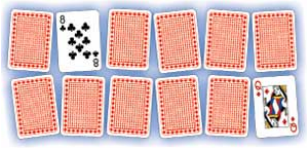
ENGG1110 Problem Solving by Programming (2018-2019 Term 2)
Project – Concentration (A Card Game)

---

1. Introduction

 [1]

Concentration, also known as *"Match Match"*[2], is a simple card game emphasizing a good memory that most of us should have played in our childhood. In this game, all 52 cards (jokers excluded) will be laid face down on a surface in orderly rows and columns. A group of players will take turns to pick two cards at a time and flip the cards to check whether they are of the same rank or not. If they are of the same rank, the two cards will be removed and kept by the player. If the cards are not of the same rank, they are turned face down again, and play passes to the next player. The game ends when the last pair has been picked up. The player who got the most of the cards is the winner. It is a game to practice one's memory as good memory can help retrieve those flipped cards which match in rank.

In this project, your task is to implement the Concentration game using C language. The program supports two modes: 1. Player vs. Computer, and 2. Computer vs. Computer.

In Mode 1, the program lets the player enter the coordinates of two cards via the console. Error checking should be implemented to make sure the entered coordinates are valid. Then two cards will be flipped and the matching result is shown. After that, the computer will pick randomly two cards and check whether they match or not. The above process will repeat until all cards have been flipped face up. At that time, the total number of successfully flipped cards by both players and the winner, if any, will be displayed.

Mode 2 is similar to Mode 1 except that two computer players, i.e. computer 1 and computer 2, will be used for the whole process. This is actually a simulation. Finally, the winner will also be displayed as is the case of mode 1.

2. Simplified Rules of the Game

To ease your code design and implementation, we have simplified the rules of the game as follows:

I. **Two Players Only**: you can always assume that there are only two players (whether human or computer) playing this game though in reality the game can support more than two players.
II. **Matching Ranks Only**: Some game rules require the matched pair to be in the same color (e.g. six of hearts ♥ = six of diamonds ♦, queen of clubs ♣ = queen of spades ♠) besides of the same rank. For simplicity, we won't consider color matching but rank matching only.
III. **One Flip**: The original game rules may reward the player who succeeds in matching a pair with one more turn to flip cards and continue until hitting a mismatch. To make the control flow simpler, this rewarding scheme is removed. After a player finishes flipping of two cards, no matter if they match or not, it will be the another player's turn to flip cards.

---

[1] Image source: https://s.hswstatic.com/gif/how-to-play-concentration-1.jpg.jpg
[2] https://en.wikipedia.org/wiki/Concentration_(game)

3. Requirements

You are required to use the following techniques in this project:

I. **Structures in C** to represent a card,

II. **Random number generator** to construct moves from a computer player

Important: Failure to apply both the above techniques will result in mark deduction.

The desktop surface should be shown as a 4 by 13 rectangular array as follows. Note that the cards are initially represented as "XXX" (card face down; value hidden).

```
XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX
XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX
XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX
XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX  XXX
```

When the game reaches the end, it may look like this picture:

```
C J   D 8   C 2   C 3   D Q   D10   D K   S 3   S 6   C10   C 7   D 6   C 8
H 7   D 7   H J   H K   S10   S A   S 7   D 3   D A   D J   H 4   D 2   S 5
C Q   H 6   S 4   H 8   H A   D 4   S Q   H 3   C 4   H 9   S 8   S 2   C 9
C 5   S J   C 6   S 9   H Q   H10   H 5   C A   D 5   D 9   H 2   S K   C K
```

All cards have been flipped face up, showing each card's rank and suit. Note the following scheme for representing the rank and suit of each card in the above display:

Rank	Shown as
Ace	A
2 - 10	2 - 10
Jack	J
Queen	Q
King	K

Suit	Shown as
Club ♣	C
Diamond ♦	D
Spade ♠	S
Heart ♥	H

Other detailed requirements on the display and the game logic will be covered in later sections.

4. Detailed Requirements Specification

Your program should perform the following tasks.

I. Display the program menu

Your program should print the following main menu for this game to the screen:

```
*****************************
* Welcome to Concentration! *
*****************************
Please choose a mode of the game:
1. Human vs. Computer
2. Computer vs. Computer
0. Quit
Your choice:
```

The user will choose whether he or she wants to play with the computer or just sees the simulation of the game between two computer players.

Important note:
You may assume that the user will only enter a single integer but input validation against valid choices is still required here. Your program should keep looping to display this menu and prompt the user for input until a valid menu choice (i.e. 1 or 2 or 0) is entered.

