MCD4720 - Fundamentals of C++

Assignment 2 and 3 - Trimester 2, 2021

Assignment Submissions

This project will be submitted in two parts. Both parts of the assessment are equally important for the successful completion of your project, so it is essential that you understand the requirements of both parts before you start.

- **Assignment 2: Mastermind (Part A: Project Plan)**

  Due Date: August 10, 2021, 11:55PM (Week 7)

  Marks: This assignment will be marked out of 100 points.

  Weighting: 10% of your final mark for the unit.

  This assignment is the first part of a larger project, which you will complete in Assignment 3. This task consists of your project planning documentation. It will include details of the requirements & analysis of your program, including UML Class diagrams.

  The purpose of this assignment is to get you comfortable with planning a C++ programming project for Assignment 3. The task is detailed later in this assignment specification, as are the specific marks allocation.

- **Assignment 3: Mastermind (Part B: C++ Project Implementation)**

  Due Date: September 3, 2021, 11:55PM (Week 10)

  Marks: This assignment will be marked out of 100 points.

  Weighting: 20% of your final mark for the unit.

  This assignment consists of your implementation of your project, as outlined in your Project Planning document (Assignment 2).

  Your project must follow your project plan and must be submitted as a Visual Studio project, including all header and .cpp files, and any appropriate text files to ensure the program compiles and runs.

  This assignment consists of one Application file and associated custom Class files. The purpose of this assignment is to get you comfortable with designing and implementing basic multi-class C++ programs. The task is detailed later in this assignment specification, as are the specific marks allocation.

Late submission:

- By submitting a Special Consideration Form or visit this link: https://lms.monashcollege.edu.au/course/view.php?id=1331

- Or, without special consideration, you lose 5 marks per day that you submit late (including weekends). Submissions will not be accepted more than 14 days late for assignment 2 and submissions will not be accepted more than 14 days or after week 12 whichever is earlier for assignment 3.

  This means that if you got *Y* marks, only (Y-n×5) will be counted where *n* is the number of days you submit late.

Marks: This assignment will be marked out of 100 points, and count for 10% of your total unit marks.

Plagiarism: It is an academic requirement that the work you submit be original. If there is any evidence of copying (including from online sources without proper attribution), collaboration, pasting from websites or textbooks, **Zero marks** may be awarded for the whole assignment, the unit or you may be suspended or excluded from your course. Monash Colleges policies on plagiarism, collusion, and cheating are available here or see this link: https://www.monashcollege.edu.au/__data/assets/pdf_file/0010/17101/dip-assessment-policy.pdf
Further Note: When you are asked to use Internet resources to answer a question, this **does not mean copy-pasting text** from websites. Write answers in your own words such that your understanding of the answer is evident. Acknowledge any sources by citing them.

Submission Instructions:

This project will be submitted in two parts:

- **Assignment 2 – Part A:** consists of your project planning documentation.

  This document will include an outline of your program structure and UML Class diagrams.

  The assignment must be created and submitted as a single Word or PDF document to the Moodle site. This document must clearly identify both your Name and Student ID to facilitate ease of assessment and feedback.

  Your document file **MUST** be named as follows:

  "***YourFistNameLastNameID_A2***.docx" or "***YourFistNameLastNameID_A2***.pdf".

  This file must be submitted via the Moodle assignment submission page.

  The document should contain the project plan and the UML diagrams. You can use Microsoft Visio to draw the UML diagrams or you can use any other software, provided that the diagrams are included in your submitted document.

  Explicit assessment criteria are provided, however please note you will also be assessed on the following broad criteria:

  ✓ Detail of a proposed project plan for the overall project.
  ✓ Creating accurate and complete UML diagrams.
  ✓ Applying a solid Object-Oriented Design (OOD) for the overall project
  ✓ Using appropriate naming conventions, following the unit Programming Style Guide.

- **Assignment 3 – Part B:** consists of your implementation of your game project.

  Your project must follow your project plan and must be submitted as a Visual Studio project, including all header and code files, and any appropriate text files to ensure the program compiles and runs.

  You may complete the tasks in your preferred IDE, however you **MUST** create a Visual Studio project in order to submit. Your project folder must be identified by using your name and assignment number, such as ***YourFirstNameLastNameID_A3.***

  The entire project folder must then be zipped up into one zip file for submission. The zip file **MUST** be named "***YourFistNameLastNameID_A3***.zip". This zip file must be submitted via the Moodle assignment submission page.

  Explicit assessment criteria are provided, however please note you will also be assessed on the following broad criteria:

  ✓ Meeting functional requirements as described in the assignment description
  ✓ Demonstrating a solid understanding of C++ concepts, including good practice
  ✓ Demonstrating an understanding of specific C++ concepts relating to the assignment tasks, including object-oriented design and implementation and the use of Pointers
  ✓ Following the unit Programming Style Guide
  ✓ Creating solutions that are as efficient and extensible as possible
  ✓ Reflecting on the appropriateness of your implemented design and meeting functional requirements as described in the assignment description

NOTE! Your submitted program MUST compile and run. Any submission that does not compile will automatically awarded **a 50 marks penalty**. This means it is your responsibility to continually compile and test your code as you build it.

NOTE! Your submitted files must be correctly identified and submitted (as described above). Any submission that does not comply will receive an automatic **20 marks penalty** (applied after marking). For assignment 3, your Visual Studio project should include all .cpp and .h files.

If you have any questions or concerns please contact your tutor as soon as possible.

Assignment 2: Pairs (Part A)

You are to implement a computer-based variant of the card game Pairs. This is a card game for 2-8 players. The basic game of Pairs is a simple press-your-luck game with no winner, just one loser. Players score points by catching pairs, or by folding. The first player with too many points is the loser!

