Assignment 01: การเรียนรู้ Conditional Logic (if-else,switch-case) สำหรับ Game Development

📋 ภาพรวมของ Assignment

เรียนรู้พื้นฐานของ conditional statements และ decision-making logic โดยการ implement 17 methods ที่ใช้งานจริง Assignment นี้เน้นที่ if-else statements, switch-case structures และการรวม conditions หลายตัวเพื่อสร้าง scenarios แบบเกม แต่ละ method จะแสดงผลลัพธ์โดยใช้ `Debug.Log()` (alias ของ`AssignmentDebugConsole.Log()`) และต้องตรงกับผลลัพธ์ที่คาดหวังจาก test cases อย่างแม่นยำ

🎯 จุดประสงค์การเรียนรู้

- เรียนรู้โครงสร้าง if-else และ switch-case conditional
- Implement boolean logic ที่ซับซ้อนด้วย AND/OR operators
- จัดการ edge cases และตรวจสอบ user inputs อย่างมีประสิทธิภาพ
- นำ conditional logic มาใช้ในสถานการณ์ game development จริง
- เขียน code ที่สะอาด อ่านง่าย และปฏิบัติตาม best practices

📚 โครงสร้างของ Assignment

- **Example Methods (7 methods)** - การ implement ฝึกหัดด้วย conditional logic พื้นฐาน พร้อมกันในห้องเรียน
- **Level 1: Simple (7 methods)** - การดำเนินการ if-else และ switch-case ขั้นพื้นฐาน
- **EX Level 2: Moderate (4 methods)** - Conditional logic ผสมผสานกับ game mechanics

---

Example Methods

Methods เหล่านี้แสดงแนวคิด conditional พื้นฐาน Implement เพื่อฝึกหัดแต่จะไม่มีการให้คะแนน

1. SyntaxIf (2 test cases)

วัตถุประสงค์: แสดงความเข้าใจโครงสร้างของ If โดยเข้าใจสโคปของ If ในการใช้งาน

Method Signature:

```
void SyntaxIf(bool isSixoClock)
```

Logic ที่ต้อง implement:

- ตรวจสอบว่า isSixoClock เป็นจริงหรือไม่
- ถ้า isSixoClock เป็นจริง ให้แสดงข้อความ "You can get in"
- แสดงข้อความ "Crack Crack!!!!" เสมอ

Test Cases:

```
AS01.E1: Input: true → Output:
You can get in
Crack Crack!!!!
AS01.E1b: Input: false → Output:
Crack Crack!!!!
```

Game Context: ระบบตรวจสอบ password, การ authentication ผู้ใช้

Implementation Hint:

```
// if (password != "Moon")
//   Debug.Log("wrong password");
// if (password == "Moon")
//   Debug.Log("password is correct");
```

2. StringComparisonExample (2 test cases)

วัตถุประสงค์: แสดงการเปรียบเทียบ string โดยใช้ if statements (==, !=)

Method Signature:

```
void StringComparisonExample(string password)
```

Logic ที่ต้อง implement:

- ตรวจสอบว่า password เท่ากับ "Moon" หรือไม่
- แสดงข้อความที่เหมาะสมสำหรับ password ที่ถูกต้อง/ผิด
- แสดงผลลัพธ์การเปรียบเทียบ boolean

Test Cases:

```
AS01.E1: Input: "Moon" → Output:
password is correct
AS01.E1b: Input: "Sun" → Output:
wrong password
```

Game Context: ระบบตรวจสอบ password, การ authentication ผู้ใช้

Implementation Hint:

```
// if (password != "Moon")
//   Debug.Log("wrong password");
// if (password == "Moon")
//   Debug.Log("password is correct");
```

3. NumberComparisonExample (3 test cases)

วัตถุประสงค์: แสดงการเปรียบเทียบตัวเลขโดยใช้ if statements

Method Signature:

```
void NumberComparisonExample(int number)
```

Logic ที่ต้อง implement:

- เปรียบเทียบ number กับ 10 โดยใช้ comparison operators ทั้งหมด (>, >=, <, <=, !=, ==)
- แสดงผลลัพธ์สำหรับการเปรียบเทียบแต่ละตัวที่เป็น true

Test Cases:

```
AS01.E2: Input: 11 → Output:
My Number > 10
My Number >= 10
My Number != 10
AS01.E2b: Input: 10 → Output:
My Number == 10
My Number >= 10
My Number <= 10
AS01.E2c: Input: 9 → Output:
My Number < 10
My Number <= 10
My Number != 10
```

Game Context: การเปรียบเทียบคะแนน, ข้อกำหนดระดับ, การตรวจสอบ stats

Implementation Hint:

```
// if (number > 10) Debug.Log("My Number > 10");
// if (number < 10) Debug.Log("My Number < 10");
// ...
```

4. AndOrOperatorExample (3 test cases)

วัตถุประสงค์: แสดง AND และ OR operators ใน if statements (&&, ||)

Method Signature:

```
void AndOrOperatorExample(int number)
```

Logic ที่ต้อง implement:

- ตรวจสอบว่า number อยู่ระหว่าง 8 และ 12 (exclusive) โดยใช้ AND
- ตรวจสอบว่า number ตรงกับเงื่อนไข OR (> 8 OR < 12)
- แสดงข้อความที่เหมาะสม

Test Cases:

```
AS01.E3: Input: 10 → Output:
My Number  8 > < 12
My Number  or 8 || 12
AS01.E3b: Input: 7 → Output:
My Number  or 8 || 12
AS01.E3c: Input: 13 → Output:
My Number  or 8 || 12
```

Game Context: การตรวจสอบช่วง, conditional game mechanics

Implementation Hint:

```
// if (number > 8 && number < 12)
// if (number > 8 || number < 12)
```

5. GuessingNumberExample (2 test cases)

วัตถุประสงค์: เกมทายตัวเลขง่ายๆ โดยใช้ if-else statements โดย randomNumber จะถูกส่งมาจาก Test case

Method Signature:

```
void GuessingNumberExample(int guessingNumber, int randomNumber)
```

Logic ที่ต้อง implement:

- แสดงตัวเลขเป้าหมาย โดย randomNumber จะถูกส่งมาจาก Test case
- เปรียบเทียบการทายกับเป้าหมาย
- แสดงข้อความชนะหรือแพ้
- ชนะ : Guessing number {guessingNumber} Congratulations! You guessed the correct number.
- แพ้ : I guess we can just agree to disagree. **Test Cases:**

```
AS01.E4: Input: 5, 5 → Output:
Guessing number 5 Congratulations! You guessed the correct number.
AS01.E4b: Input: 3, 5 → Output:
Guessing number 5 I guess we can just agree to disagree.
```

Game Context: Mini-games, random events, mechanics ที่ขึ้นอยู่กับโชค

6. GuessingNumberMoreOrLessExample (3 test cases)

วัตถุประสงค์: เกมทายตัวเลขขั้นสูงพร้อม hints ทิศทางโดยใช้ if-else-if โดย randomNumber จะถูกส่งมาจาก Test case

Method Signature:

```
void GuessingNumberMoreOrLessExample(int guessingNumber, int randomNumber)
```

Logic ที่ต้อง implement:

- แสดงตัวเลขเป้าหมาย โดย randomNumber จะถูกส่งมาจาก Test case
- ให้ feedback: ต่ำเกินไป, สูงเกินไป หรือถูกต้อง
- ต่ำเกินไป : Guessing number {guessingNumber} Too low! Try again.
- สูงเกินไป : Guessing number {guessingNumber} Too high! Try again.
- ถูกต้อง :Guessing number {guessingNumber} Congratulations! We are same mind. **Test Cases:**

```
AS01.E5: Input: 3, 5 → Output:
Guessing number 5 Too low! Try again.
AS01.E5b: Input: 7, 5 → Output:
Guessing number 5 Too high! Try again.
AS01.E5c: Input: 5, 5 → Output:
Guessing number 5 Congratulations! We are same mind.
```

