**The German International University**
**Faculty of Informatics and Computer Science**
**Dr. Nada Sharaf**

**Programming III**, Winter Semester 2024/2025
**Milestone 02**

Deadline 17th of December 11:59 PM

# General Description

In this milestone, you are requested to implement the core game-play for the **Memory Match Card Game**. Two players will take turns revealing cards from a shuffled grid. Depending on the types of cards revealed, various scoring and game rules will be triggered. The game will continue until all pairs of cards are matched and hidden from the grid. The player with the highest score at the end of the game will be announced the winner, or the game will be a tie if both players have the same score.

# Functional Requirements

a) **Player Turns and Card Flipping:**

- The game will have **two players** who take alternating turns.
- On each turn, a player selects **two cards** from the grid by entering their coordinates.
- The selected cards are flipped and revealed to both players.

b) **Scoring Rules for Standard Cards:**

- If the two revealed cards are **standard cards** and they **match** as they hold the same number, the player:
  - Gains **1 point**.
  - Takes another turn.
  - Remove the matched cards from the grid
- If the revealed cards **do not match**, they are flipped back and kept in the grid, and the next player takes their turn.

c) **Handling Special Cards:**

- Bonus Card is represented in number **7**
- Penalty Card is represented in number **8**
- **Bonus Card Rules:**
  - If one **bonus card** is revealed (and the other is a standard card), the player gains:
    - **+1 point only**
    - The bonus card is then get **removed** from the grid and the standard card will be kept and hidden.
  - If **both revealed cards** are **bonus cards**, the player has two options:
    - **+2 points**.
    - Or gain **+1 point** and take another turn.
    - Both bonus cards will then be removed from the grid
- **Penalty Card Rules:**

- If one **penalty card** is revealed (and the other is a standard card), the player:
  - Loses **1 point**.
- If **both revealed cards** are **penalty cards**, the player has two options:
  - Lose **2 points**.
  - Or lose **1 point** and **skip the next turn**.

- **Bonus and Penalty Card Rules:**
  - If one bonus card is revealed and the other is a penalty card, they will not have effect on the game play or the score. Both cards will get **removed** from the grid.

d) **Game End and Winner Determination:**

- The game ends when all pairs of cards are matched and removed from the grid.
- The player with the **highest score** is declared the **winner**.
- If both players have the **same score**, the game will end in a **tie**, and the result will be announced accordingly.

# Class Requirements

a) **Card Class and Subclasses:**

- Implement `StandardCard`, `BonusCard`, and `PenaltyCard` with their respective scoring effects.
- Cards should have methods to reveal(), hide(), and display() its current state.

b) **Player Class:**

- The player class should manage all info related to the current player such as name and score.
- Add methods to increase or decrease the score based on the type of cards and their features.

c) **Deck Class:**

- Handle cards with all their functionalities such as reset the grid to * format, removing a card from the grid in a specified condition, etc..
- Responsible for displaying the grid.

d) **Game Class:**

- Implement the main game functionality to handle player turns, card flipping, and score updates throughout the gameplay.
- Implement logic for handling special card effects as described.
- Handle the logic for determining when the game ends and announce the winner.

# Game Flow

The flow of the Memory Matching Game is as follows:

a) **Initial Grid Visualization**: Display the grid with all cards hidden (represented by ***).

b) **Player Information**: Show the names and scores of both players, and specify whose turn it is.

c) **Player's Move**:

- Ask the current player to enter the coordinates (x, y) for two cards to reveal via console input (cin).
- Update the grid by revealing the selected cards and display the updated scores (if any points were awarded).

d) **Turn Change**:

- Prompt the current player to press `Enter` to hide the cards again.
- Switch turns between the two players.

e) **Continue Playing**: Repeat steps c and d until the grid is empty (i.e., all cards are matched and removed).

f) **Game End**: Once the grid is empty, announce that the game has ended.

g) **Announce the Winner**: Compare the scores of both players. Announce the player with the higher score as the winner, and display their information.

# Deliverables

- **Source Code Files:**
  - Header files for each class.
  - Multiple `.cpp` files are allowed for every class
  - A one single `main()` function in the main source file (.cpp) to initialize and run the game.
- **Documentation:**
  - Brief explanations of how each functionality was implemented.
  - Description of special card behaviors and how they affect gameplay.
- **Submission:**
  - Compress all the files into a single `.zip` file named: `TeamNumber_SubmitterName_ID.zip`.

# Notes

- Focus on correctly implementing the turn-based logic and handling the various card types according to their rules.
- Ensure proper memory management by dynamically allocating and deallocating memory for the cards.
- Provide clear output in the console for each action, including card flips, scoring updates, and end-of-game announcements.