60.000	Pass
21	13	Weekend	Vé = 80.000	Pass
22	14	Weekend	Vé = 80.000	Pass
23	24	Weekend	Vé = 80.000	Pass
24	59	Weekend	Vé = 80.000	Pass
25	60	Weekend	Vé = 80.000	Pass
26	61	Weekend	Vé = 120.000	Pass
27	62	Weekend	Vé = 120.000	Pass
28	81	Weekend	Vé = 120.000	Pass
29	99	Weekend	Vé = 120.000	Pass
30	100	Weekend	Vé = 120.000	Pass