VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

INTRODUCTION TO COMPUTING - CO1005

ASSIGNMENT 1

THE INFINITY CASTLE BATTLE

Author: MEng. Mai Duc Trung, BEng. Nguyen Minh Tam

HO CHI MINH CITY, 11/2025

PROJECT SPECIFICATION

Version 1.0

1 Learning Outcomes

After completing this assignment, students will be able to:

- Effectively apply conditional statements, operators, and basic control structures in C++.
- Understand and simulate decision-making processes based on predefined rules and formulas.

2 Introduction

In the world of *Kimetsu no Yaiba*, the battle within the **Infinity Castle** is one of the fiercest encounters between the Demon Slayer Corps and Muzan Kibutsuji's Upper Moons. Students are required to write a C++ program that simulates four distinct scenes in this legendary battle, each demanding the application of formulas and logical conditions to determine the character's outcome.

This assignment helps students practice reading requirements, handling input, applying nested conditions, and producing clearly formatted output.

3 List of Variables and Meanings

Table 1 describes all the variables used in the program "Infinity Castle", including their data types and meanings.

4 Description

The story begins when all members of the Demon Slayer Corps are drawn into the Infinity Castle, controlled by Nakime. Inside, each Hashira faces a life-or-death battle against the Upper Moons, culminating in the final confrontation with Muzan Kibutsuji. Your program will simulate this through four main scenes.

Variable	Data Type	Meaning
`slayerLevel`	`int`	Demon slayer's level (1–10).
`hp`	`int`	Current health points of the demon slayer.
`breathingMastery`	`double`	Breathing Technique mastery level (0.0–1.0).
`hasTalisman`	`int`	Indicates if the slayer has a talisman to open the gate (0: no, 1: yes).
`timeOfDay`	`char`	Time of day, `D` = daytime, `N` = night.
`demonPresence`	`int`	Indicates if a demon is present at the gate (0: no, 1: yes).
`demonRank`	`int`	Rank of the Upper Moon (1–6).
`swordSharpness`	`double`	Sharpness of the Nichirin sword (0–100).
`allyCount`	`int`	Number of allies assisting in battle.
`bossHP`	`int`	Initial HP of the boss in Infinity Castle.
`totalDamage`	`int`	Total damage dealt by the slayer during the fight.
`specialMoveReady`	`int`	Indicates if the ultimate move is ready (0: not ready, 1: ready).

Table 1: List of variables and their meanings in the *Infinity Castle* program

Scene 1 – Power Classification

Compute the total power index of the demon slayer using the formula:

$$power = slayerLevel \times 10 + \frac{hp}{10} + breathingMastery \times 50$$

Classification rules:

- $power \geq 120 \Rightarrow$ Hashira
- $80 \leq power < 120 \Rightarrow$ Elite
- $power < 80 \Rightarrow$ Novice

Output format:

```
[Scene 1] Rank: <Hashira/Elite/Novice> (power = <value>)
```

Scene 2 – The Infinity Castle Gate

Check whether the gate can be opened based on the following rules:

- If `hasTalisman == 0` $\Rightarrow$ `Denied: No talisman.`
- If `timeOfDay` is not '`D`' or '`N`' $\Rightarrow$ `Warning: invalid timeOfDay.`

- If `timeOfDay == 'N'` and `demonPresence == 1` $\Rightarrow$ `Open silently.`
- Otherwise $\Rightarrow$ `Open cautiously.`

Output format:

`[Scene 2] <result>`

Scene 3 – Battle Strategy

Compute the battle advantage index:

$$adv = (101 - demonRank \times 15) + swordSharpness \times 0.4 + allyCount \times 5$$

Classify the strategy:

- $adv \geq 100 \Rightarrow$ `Engage head-on`
- $60 \leq adv < 100 \Rightarrow$ `Harass and probe`
- $adv < 60 \Rightarrow$ `Retreat and regroup`

Output format:

`[Scene 3] <strategy> (adv = <value>)`

Scene 4 – Final Outcome

Compute the boss's remaining HP:

$$finalHP = bossHP - totalDamage$$

Apply the following logic:

- If $finalHP \leq 0 \Rightarrow$ `Boss defeated! (finalHP = 0)`
- If $finalHP > 0$, `specialMoveReady == 1` and $finalHP \leq 50 \Rightarrow$ `Use special move to finish!`
- Otherwise $\Rightarrow$ `Withdraw to heal.`

Output format:

`[Scene 4] <result>`

5 Input Format

All variables are input on a **single line** in the following order:

```
slayerLevel hp breathingMastery hasTalisman timeOfDay demonPresence demonRank
        swordSharpness allyCount bossHP totalDamage specialMoveReady
```

Example Input:

```
8 950 0.8 1 N 1 3 72.5 2 420 380 1
```

Example Output:

```
[Scene 1] Rank: Elite (power = 119.0)
[Scene 2] Open silently.
[Scene 3] Harass and probe (adv = 93.0)
[Scene 4] Use special move to finish! (finalHP = 40)
```

6 Implementation Guidelines

- Use `iomanip` to format floating-point numbers (e.g., `setprecision(1)`).
- Each scene should be processed in the exact order described.
- All floating-point values should be displayed with one decimal place.
- Be careful with nested conditionals to ensure correct logical flow.

7 Additional Notes

- Students must complete the assignment independently; code sharing is strictly prohibited.
- Any form of plagiarism or cheating will be handled according to university policy.
- The instructor reserves the right to make all final decisions regarding grading and assessment.

——————————END——————————