You can watch a video on how to play here: https://www.youtube.com/watch?v=bq7Em3p7oS0 or https://www.youtube.com/watch?v=rcV45WyiWqs or online rules here:

https://cheapass.com/wp-content/uploads/2018/02/PairsCompanionBookWebFeb2018.pdf

In your version, for the basic assignment you only need to implement a 2-player game.

For Part A (Assignment 2) of the assignment you will focus on the planning of the project. In Part B (Assignment 3) you will focus on creating the various interactive objects in the game and program the player interactions.

Basic Game Play:

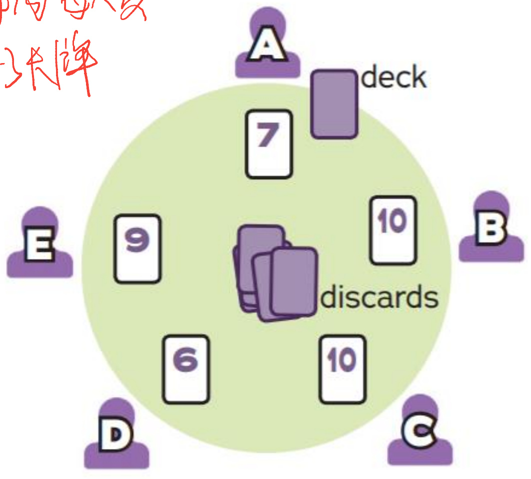
In this program, the computer (dealer) controls a triangular deck of cards. The deck is made up of 1x1, 2x2s, 3x3s, all the way up to 10x10s, making a deck of 55 cards. This is NOT a standard deck of playing cards.

The basic game play is as follows:

- Shuffle the deck and burn (discard) five cards, facedown, into the middle of the table. This creates the start of the discard pile. Each time you reshuffle, you will burn five cards. This makes it harder for players to guess what cards are left at the end of the deck.
- To start each round, deal one card face up to each player. The player with the lowest card will go first. If there is a tie for lowest card, deal a second card to the tied players to break the tie. If the second card creates a pair, discard it and deal another.

  每局 每人发一张牌

  - Example: In the diagram at right, Player A is dealing. She shuffles the deck and burns five cards into the middle. She then deals one card to each player, face up. Player D will go first because she has the lowest card, a 6.



- **On your turn:** you have two choices – you may hit (take a card), or fold (end the round).
  - If you get a pair, or fold, the round is over and you score points.
  - If not, play passes to the left.

如果有一对，此轮结束，分数为对子牌牌面值

- **Hitting:** When you take a hit, you're hoping not to get a pair. If you catch a pair, the round ends, and you score points equal to the rank of the paired card.
  - For example, if you catch a pair of 8's, you score 8 points. Keep one of those cards aside face up, to track your score.
- **Folding:** You can surrender (fold) instead of taking a card. This also ends the round. When you fold, you must take the lowest card in play and keep it for points. You may choose this card from any player's stack, including your own.

可以选所有人的牌

- Folding can be better than hitting, depending on your odds of catching a pair, but it's up to you to decide when to do it.

一轮结束，弃掉所有牌

- **Ending the Round:** As soon as one person catches a pair or folds, the round is over. Discard all the cards in players' stacks, face down into the middle, and deal another round.
  - Scoring cards (those cards that were kept aside for points) are not discarded.

判断牌是否发完

- **Reshuffling**: When the deck runs out, just shuffle and continue. Pause the deal, reshuffle the discards, and resume dealing where you left off. (Remember to burn five cards.)

- **Losing the Game:** There is no winner, just one loser. The game ends when one player reaches the target score. This number changes with the number of players. The formula for 2 to 6 players is: (60 ÷ number of player) + 1. For 7 or 8 players, keep the value at 11.

Players:	**2**	**3**	**4**	**5**	**6+**
Score:	*31*	*21*	*17*	*13*	*11*

  - For example, in a 4-player game, the loser is the first player to score 16 points.

平局 两张一样的最小牌

- **Breaking a Tie:** If 2 or more players are tied for the lowest card, deal additional cards to the tied players, and use those cards a tie-breakers. If the tie-breakers are tied, deal more cards until one player has the lowest card.

  If someone gets a pair during this process, discard the paired card and deal a replacement. Players cannot be knocked out by a pair during this process, although they can sometimes wind up with several extra cards!

- **Dealing:** deal cards in a consistent order, starting with the first player, and continuing in clockwise order around the table. Deal tie breaking cards in the same order.

- **Pair:** any 2 cards in a single player's stack that have the same rank – the cards do not have to be next to each other to be a pair. Other players may have the same value card but this is not classed as a pair in this game. 每个玩家有2张相同点数的牌就能凑成一对

There are many variations to this basic game, some of which you will be able to implement as part of the extra functionality for your assignment.

Project Plan

Having a clear plan for your project before you begin coding is essential for finishing a successful project on time and with minimal stress. So, part of this assignment is defining what will be in your project and what you need to do to actually develop it.

> ***Important:*** *You must also read the requirements for Assignment 3 in order to be able to complete the documentation required for Assignment 2.*

The documentation you must submit will include the following:

✓ **A brief description for the setting of your text adventure.**

This could be the information you have on the initial screen of your game to set the atmosphere and inform the player what they need to do in order to win the game.

You can use the introduction above (in *italics*) as a template for your description.

Note: This description must be saved as a text file and read into your game when the program begins. You can also use this as a convenient way to display "help" if the player asks for it.

✓ **A development outline of your game**

Using a simple outline format (numbered or bullet points) state the main actions that the program will have and then, as sub-points, state the things you will need to do to make that happen.

The outline structure should contain all the elements of your game, as this is a high-level description of your approach to the development of your program. You should include at least the following headings and provide examples of happens under each section.