Game Context: Puzzle games, ระบบปรับความยาก

7. VerifyIdentityExample (4 test cases)

วัตถุประสงค์: การตรวจสอบตัวตนหลายระดับโดยใช้ nested if statements

Method Signature:

```
void VerifyIdentityExample(string username, string password, int age, bool isPaid)
```

Logic ที่ต้อง implement:

- ตรวจสอบ username และ password โดย Testcase จะกำหนดให้ username == "user", password == "user123"
  - ถ้า username และ password ถูกต้อง จะแสดงข้อความ "You have user access."
  - ถ้า username และ password ไม่ถูกต้อง จะแสดงข้อความ "You have guest access."
- จากนั้นตรวจสอบสถานะ VIP
  - ถ้า isPaid เป็นจริง จะแสดงข้อความ "welcome vip member"
  - ถ้า isPaid เป็นเท็จ จะแสดงข้อความ "welcome free member"
- จากนั้นอายุสำหรับการเข้าถึงเนื้อหาเพิ่มเติม
  - ถ้า age มีค่ามากกว่า 18 จะแสดงข้อความ "You have access to exclusive content"

Test Cases:

```
AS01.E7: Input: "user", "user123", 20, true → Output:
You have user access.
welcome vip member
You have access to exclusive content

AS01.E7b: Input: "user", "user123", 15, true → Output:
You have user access.
welcome vip member

AS01.E7c: Input: "user", "user123", 20, false → Output:
You have user access.
welcome free member

AS01.E7d: Input: "guest", "pass", 20, false → Output:
You have guest access.
```

Game Context: การ authentication ผู้ใช้, การเข้าถึงเนื้อหา premium, การตรวจสอบอายุ

Game Context: Mini-games, random events, mechanics ผู้เล่นต่อสู้ AI

Level 1: Simple (7 methods)

1. CheckNumberSign (5 test cases)

วัตถุประสงค์: กำหนดว่าตัวเลขเป็นบวก ลบ หรือศูนย์

Method Signature:

```
void CheckNumberSign(int number)
```

Logic ที่ต้อง implement:

- ใช้ if-else-if chain เพื่อตรวจสอบเครื่องหมายของตัวเลข
- แสดง "Positive", "Negative" หรือ "Zero"

Test Cases:

```
AS01.01: Input: 5 → Output: Positive
AS01.01b: Input: -3 → Output: Negative
AS01.01c: Input: 0 → Output: Zero
AS01.01d: Input: 2147483647 → Output: Positive
AS01.01e: Input: -2147483648 → Output: Negative
```

Game Context: การคำนวณพลังชีวิต/ความเสียหาย, การตรวจสอบคะแนน, การตรวจจับทิศทาง

Implementation Hint:

```
// if (number > 0)
//   Debug.Log("Positive");
// else if (number < 0)
//   Debug.Log("Negative");
// else
//   Debug.Log("Zero");
```

2. GetDayName (10 test cases)

วัตถุประสงค์: คืนค่าชื่อวันสำหรับจำนวนเต็มที่กำหนด

Method Signature:

```
void GetDayName(int day)
```

Logic ที่ต้อง implement:

- ใช้ if-else-if หรือ switch-case เพื่อแมปตัวเลขกับชื่อวัน
- จัดการ input ที่ไม่ถูกต้องด้วย "Invalid day"
- โดยกำหนดให้ 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday, 7=Sunday, other=Invalid day

Test Cases:

```
AS01.02: Input: 1 → Output: Monday
AS01.02b: Input: 2 → Output: Tuesday
AS01.02c: Input: 3 → Output: Wednesday
AS01.02d: Input: 4 → Output: Thursday
AS01.02e: Input: 5 → Output: Friday
AS01.02f: Input: 6 → Output: Saturday
AS01.02g: Input: 7 → Output: Sunday
AS01.02h: Input: 0 → Output: Invalid day
AS01.02i: Input: 8 → Output: Invalid day
AS01.02j: Input: -5 → Output: Invalid day
```