II. Human vs. Computer Mode (II-VI)

If menu option 1 is chosen, the program should display the deck of cards and let the player input the coordinates of the first card and then the second card being flipped.

```
****************************
* Welcome to Concentration! *
****************************
Please choose a mode of the game:
1. Human vs. Computer
2. Computer vs. Computer
0. Quit
Your choice: 1

XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX

Please input 1st card:
```

III. The player should then input the coordinates.

```
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX

Please input 1st card: 0 1
You chose 1st card (0, 1) and it is D 8
Please input 2nd card: 2 2
You chose 2nd card (2, 2) and it is D Q
```

The program prompts for the row index (0-3) and column index (1-13) of the first card being picked. It echoes the input and reveals the selected card's suit and rank. It does the same for the second card being picked.

Error checking should be done to prevent the following conditions:

1. Invalid coordinates

Ensure that the input row and column indexes are within 0 - 3 and 1 - 13 respectively (special: we start from 1 instead of 0 for column). If it is not the case, the program should prompt the user to enter again. A screenshot of an invalid case is shown as follows:

```
Please input 1st card: 0 18
Row and column must be within 0 - 3 and 1 - 13
Please input 1st card:
```

Note: For simplicity, you may assume that the user always enters two integers (no non-digit characters or floating-point numbers). So, your validation is just to check the input against the specified valid range of integer values for the coordinates.

2. The chosen card must not have been flipped before.

For example, in the following snapshot of the game, the two cards located at coordinates (1, 2) and (3, 4) have been flipped face up. They cannot be chosen anymore. If the human player chooses either of them, the error message "The card must not be flipped already" should be displayed.

```
XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX
XXX    H 6    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX
XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX
XXX    XXX    XXX    D 6    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX

(Some output is skipped here for ease of discussion.)

Please input 1st card: 1 2
The card must not be flipped already
Please input 1st card:
```

3. The second card must not be the same as the first

If the human player enters the same coordinates for both the first and second cards being picked, the program should show an error message "The second card cannot be the same as first" and prompt for the input of the second card again.

```
Please input 1st card: 0 12
You chose 1st card (0,12) and it is H K
Please input second card: 0 12
The second card cannot be the same as first
Please input second card:
```

Note: The computer player picks cards by generating a pair of random integers within the valid row and column bounds. If it encounters an already flipped card or the second card being same as the first, it will automatically drop the selection and generate a new choice until it is valid. So, we won't show all these error messages for a computer's turn.

IV. The result of the human player should be displayed. The screenshot below shows that if the player successfully removed two cards from the desktop, the face values of the removed cards will be kept displayed at their original positions till the end of the game.

```
Please input 1st card: 1 2
You chose 1st card (1, 2) and it is H 6
Please input 2nd card: 3 4
You chose 2nd card (3, 4) and it is D 6
You got a matched pair!

XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX
XXX    H 6    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX
XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX
XXX    XXX    XXX    D 6    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX    XXX
```

(Recall from Section 2 that the player who successfully matched a pair will NOT be rewarded one more turn to flip cards in this project. So, it is now the computer player's turn.)

V. The computer will then make its choice for a pair of cards to flip. The choice made by the computer can be purely random pick (the simplest form of implementation). You may also take on a challenge to implement a smart one (see the appendix; note: developing a smart algorithm for this part is NOT a mandatory requirement of this project and does not count for extra marks. But it is a good self-learning exercise to enhance your programming skills).

Based on the last example, after the human player finishes his/her flip, the computer makes its selection of two cards on the desktop. The selected cards' positions and face values will be printed as shown in the screenshot below. But this time, the pair does not match in rank, and the cards remain face-down (shown as "XXX").

```
Please input 1st card: 1 2
You chose 1st card (1, 2) and it is H 6
Please input 2nd card: 3 4
You chose 2nd card (3, 4) and it is D 6
You got a matched pair!

XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     H 6     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     D 6     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX

Computer chose 1st card (0, 2) and it is H J
Computer chose 2nd card (0, 8) and it is C 2

XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     H 6     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     XXX     XXX     D 6     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX

Flipped cards: 2; You: 2; Computer: 0;

Please input 1st card:
```

Now, both players have finished one turn each. The state of the game, mainly the statistics of flipped cards (i.e. cards facing up) made by the human (You) and the machine (Computer), will be updated and shown accordingly. In this example, the human player got two cards matched. So the counts next to "Flipped cards" and "You" are both incremented by 2.