- The game setup (everything that happens before the game starts)
- The player's turn (the sequence of events that happen during a turn)
- Processing player input (include each of the commands your player can use)
- Providing feedback to the player (in response to the player's interactions)
- The end game conditions (include all win and lose conditions)
- Additional Features included, if any – see Assignment 3
- Outline the functionality of all your game classes – see Assignment 3

Here is an example to get you started with your project outline:

- The Game Setup
  - Display an overview of the game rule so the player knows what to do to win.
    - read this information from a text file
  - Initialise the game elements:
    - add the player – ask for the player's name, set default variables
    - all the other things that will happen during initialisation including
      - creating the characters and the game world
      - initialising other game variables (list them here)

As you can see, you only have to describe the actions the program will take, not the code, for the outline. The idea here is to give you a starting point for when you start writing your code as you can use this as a checklist of the things you need to include.

✓ ***UML* Diagrams**

UML diagrams are designed to make structuring your program easier. How to create them will be covered in class, but the general structure is shown here – see Assignment 3 for more details about classes.

ClassName
list of attributes (variables)
list of behaviours (functions)

You will need UML diagrams for each of the classes you include in your game – at least a Suspect, Item and Application (main) class.

Assignment 2: Marking Criteria [100 marks in total]

1. *Requirements and Analysis Document (Project Plan) [80]*

1.1. Description of the rules (the introduction to your game) **[8]**

1.2. Outline includes all game functionality (from Part B) **[12]**

1.3. Each section is broken into logical tasks **[15]**

1.4. Tasks are performed in a logical order **[15]**

1.5. Task descriptions given provide sufficient detail **[30]**

2. *UML Diagrams [20]*

2.1. Correct structure used (Name, Attributes, Behaviours) **[4]**

2.2. Included the correct designations for public (+) and private (-) data **[8]**

2.3. All variables and functions have meaningful names **[8]**

Assignment 3: Pairs (Part B)

You are to implement the Pairs Card Game you started in Assignment 2 by creating a Visual Studio Project using your project plan as described in your previous submission.

Your completed Pairs card game must demonstrate the following:

✓ You MUST implement your program using the following classes, as a minimum, you may include more (as appropriate for your game design):

- **Player class:** holds the player's details including their name, current score and a collection of cards (the player's stack in the game).
- **Card class:** holds the card's details including its rank, a visual representation of the card and its status – in the deck, discarded, dealt to or held by a player.
- **Application file:** holds the main() function and controls the overall flow of the game.

You may include other relevant attributes and behaviours to these classes, as identified in your project plan.

✓ The **Player** must be able to do the following:

- assign a name which is requested at the start of the game and used in the feedback given
- choose to take a card (hit) or fold and see appropriate feedback as a result
- continue playing until the round ends – someone gets a pair or folds
- quit the game at any time – during or after a game

✓ The **Cards** in the game should have the following characteristics:

Vector

- have a value from 1 to 10
- if the card is dealt to a player or discarded, it should be unable to be used again
- display a visual representation (eg: [1] = a 1 card, [10] = a 10 card) when dealt to a player

✓ The **Game Application** must do the following: Main方法 .

- display the "how to play" information at the start of the game
- create the players and a deck of 55 cards consisting of 1x1, 2x2s, 3x3s … 10x10s
- display an appropriate and uncluttered user interface providing relevant information to the player at all times
- ask for and allow the player enter an option to take a card or fold (taking any face up card)
- display the updated player hands after each card is dealt – all players' cards are visible at all times
- terminate the game (player loses) when the player has reached or passed the target
- provide player stats at the end of the game (wins, loses and score)
- the player should be able to QUIT the game at any time

Program Reflection

You must also provide a 300-word written reflection of your object-oriented design and how well you believe it was to implement. You should cover the following areas:

- Discuss why you designed it the way you did.
  - Why did you create your classes that way?
  - How does this reflect an OO design or approach?
- Discuss how well you were able to code it
  - Highlight any issues you found once you tried to implement your design.
  - How did you resolve these issues?

- If you were to do this project again, discuss how you might change your design to make your solution easier to implement, more efficient, or better for code reuse.

This must be a Word or PDF document which must be included in the same folder as your **.sln* file. Your document file **MUST** be named as follows:

“***YourFistNameLastName_A3***.docx” or “***YourFistNameLastName_A3***.pdf”.

Extra Functionality

The marking criteria indicates that you should make some individual additions to this in order to achieve the final 20% of the mark.

Following is a list of additional features you can include, with the maximum number of marks **[x]** you can earn for each one. You may implement one or more features from the list, but you will only score up to the maximum of 20% of the total assignment marks or 20 marks.

You should aim to add some additional creative elements to the gameplay, as well as advanced object-oriented design elements or advanced use of pointers.

- The player can set the number of cards that are “burned” (discarded) after each shuffle (for example 4, 5, 6, or a random number from 4 to 6). **[2]**
- Use chips to keep score. Each player starts with 4-12 chips (2x the number of players). At the end of each game the loser must pay one chip to every player. Once a player loses all their chips, they are out of the game. The last player with all the chips is the winner. **[3]**
- Display the cards using ASCII art. You can use different images related to a theme or “pip” patterns for the numbers. **[4]**
- Allow the game to be saved and restored at the player’s request. **[4]**
- Create a Hand class to hold all the details of each player’s hand of cards for each round. A player’s hand is emptied at the start of each round. The Hand class must incorporate into the Player class. **[5]**
- Continuous Pairs alternative game play option. The players must select which version of Pairs to play at the start of the game. This version is almost the same as basic Pairs, but it is a single long round, rather than several short ones. Whenever a player pairs up or folds, *only that player’s cards are discarded*, and the game continues. The player whose cards were just cleared is still in the game, with an empty stack and everyone else keeps their cards. There 2 rule modifications: 1) when you have no cards you automatically hit, and 2) when you fold, you may take any card in play (it doesn’t have to be the lowest). **[5]**
- Allow the game to be played with one or more computer players. The player can select the number of opponents to play against. The computer players should make reasonably intelligent choices based on the visible cards in the other player’s cards. **[10]**
- Implement a more sophisticated AI for the computer players. You may create different player types, each with their own strategies for playing, such as an easy player, a cautious player, an aggressive player, etc. The human player can select the opponent’s difficulty level and number of their opponents at the beginning of the game. **[10]**
- For the advance assignment you need to implement a 3 or more-player game. At the beginning of the game you should enter the number of players (2-8). 2 for basic and 3+ for advance. **[10]**