Game Context: ระบบ quest รายวัน, calendar events, mechanics ที่ขึ้นอยู่กับเวลา

Implementation Hint:

```
// switch (day)
// {
//   case 1: Debug.Log("Monday"); break;
//   case 2: Debug.Log("Tuesday"); break;
//   ...
```

```
//   default: Debug.Log("Invalid day"); break;
// }
```

3. ValidatePassword (7 test cases)

วัตถุประสงค์: ตรวจสอบ password input ด้วยการจับคู่ string

Method Signature:

```
void ValidatePassword(string inputPassword, string correctPassword)
```

Logic ที่ต้อง implement:

- เปรียบเทียบ strings แบบแม่นยำ (case-sensitive)
- แสดง "True" หรือ "False"
- โดย testcase กำหนดให้ correctPassword มีค่า "secret123"

Test Cases:

```
AS01.03: Input: "secret123", "secret123" → Output: True
AS01.03b: Input: "wrongpass", "secret123" → Output: False
AS01.03c: Input: "", "" → Output: True
AS01.03d: Input: "secret123", "Secret123" → Output: False
AS01.03e: Input: " secret123 ", "secret123" → Output: False
AS01.03f: Input: "secret123", "" → Output: False
AS01.03g: Input: "", "secret123" → Output: False
```

Game Context: ระบบ login, พื้นที่ปลอดภัย, การตรวจสอบ cheat code

Implementation Hint:

```
// if (inputPassword == correctPassword)
//   Debug.Log("True");
// else
//   Debug.Log("False");
```

4. GetGrade (14 test cases)

วัตถุประสงค์: คืนค่าเกรดตัวอักษรสำหรับคะแนนที่กำหนด

Method Signature:

```
void GetGrade(int score)
```

Logic ที่ต้อง implement:

- ใช้ if-else chain กับ score thresholds
- A:80, B:70, C:60, D:50, F: อื่นๆ

Test Cases:

```
AS01.04: Input: 95 → Output: A
AS01.04b: Input: 85 → Output: A
AS01.04c: Input: 80 → Output: A
AS01.04d: Input: 75 → Output: B
AS01.04e: Input: 70 → Output: B
AS01.04f: Input: 65 → Output: C
AS01.04g: Input: 60 → Output: C
AS01.04h: Input: 55 → Output: D
AS01.04i: Input: 50 → Output: D
AS01.04j: Input: 49 → Output: F
AS01.04k: Input: 0 → Output: F
AS01.04l: Input: 100 → Output: A
AS01.04m: Input: -1 → Output: F
AS01.04n: Input: 101 → Output: A
```

Game Context: ระบบจัดอันดับผู้เล่น, ระดับ achievement, การประเมินผลงาน

Implementation Hint:

```
// if (score >= 80)
//   Debug.Log("A");
// else if (score >= 70)
//   Debug.Log("B");
// ...
```

5. IsLeapYear (9 test cases)

วัตถุประสงค์: ตรวจสอบว่าปีเป็นปีอธิกสุรทินหรือไม่โดยใช้กฎที่ซับซ้อน

Method Signature:

```
void IsLeapYear(int year)
```

Logic ที่ต้อง implement:

- หารด้วย 400 ลงตัว: ปีอธิกสุรทิน
- หารด้วย 100 ลงตัว (แต่ไม่หาร 400 ลงตัว): ไม่ใช่ปีอธิกสุรทิน
- หารด้วย 4 ลงตัว (แต่ไม่หาร 100 ลงตัว): ปีอธิกสุรทิน
- อื่นๆ: ไม่ใช่ปีอธิกสุรทิน