The game continues with the human and machine players taking turns to flip pairs. Below is a later snapshot of the game in which the computer player successfully matched a pair. The flipped card statistics is updated each time a matched pair is made.

```
XXX     XXX     XXX     XXX     D 2     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
H Q     H 6     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     D Q     XXX     D J     XXX     XXX     S 2     D A     XXX     XXX     S A     XXX     XXX
D 3     XXX     XXX     D 6     XXX     C J     S 3     XXX     XXX     XXX     XXX     XXX     XXX

Computer chose 1st card (0, 9) and it is C10
Computer chose 2nd card (3, 2) and it is D10
Computer got a matched pair!

XXX     XXX     XXX     XXX     D 2     XXX     XXX     XXX     C10     XXX     XXX     XXX     XXX
H Q     H 6     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
XXX     D Q     XXX     D J     XXX     XXX     S 2     D A     XXX     XXX     S A     XXX     XXX
D 3     D10     XXX     D 6     XXX     C J     S 3     XXX     XXX     XXX     XXX     XXX     XXX

Flipped cards: 14; You: 12; Computer: 2;

Please input 1st card:
```

VI. Successive pick by the computer and human player will continue until all cards have been flipped. At the end, the program will show a game over message telling the total number of turns played (one turn refers to the flipping of two cards no matter if they match or not) and either one of the following messages:

- "You win!"
- "Computer wins!"
- "A tie!"

to indicate whether the human player or the computer has won the game, or it is a tie. Below is a sample screenshot near the end of a game.

```
...
(Prior output is skipped here for ease of discussion.)
...

Computer chose 1st card (0,13) and it is H K
Computer chose 2nd card (1, 4) and it is D K
Computer got a matched pair!

S 7     D A     D Q     C 2     H Q     S 8     D 9     S 2     C 6     C 3     H A     S 9     H K
D 7     S 5     S J     D K     C Q     XXX     D10     S 3     H J     C 9     S 6     C 7     H 9
D 6     D 3     D 5     S 4     D 4     XXX     C 5     D J     C K     C J     S K     D 2     H 2
H 4     C A     C 4     S10     S A     H10     H 6     H 3     H 5     C10     H 7     S Q     C 8

Flipped cards: 50; You: 28; Computer: 22;

Please input 1st card: 2 6
You chose 1st card (2, 6) and it is D 8
Please input 2nd card: 1 6
You chose 2nd card (1, 6) and it is H 8
You got a matched pair!                                                    Final desktop configuration
S 7     D A     D Q     C 2     H Q     S 8     D 9     S 2     C 6     C 3     H A     S 9     H K
D 7     S 5     S J     D K     C Q     H 8     D10     S 3     H J     C 9     S 6     C 7     H 9
D 6     D 3     D 5     S 4     D 4     D 8     C 5     D J     C K     C J     S K     D 2     H 2
H 4     C A     C 4     S10     S A     H10     H 6     H 3     H 5     C10     H 7     S Q     C 8

Flipped cards: 52; You: 30; Computer: 22;      Final game statistics

*** End of Game ***
429 turns played
You win!                                       Game over message


Please choose a mode of the game:
1. Human vs. Computer
2. Computer vs. Computer
0. Quit
```

VII. The program then will return to the main menu.

Note: the program does not end after a game finishes. It returns to the main menu to let the human player decide whether to play another game. The program terminates only when the menu option 0 is entered. During the course of a game (whether in mode 1 or mode 2), there is no special input to end the game prematurely. You may press the "Abort" button in the Code::Blocks IDE, or press Ctrl + C to terminate the running process.

VIII. Computer vs. Computer Mode

If menu option 2 is chosen, the program runs in a simulation mode. Two computer players will pick their choices of card pairs in turns successively like in Mode 1, but with all choices automatically generated (without any user attention).