You certainly do not have to implement all of the above to earn marks for extra functionality. Just remember the maximum number of marks you can earn are given in [x]. It is up to you!

Assignment 3: Marking Criteria [up to 100 marks in total]

Does the program compile and run? Yes or No

- **50 marks penalty will be applied for a non-compiling program.**

1. Class Design [15]

1.1. Player Class **[5]**
- 1.1.1. Has an appropriate header file [2]
- 1.1.2. Required data members and member functions using meaningful names [1]
- 1.1.3. Contains only aspects that relate to a “player” (has no data members or member functions that are not directly related to a Player) [2]

1.2. Card Class **[5]**
- 1.2.1. Has an appropriate header file [2]
- 1.2.2. Required data members and member functions using meaningful names [1]
- 1.2.3. Contains only aspects that relate to a “board” (has no data members or member functions that are not directly related to the board) [2]

1.3. Game Application **[5]**
- 1.3.1. Has an appropriate header file [2]
- 1.3.2. Has appropriate variables and functions using meaningful names [2]
- 1.3.3. The main() function has appropriate function calls to keep it uncluttered [1]

2. Functionality [35]

2.1. Game set up: initialising the player, game variables and creating a collection of cards **[5]**
2.2. Implementation of a clear and uncluttered User Interface display **[5]**
2.3. Implementation of the basic game mechanics (hit and fold) **[5]**
2.4. Implementation of dealing and monitoring the cards to all players **[5]**
2.5. Implementation score tracking and updating as required **[5]**
2.6. Successful implementation of action processes and feedback displayed to the player **[5]**
2.7. Appropriate end game conditions triggered **[5]**

3. Quality of Solution and Code [20]

3.1. Does the program perform the functionality in an efficient and extensible manner? **[12]**
- 3.1.1. Appropriate use of functions and function calls [1]
- 3.1.2. Appropriate use of data types [1]
- 3.1.3. Appropriate use of decisions, loops and other programming techniques [2]
- 3.1.4. Appropriate use of references and/or pointers [5]
- 3.1.5. Appropriate use of good programming practices and techniques [2]
- 3.1.6. Extraneous and redundant code removed [1]

3.2. Has a well-designed OO program been implemented? **[4]**
- 3.2.1. Contains classes appropriate to the assignment brief [3]
- 3.2.2. Program structures support and OO design [1]

3.3. Has the Programming Style Guide been followed appropriately? **[4]**
- 3.3.1. Appropriate commenting and code documentation [2]
- 3.3.2. Correct formatting of code within *.h and *.cpp files [2]

4. Extra Functionality [20]

4.1. The player can set the number of cards that are “burned” (discarded) after each shuffle **[2]**
4.2. Use chips to keep score. No chips = out of game, winner takes it all **[3]**
4.3. Display the cards using ASCII art, using different images or pip patterns **[4]**
4.4. Allow the game to be saved and restored at the player’s request **[4]**
4.5. Create a Hand class incorporated into the Player class **[5]**
4.6. Continuous Pairs alternative game play option. **[5]**
4.7. Allow the game to be played with one or more computer players **[10]**
4.8. Implement a more sophisticated AI for the computer players **[10]**
4.9. For the advance assignment you need to implement a 3 or more-player game **[10]**

5. Reflection [10]

5.1. Discussion of motivations for the program design [3]
5.2. Discussion of how well the design was to implement [3]
5.3. Discussion of what they would do differently if they were to start it again [4]

Assignment Notes:

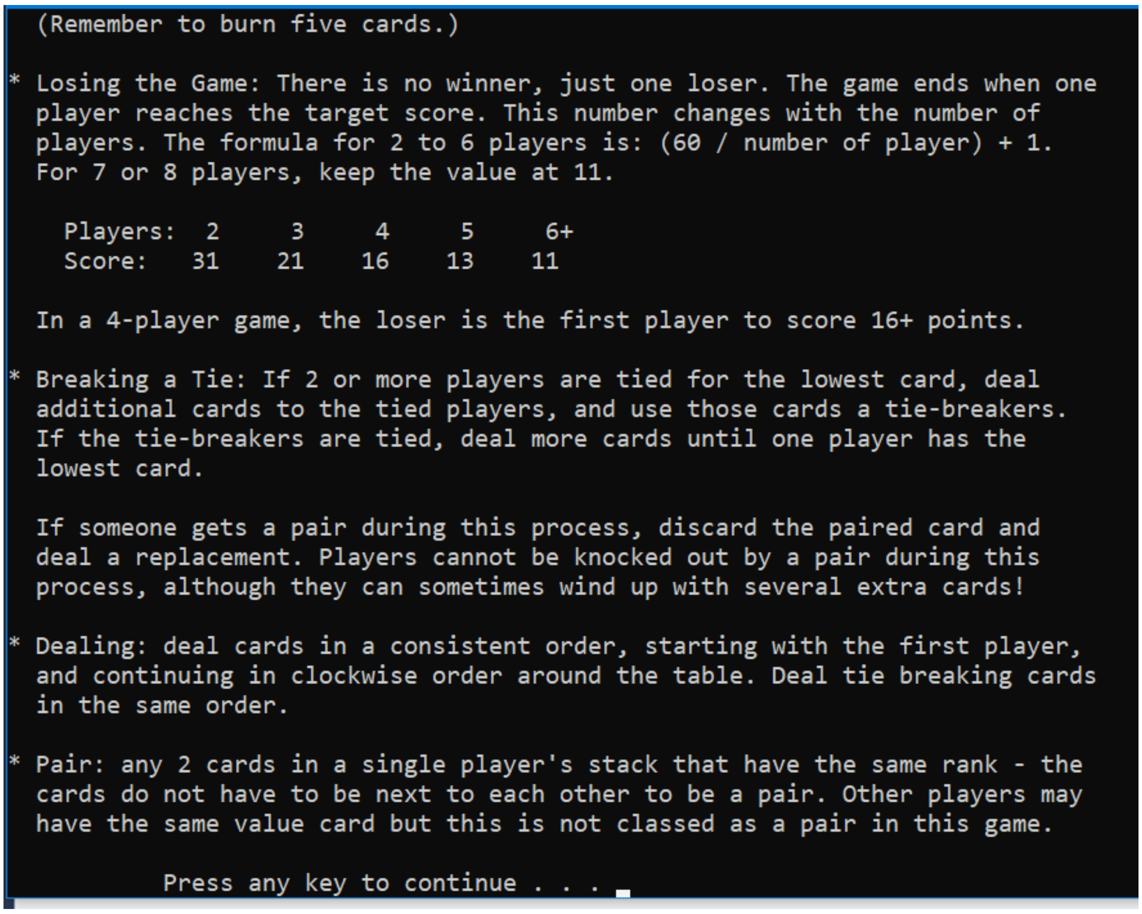
It is your responsibility to know what is expected of you. If you are unsure about anything, ask your tutor sooner rather than later. Write any questions and/or notes here to help you clarify what you have to do.

Note: Your tutor may ask you to program part of your assignment in class to determine that you have completed the work yourself. Failure to do this to an acceptable level will result in you being referred to the Subject Leader for plagiarism.

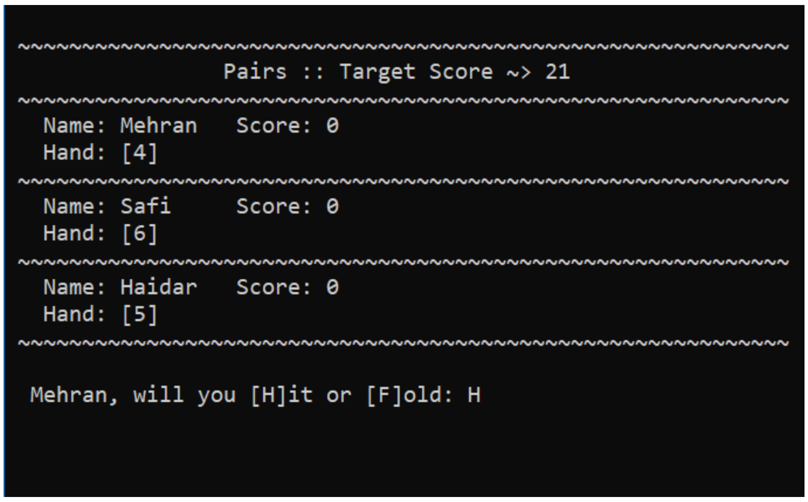
In-Class interviews: Also, you may be required to demonstrate your code to your tutor after the submission deadline. Failure to demonstrate will lead to zero marks being awarded to the entire assignment.

Note: Submitting only .sln file, a zero mark will be granted.

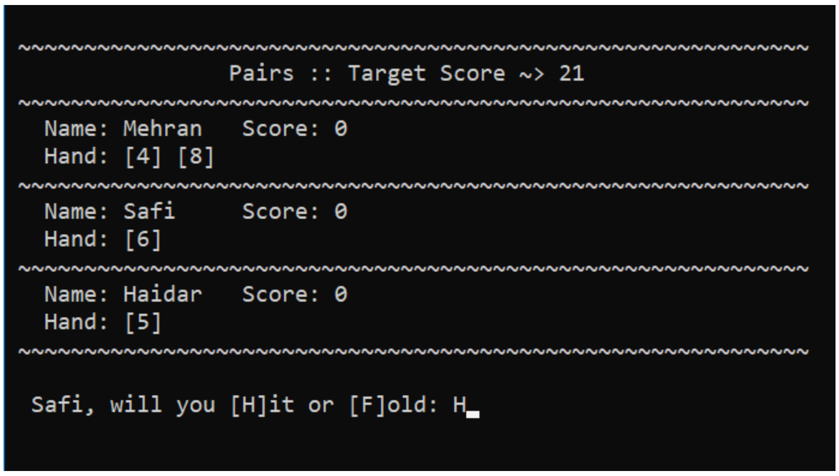
Here are some sample screen shots to help you develop your user interface:



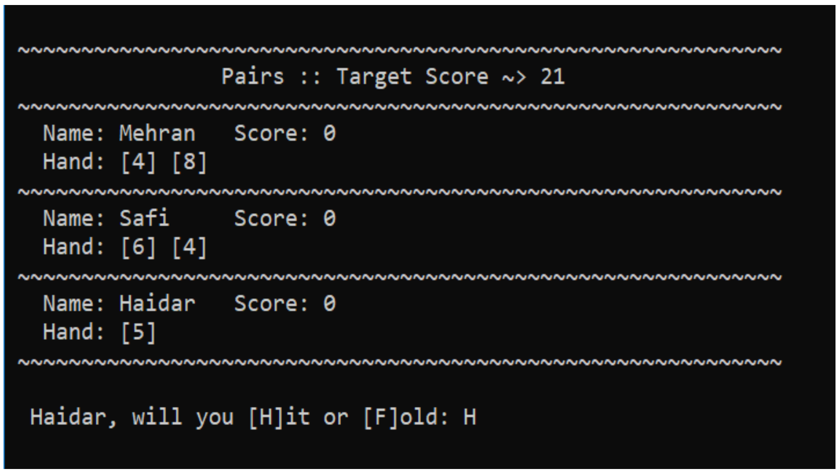
```
   (Remember to burn five cards.)

 * Losing the Game: There is no winner, just one loser. The game ends when one
   player reaches the target score. This number changes with the number of
   players. The formula for 2 to 6 players is: (60 / number of player) + 1.
   For 7 or 8 players, keep the value at 11.

     Players:  2     3     4     5     6+
     Score:   31    21    16    13    11

   In a 4-player game, the loser is the first player to score 16+ points.

 * Breaking a Tie: If 2 or more players are tied for the lowest card, deal
   additional cards to the tied players, and use those cards a tie-breakers.
   If the tie-breakers are tied, deal more cards until one player has the
   lowest card.

   If someone gets a pair during this process, discard the paired card and
   deal a replacement. Players cannot be knocked out by a pair during this
   process, although they can sometimes wind up with several extra cards!

 * Dealing: deal cards in a consistent order, starting with the first player,
   and continuing in clockwise order around the table. Deal tie breaking cards
   in the same order.

 * Pair: any 2 cards in a single player's stack that have the same rank - the
   cards do not have to be next to each other to be a pair. Other players may
   have the same value card but this is not classed as a pair in this game.

            Press any key to continue . . .
```



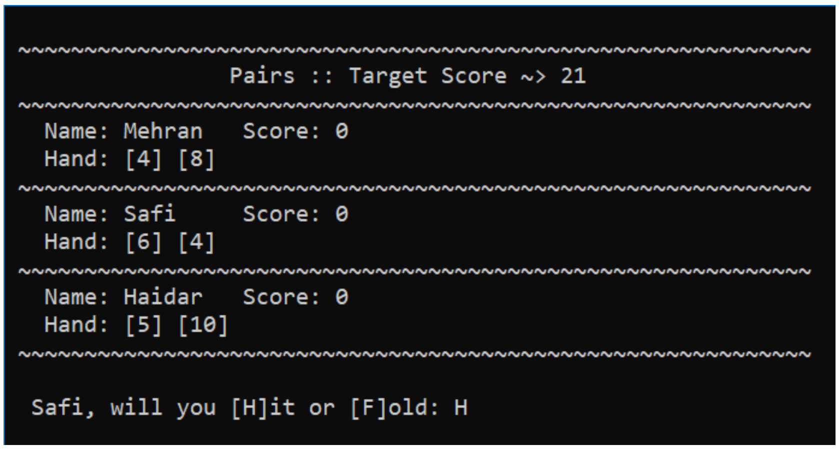
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Mehran, will you [H]it or [F]old: H
```



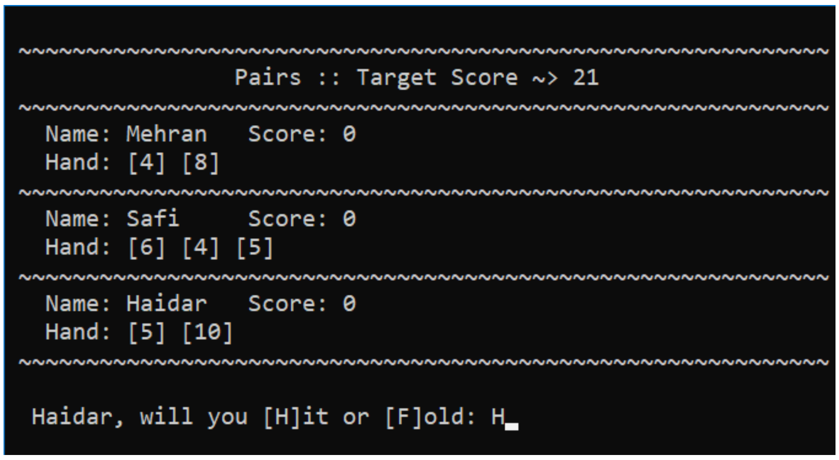
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Safi, will you [H]it or [F]old: H
```