Test Cases:

```
AS01.05: Input: 2000 → Output: True
AS01.05b: Input: 2020 → Output: True
AS01.05c: Input: 1900 → Output: False
AS01.05d: Input: 2004 → Output: True
AS01.05e: Input: 2100 → Output: False
AS01.05f: Input: 2400 → Output: True
AS01.05g: Input: 1999 → Output: False
AS01.05h: Input: -400 → Output: True
AS01.05i: Input: 0 → Output: True
```

Game Context: ระบบปฏิทิน, seasonal events, mechanics ที่ขึ้นอยู่กับเวลา

Implementation Hint:

```
// if (year % 400 == 0)
//   Debug.Log("True");
// else if (year % 100 == 0)
//   Debug.Log("False");
// else if (year % 4 == 0)
//   Debug.Log("True");
// else
//   Debug.Log("False");
```

6. Calculate (12 test cases)

วัตถุประสงค์: สร้างโปรแกรมเครื่องคิดเลขอย่างง่ายโดยตรวจสอบเครื่องหมายก่อนตัดสินใจดำเนินการพร้อมวิธีป้องกันการคำนวณผิดพลาด

Method Signature:

```
void Calculate(double num1, char op, double num2)
```

Logic ที่ต้อง implement:

- รองรับ operators +, -, *, /
- จัดการการหารด้วยศูนย์ จะต้องแสดงข้อความ `Error: Cannot divide by zero.`
- จัดรูปแบบตัวเลขโดยไม่มีทศนิยมที่ไม่จำเป็น
- ตรวจสอบ operator input

Test Cases:

```
AS01.06: Input: 5.0, '+', 3.0 → Output: Result: 8
AS01.06b: Input: 5.0, '-', 3.0 → Output: Result: 2
AS01.06c: Input: 5.0, '*', 3.0 → Output: Result: 15
```

```
AS01.06d: Input: 6.0, '/', 2.0 → Output: Result: 3
AS01.06e: Input: 6.0, '/', 0.0 → Output: Error: Cannot divide by zero.
AS01.06f: Input: 6.0, 'x', 2.0 → Output: Invalid operator. Please use +, -, *, or
/.
AS01.06g: Input: -5.0, '+', -3.0 → Output: Result: -8
AS01.06h: Input: 0.0, '+', 0.0 → Output: Result: 0
AS01.06i: Input: 1e10, '+', 1e10 → Output: Result: 20000000000
AS01.06j: Input: 0.1, '+', 0.2 → Output: Result: 0.3
AS01.06k: Input: 5.0, ' ', 3.0 → Output: Invalid operator. Please use +, -, *, or
/.
AS01.06l: Input: 5.0, 'X', 3.0 → Output: Invalid operator. Please use +, -, *, or
/.
```

Game Context: การคำนวณความเสียหาย, การจัดการ resources, การคำนวณ stats

Implementation Tips:

- ใช้ switch-case สำหรับ operators
- จัดรูปแบบผลลัพธ์โดยใช้ string formatting ที่เหมาะสม

Implementation Hint:

```
// switch (op)
// {
//   case '+': result = num1 + num2; break;
//   case '/':
//     if (num2 == 0)
//       Debug.Log("Error: Cannot divide by zero.");
//     else
//       result = num1 / num2;
//     break;
//   default:
//     Debug.Log("Invalid operator. Please use +, -, *, or /.");
//     return;
// }
```

7. GetSeason (16 test cases)

วัตถุประสงค์: คืนค่าฤดูกาลสำหรับเดือนที่กำหนดพร้อมการตรวจสอบที่ครอบคลุม

Method Signature:

```
void GetSeason(int month)
```

Logic ที่ต้อง implement:

- Winter: December (12), January (1), February (2)
- Spring: March (3) ถึง May (5)

- Summer: June (6) ถึง August (8)
- Fall: September (9) ถึง November (11)
- Invalid: เดือนนอกช่วง 1-12