Note: When running in Mode 2, we want to make it run faster. So, you should skip printing the state of the desktop but only the choices made by each player in each turn. Only the initial and final state of the desktop will be displayed. Below is an example screenshot segment showing the very last part of a game run.

```
****************************
* Welcome to Concentration! *
****************************
Please choose a mode of the game:
1. Human vs. Computer
2. Computer vs. Computer
0. Quit
Your choice: 2
```

Initial desktop configuration

```
XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX
XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX
XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX
XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX   XXX

Computer 1 chose 1st card (0, 2) and it is H J
Computer 1 chose 2nd card (0, 8) and it is C 2
Computer 2 chose 1st card (1, 1) and it is H Q
Computer 2 chose 2nd card (2, 9) and it is S 6
Flipped cards: 0; Computer 1: 0; Computer 2: 0;

...
(Lengthy output is skipped here for ease of discussion.)
...

Computer 1 chose 1st card (1, 1) and it is H Q
Computer 1 chose 2nd card (3, 5) and it is C Q
Computer 1 got a matched pair!
Computer 2 chose 1st card (0, 8) and it is C 2
Computer 2 chose 2nd card (2,13) and it is D K
Flipped cards: 46; Computer 1: 20; Computer 2: 26;

Computer 1 chose 1st card (2,13) and it is D K
Computer 1 chose 2nd card (0, 5) and it is D 2
Computer 2 chose 1st card (0, 7) and it is H K
Computer 2 chose 2nd card (2,13) and it is D K
Computer 2 got a matched pair!
Flipped cards: 48; Computer 1: 20; Computer 2: 28;

Computer 1 chose 1st card (0, 8) and it is C 2
Computer 1 chose 2nd card (3, 9) and it is H 3
Computer 2 chose 1st card (0, 8) and it is C 2
Computer 2 chose 2nd card (0, 5) and it is D 2
Computer 2 got a matched pair!
Flipped cards: 50; Computer 1: 20; Computer 2: 30;

Computer 1 chose 1st card (3, 1) and it is D 3
Computer 1 chose 2nd card (3, 9) and it is H 3
Computer 1 got a matched pair!
```

In this example, when Computer 1 picks the last matching pair, no more cards can be chosen by Computer 2 from the desktop. The game is over. The program should print the final desktop configuration, the final game statistics, and the end of game message telling who has won.

Final desktop configuration

```
D 8   H J   S10   C 7   D 2   C 6   H K   C 2   C10   H 5   H 2   C K   D 4
H Q   H 6   D 7   S 9   S K   H 8   S 4   S 5   D 5   S 7   S Q   H 9   S 8
C 5   D Q   C 3   D J   C 8   H A   S 2   D A   S 6   C A   S A   D 9   D K
D 3   D10   C 4   D 6   C Q   C J   S 3   H 4   H 3   C 9   S J   H10   H 7

Flipped cards: 52; Computer 1: 22; Computer 2: 30;
```

Final game statistics

```
*** End of Game ***
319 turns played!
Computer 2 wins!
```

Game over message

```
Please choose a mode of the game:
1. Human vs. Computer
2. Computer vs. Computer
0. Quit
Your choice:
```

5. Academic Honesty and Declaration Statement

Attention is drawn to University policy and regulations on honesty in academic work, and to the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details may be found at https://www.cuhk.edu.hk/policy/academichonesty/.

Please put the following declaration statement as the comment in the beginning of your source code (source file(s) with .c extension) and fill in your information.

```
/**
* ENGG1110 Problem Solving by Programming
*
* Course Project
*
* I declare that the project here submitted is original
* except for source material explicitly acknowledged,
* and that the same or closely related material has not been
* previously submitted for another course.
* I also acknowledge that I am aware of University policy and
* regulations on honesty in academic work, and of the disciplinary
* guidelines and procedures applicable to breaches of such
* policy and regulations, as contained in the website.
*
* University Guideline on Academic Honesty:
* https://www.cuhk.edu.hk/policy/academichonesty/
*
* Student Name : <your name>
* Student ID : <your student ID>
* Class/Section : <your class/section>
* Date : <date>
*/
```

6. Testing Platform

Your submission will be graded by using the Code::Blocks IDE on Windows. Please note that there may be some problems in opening a project on Windows if the project is created in other operating systems, such as MacOS and Linux. Therefore, make sure that your project is created on a Windows platform.

7. Submission

The deadline of the project submission is Apr 15 (Mon) 23:59. Please follow the following steps to submit your work.

a. Compress your whole Code::Blocks project folder into a file in ZIP format named as: ENGG1110<your class/section>_<your student ID>.zip (e.g. ENGG1110G_1155012345.zip)

b. Visit Blackboard for CUHK and log in with your OnePass (CWEM) password.

c. Visit the page for ENGG1110 and go to Project / Project Submission.

d. Upload and submit your file prepared in Step a.

e. Download your submission from Blackboard to see if it can be extracted and then opened by Code::Blocks on Windows successfully.

Resubmissions are allowed. But only the latest one will be graded. 10% of the project marks will be deducted for late submissions within one week, i.e., by Apr 22 (Mon) 23:59. Late submissions more than one week will not be graded.