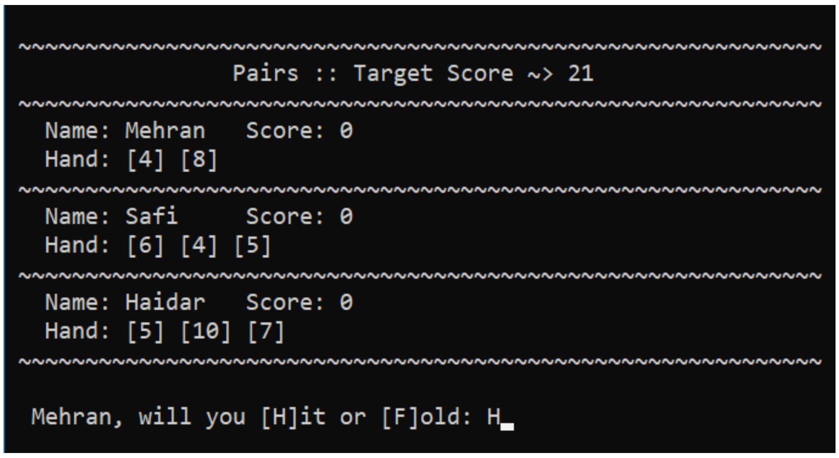
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Haidar, will you [H]it or [F]old: H
```



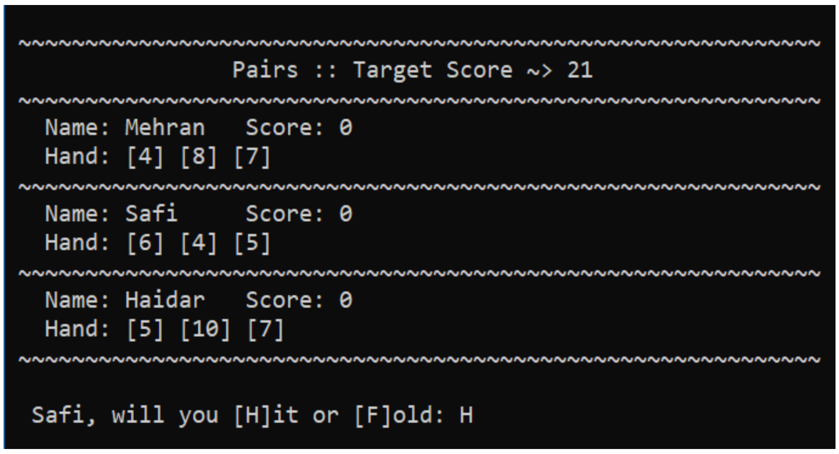
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5] [10]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Safi, will you [H]it or [F]old: H
```



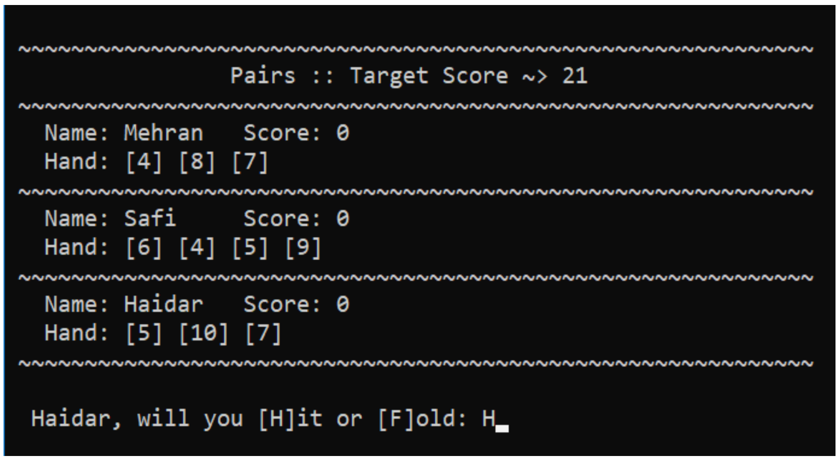
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4] [5]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5] [10]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Haidar, will you [H]it or [F]old: H
```



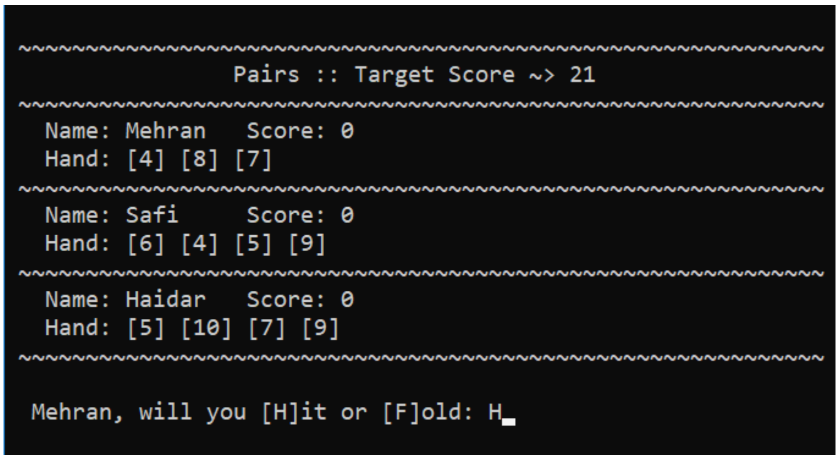
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4] [5]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5] [10] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Mehran, will you [H]it or [F]old: H
```



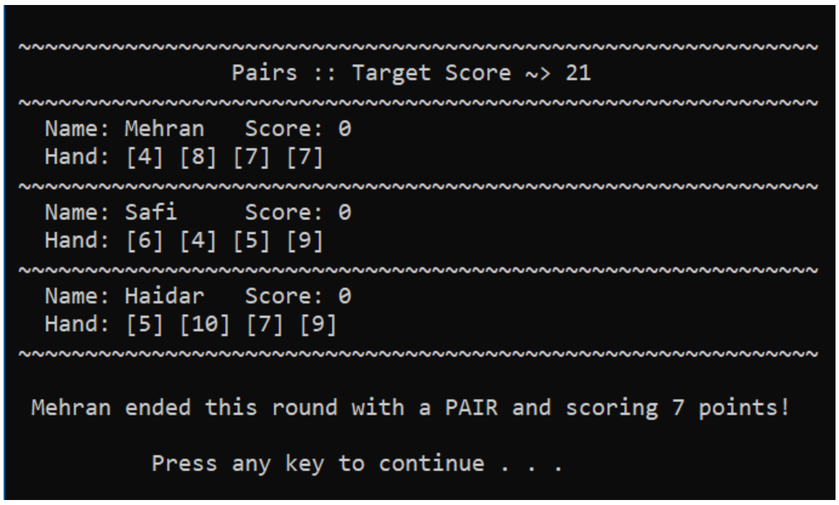
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4] [5]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5] [10] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Safi, will you [H]it or [F]old: H
```



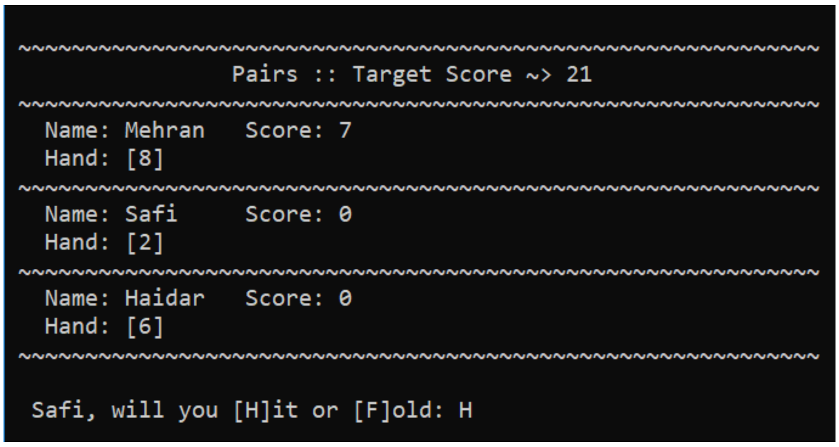
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4] [5] [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5] [10] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Haidar, will you [H]it or [F]old: H_
```