Test Cases:

```
AS01.07: Input: 1 → Output: It's Winter.
AS01.07b: Input: 2 → Output: It's Winter.
AS01.07c: Input: 12 → Output: It's Winter.
AS01.07d: Input: 3 → Output: It's Spring.
AS01.07e: Input: 4 → Output: It's Spring.
AS01.07f: Input: 5 → Output: It's Spring.
AS01.07g: Input: 6 → Output: It's Summer.
AS01.07h: Input: 7 → Output: It's Summer.
AS01.07i: Input: 8 → Output: It's Summer.
AS01.07j: Input: 9 → Output: It's Fall.
AS01.07k: Input: 10 → Output: It's Fall.
AS01.07l: Input: 11 → Output: It's Fall.
AS01.07m: Input: 0 → Output: Invalid month number. Please enter a number between 1
and 12.
AS01.07n: Input: 13 → Output: Invalid month number. Please enter a number between
1 and 12.
AS01.07o: Input: -1 → Output: Invalid month number. Please enter a number between
1 and 12.
AS01.07p: Input: 100 → Output: Invalid month number. Please enter a number between
1 and 12.
```

Game Context: Seasonal events, ระบบสภาพอากาศ, การเปลี่ยนแปลงสิ่งแวดล้อม

Implementation Hint:

```
// if (month >= 1 && month <= 12)
// {
//   if (month == 12 || month == 1 || month == 2)
//     Debug.Log("It's Winter.");
//   else if (month >= 3 && month <= 5)
//     Debug.Log("It's Spring.");
//   ...
// }
// else
//   Debug.Log("Invalid month number. Please enter a number between 1 and 12.");
```

EX Level 2: Moderate (2 methods)

1. PurchasingSystemExample (4 test cases)

วัตถุประสงค์: ระบบซื้อขายโดยใช้ nested if statements

Method Signature:

```
void PurchasingSystemExample(int quantity, int price, int payment)
```

Logic ที่ต้อง implement:

- ตรวจสอบว่าสินค้ามีใน stock หรือไม่
  - ถ้าไม่เพียงพอให้แสดงข้อความ "สินค้าหมด"
- จากนั้นตรวจสอบการชำระเงินเพียงพอ
  - ถ้าเพียงพอให้แสดงข้อความ "คุณได้รับสินค้าแล้ว"
  - ถ้าไม่พอให้แสดงข้อความ "คุณมีเงินไม่พอ"
- จากนั้นคำนวณและแสดงเงินทอนถ้าเหมาะสม
  - แสดงข้อความ "คุณได้รับเงินทอน {change} บาท"

Test Cases:

```
AS01.E6: Input: 1, 10, 20 → Output:
คุณได้รับสินค้าแล้ว
คุณได้รับเงินทอน 10 บาท
AS01.E6b: Input: 1, 10, 10 → Output:
คุณได้รับสินค้าแล้ว
AS01.E6c: Input: 1, 10, 5 → Output:
คุณมีเงินไม่พอ
AS01.E6d: Input: 0, 10, 20 → Output:
สินค้าหมด
```

Game Context: ร้านค้าในเกม, การจัดการ inventory, ระบบเศรษฐกิจ

2. RockPaperScissorsExample (6 test cases)

วัตถุประสงค์: Logic เกมคลาสสิกโดยใช้ if-else statements

Method Signature:

```
void RockPaperScissorsExample(int userChoice, int computerChoice)
```

Logic ที่ต้อง implement:

- ตรวจสอบ user choice (0=Rock, 1=Paper, 2=Scissors)
- กำหนดผู้ชนะโดยใช้กฎของเกม
- แสดงผลลัพธ์เป็นภาษาไทย
- computerChoice จะถูกส่งมาจาก Test Case

Test Cases:

```
AS01.E8: Input: 0, 0 → Output: เสมอ
AS01.E8b: Input: 0, 2 → Output: คุณชนะ!
AS01.E8c: Input: 1, 0 → Output: คุณชนะ!
AS01.E8d: Input: 2, 1 → Output: คุณชนะ!
AS01.E8e: Input: 2, 0 → Output: คุณแพ้!
AS01.E8f: Input: 3, 0 → Output: กรุณาเลือกเป็นตัวเลขที่ถูกต้อง
```