Reminder: Double-checking by step e is an important step that you can't afford missing. We won't (be able to) grade submissions that cannot be opened successfully on our systems. In any disputable case, we stick to the built-in zip file extractor on Windows 10 as the reference for judgment. Please also note that submissions in other archive formats like .rar or .7z will not be accepted, and requests to resubmit and override a wrong/corrupted file after the hard deadline will not be entertained.

8. Project Demo and Presentation

The project demo and presentation is scheduled on **Apr 17 (Wed) in the lesson**. It is conducted in **English**. Each student will need to:

1. Compile and execute the program on a lab computer using **Code::Blocks** on **Windows**.
2. Explain the code and the program design

This part contributes 10% of the project marks. Marks for this part will only be given during the project demo and presentation. A student who is **absent** on that day will **get 0 marks for this part**. Therefore, even if your project is incomplete or if you plan to submit the project late, you should still come and try to get some marks.

9. Schedule

To help you with the development, we split the development process into the following stages.

Week	Date (Wed)	Tasks
10	Mar 20	• Structure definition for cards • Display the whole deck of cards • Shuffling of cards using a random number generator By the end of this week, your program should be able to: • Display the whole deck of cards as a rectangular array • Shuffle the deck of cards and display them
11	Mar 27	• User input of coordinates • Error checking for input • Computer pick generation By the end of this week, your program should be able to: • Play through a complete mode 1 game • Display the winner
12	Apr 10	• Mode 2 logic (computer vs. computer) By the end of this week, your program should be able to: • Perform all functions of mode 1 and mode 2
13	**Apr 15**	• Project submission (by 23:59 on Blackboard)
	Apr 17	• Project Demo and Presentation (in class)

10. Some Hints and Recommendation

I. Your source code should be properly indented and formatted with consistent styles. They are important to make your program readable and easier to trace and debug.

II. Effective use of functions can help modularize your program, saving a lot of redundant code in this project. For example, Mode 1 and Mode 2 are pretty similar except for the way we get the choices (from the standard input stream vs. from a random number generator) and for the way we show the output (Mode 2 skipped printing the state of the desktop). Think about how to make good of functions for better code reuse. Good use of functions could also reduce the complexity or level of some deeply nested loops.

Appendix – Optional Features (Self-Study)

Revisiting Section 4 point V, apart from purely random pick, you may also take on an optional challenge to implement some artificial intelligence (AI) for the computer player to make smarter pick of cards. We recommend that you may attempt this advanced part only when you have finished the baseline version using simply random pick.

The kind of AI needed for this game is relatively simple; things as complex as machine learning are supposed not involved. The basic idea is to add some array for recording those cards that were once turned over (whether flipped by the human player or a machine). As we represent a card as a C structure, the positions and face values of the cards can be found by reading the array elements. When it is the computer's turn to flip cards, instead of random pick, it tries to look up from this array for a matching pair. This array storage mimics the memory of a human being. Each machine player should have its own copy of this array. The size of this array denotes the memory power of the player, say using an array of 10 elements can only "memorize" the information of 10 cards (still facing down) on the desktop. You may control the "smartness" or "memory power" of each machine player by tuning its own array's size.

As you may also notice, one good strategy to use with Concentration is not to always turn over the card you are sure of first. For example, if you know exactly where one Ace of Hearts is, but are not 100% sure where another Ace is located, you will turn over the "uncertain card" first. This way if your guess goes wrong, you can then pick a random card and have a chance of getting a match. A smart algorithm using the abovementioned array could also implement this strategy. Instead of picking a card that has been recorded in the array, the machine should pick an unknown one from the desktop for the first card. If its rank can match with any of the memorized cards (in the array), then this is a successful move. Otherwise, the machine should try to flip a new unknown to maximize the chance of memorizing new cards by saving them in the array (until the array is already full).

One more reminder is that you should remember to update players' arrays according to the latest changes of the desktop. Say, if the current player's array has stored some cards which have been removed from the desktop by the opponent player, then they should be also removed from the array, freeing up storage space for new cards to be memorized.

We, when assessing your work, can observe the AI by looking at the number of turns (in other words, the length of output in the console) required to reach the end of the game in Mode 2. Work with AI implemented should demand much fewer turns to finish a game.

Another possible experiment is to verify the hypothesis that a machine player with "better memory" (bigger array) is more likely to win over one with "poorer memory" (smaller array).

- END -