3. CalculateWeaponDamage (6 test cases)

วัตถุประสงค์: คำนวณความเสียหายของอาวุธโดยใช้ multipliers ตามประเภท

Method Signature:

```
void CalculateWeaponDamage(string weaponType, int baseDamage)
```

Logic ที่ต้อง implement:

- ใช้ damage multipliers ตามประเภทอาวุธ
- จัดการ weapon type input แบบไม่คำนึงถึงตัวใหญ่เล็ก
- แสดงความเสียหายสุดท้ายเป็น integer

Weapon Type Multipliers:

- sword: 1.3 (โบนัส 30%)
- axe: 1.4 (โบนัส 40%)
- bow: 1.2 (โบนัส 20%)
- staff: 1.5 (โบนัส 50%)
- dagger: 1.1 (โบนัส 10%)
- unknown/other: 1.0 (ไม่มีโบนัส)

Test Cases:

```
AS01.08: Input: "sword", 50 → Output: 65
AS01.08b: Input: "axe", 50 → Output: 70
AS01.08c: Input: "bow", 50 → Output: 60
AS01.08d: Input: "staff", 50 → Output: 75
AS01.08e: Input: "dagger", 50 → Output: 55
AS01.08f: Input: "unknown", 50 → Output: 50
```

Game Context: ระบบการต่อสู้, การปรับสมดุลอาวุธ, การคำนวณความเสียหาย

Implementation Tips:

- ใช้ switch-case กับ toLowerCase() สำหรับการจัดการ case
- แปลงผลลัพธ์สุดท้ายเป็น integer

Implementation Hint:

```
// double multiplier = 1.0;
// switch (weaponType?.ToLower())
// {
//   case "sword": multiplier = 1.3; break;
//   case "axe": multiplier = 1.4; break;
//   ...
//   default: multiplier = 1.0; break;
// }
// int totalDamage = (int)(baseDamage * multiplier);
// Debug.Log(totalDamage.ToString());
```

4. DeterminePlayerRank (33 test cases)

วัตถุประสงค์: กำหนดอันดับผู้เล่นและคำนวณรางวัลตามคะแนนและเวลาที่ใช้ในการเล่น

Method Signature:

```
void DeterminePlayerRank(int score, int completionTime)
```

Logic ที่ต้อง implement:

- ตรวจสอบ inputs (ไม่มีค่าลบ)
- กำหนดอันดับตาม score thresholds
- คำนวณ base coins + time bonus
- แสดงข้อความอันดับและรางวัลที่จัดรูปแบบ

Rank Thresholds:

- Gold: 8000+ คะแนน (100 base coins)
- Silver: 6000-7999 คะแนน (75 base coins)
- Bronze: 4000-5999 คะแนน (50 base coins)
- Participation: 0-3999 คะแนน (25 base coins)

Time Bonus:

- 0-30 นาที: +25 coins
- 31-60 นาที: +10 coins
- 61+ นาที: +0 coins

Test Cases:

```
AS01.09: Input: -1, 30 → Output: Invalid score or time
AS01.09b: Input: 5000, -5 → Output: Invalid score or time
AS01.09c: Input: 0, 0 → Output: Participation Rank - 50 coins earned!
AS01.09d: Input: 2000, 25 → Output: Participation Rank - 50 coins earned!
AS01.09e: Input: 2000, 35 → Output: Participation Rank - 35 coins earned!
AS01.09f: Input: 2000, 70 → Output: Participation Rank - 25 coins earned!
```

```
AS01.09g: Input: 4000, 30 → Output: Bronze Rank - 75 coins earned!
AS01.09h: Input: 4500, 45 → Output: Bronze Rank - 60 coins earned!
AS01.09i: Input: 6000, 30 → Output: Silver Rank - 100 coins earned!
AS01.09j: Input: 6500, 45 → Output: Silver Rank - 85 coins earned!
AS01.09k: Input: 8000, 30 → Output: Gold Rank - 125 coins earned!
AS01.09l: Input: 8500, 45 → Output: Gold Rank - 110 coins earned!
[Test cases เพิ่มเติมครอบคลุม boundary conditions...]
```

Game Context: ความก้าวหน้าของผู้เล่น, ระบบ leaderboard, รางวัล achievement

Implementation Tips:

- ตรวจสอบ inputs ก่อน
- ใช้ if-else chain สำหรับการกำหนดอันดับ
- ใช้ nested if-else สำหรับการคำนวณ time bonus

Implementation Hint:

```
// if (score < 0 || completionTime < 0)
// {
//   Debug.Log("Invalid score or time");
//   return;
// }
//
// string rank; int baseCoins;
// if (score >= 8000) { rank = "Gold"; baseCoins = 100; }
// else if (score >= 6000) { rank = "Silver"; baseCoins = 75; }
// ...
//
// int timeBonus = 0;
// if (completionTime <= 30) timeBonus = 25;
// else if (completionTime <= 60) timeBonus = 10;
//
// int totalCoins = baseCoins + timeBonus;
// Debug.Log($"{rank} Rank - {totalCoins} coins earned!");
```

⚠ แนวทางการ Implementation ที่สำคัญ

ข้อกำหนดคุณภาพ Code:

1. **Error Handling**: ตรวจสอบ inputs และจัดการ edge cases เสมอ
2. **Performance**: เลือกโครงสร้าง conditional ที่เหมาะสมเพื่อประสิทธิภาพ
3. **Code Structure**: ใช้ชื่อตัวแปรที่ชัดเจนและ indentation ที่เหมาะสม
4. **Comments**: เพิ่ม comments อธิบาย logic ที่ซับซ้อน
5. **Readability**: ให้ความสำคัญกับ code ที่อ่านง่ายและดูแลรักษาได้

เมื่อไหร่ควรใช้โครงสร้าง Conditional แต่ละแบบ:

ใช้ if-else เมื่อ:

- ต้องตรวจสอบช่วงหรือ conditions หลายตัว
- จัดการการตรวจสอบ input และกรณีข้อผิดพลาด
- Boolean logic ที่ซับซ้อนด้วย AND/OR operators

ใช้ switch-case เมื่อ:

- แมปค่าที่แยกส่วนไปยัง outputs (เช่น วัน, ตัวเลือกเมนู)
- จัดการกรณีที่คงที่หลายๆ กรณีอย่างมีประสิทธิภาพ
- อ่านง่ายกว่าสำหรับการเปรียบเทียบค่าที่แน่นอนหลายตัว

การทดสอบ Implementation ของคุณ:

1. รัน test cases ทั้งหมดเพื่อให้แน่ใจว่าถูกต้อง
2. ทดสอบ edge cases และ inputs ที่ไม่ถูกต้อง
3. ตรวจสอบว่า logic ของคุณจัดการทุก scenarios ที่ระบุไว้
4. ตรวจสอบว่า return types และรูปแบบตรงกันอย่างแม่นยำ
5. ตรวจสอบรูปแบบ boolean output ("True"/"False" ด้วยตัวอักษรใหญ่)

🚀 การเริ่มต้น

1. เปิด `StudentSolution.cs`
2. Implement แต่ละ method เริ่มต้นด้วย Example methods สำหรับฝึกหัด
3. รัน test cases เพื่อตรวจสอบ implementation ของคุณ
4. ไปยัง Examples, Level 1 แล้วตามด้วย Level 2 เมื่อ tests ทั้งหมดผ่าน

📝 ข้อกำหนดการส่งงาน

- เสร็จสิ้น 17 methods ทั้งหมดใน `StudentSolution.cs`
- ให้แน่ใจว่า test cases ทั้งหมดผ่าน
- Code ควรมี comments และปฏิบัติตาม best practices
- ใช้เฉพาะ `Debug.Log()` สำหรับ output

ขอให้โชคดีกับ assignment ของคุณ! 🎮 🥷