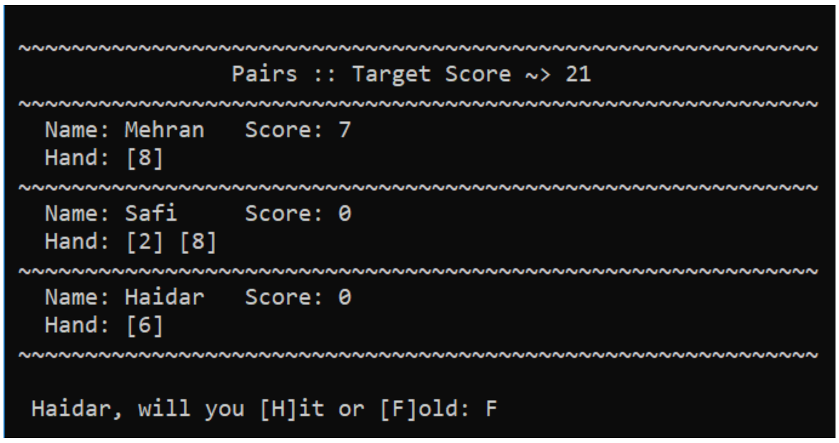
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4] [5] [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5] [10] [7] [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Mehran, will you [H]it or [F]old: H_
```



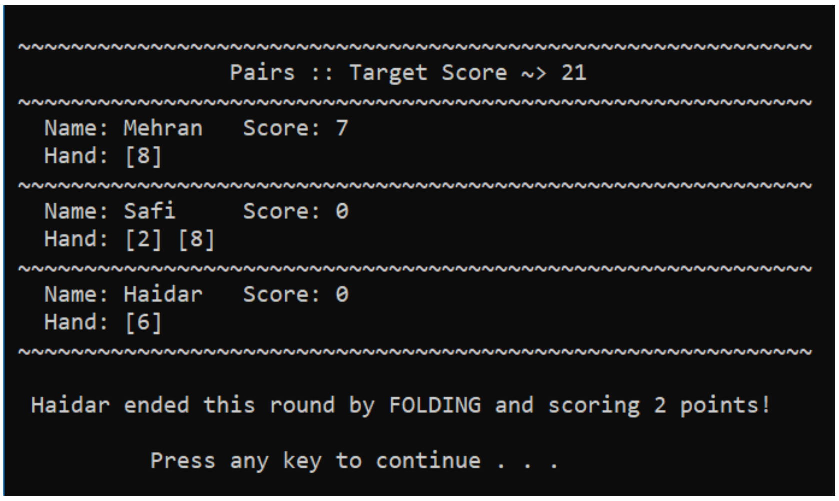
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 0
  Hand: [4] [8] [7] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [6] [4] [5] [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [5] [10] [7] [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Mehran ended this round with a PAIR and scoring 7 points!

          Press any key to continue . . .
```



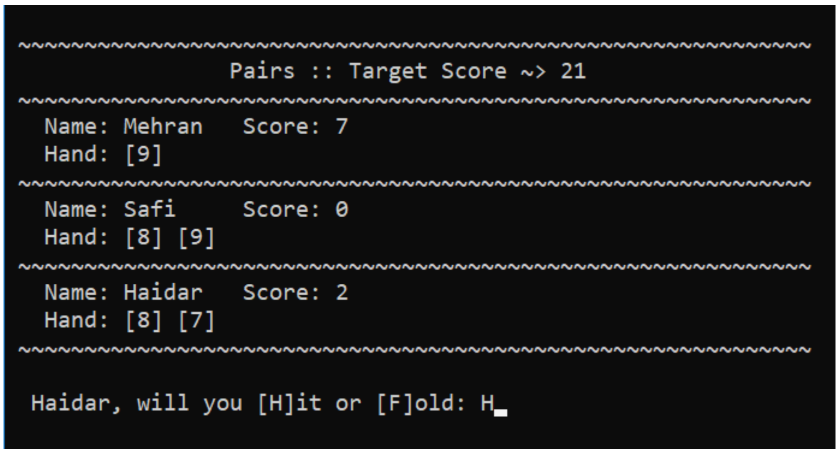
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 7
  Hand: [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [2]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [6]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Safi, will you [H]it or [F]old: H
```



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 7
  Hand: [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [2] [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [6]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Haidar, will you [H]it or [F]old: F
```



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 7
  Hand: [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [2] [8]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 0
  Hand: [6]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Haidar ended this round by FOLDING and scoring 2 points!

          Press any key to continue . . .
```



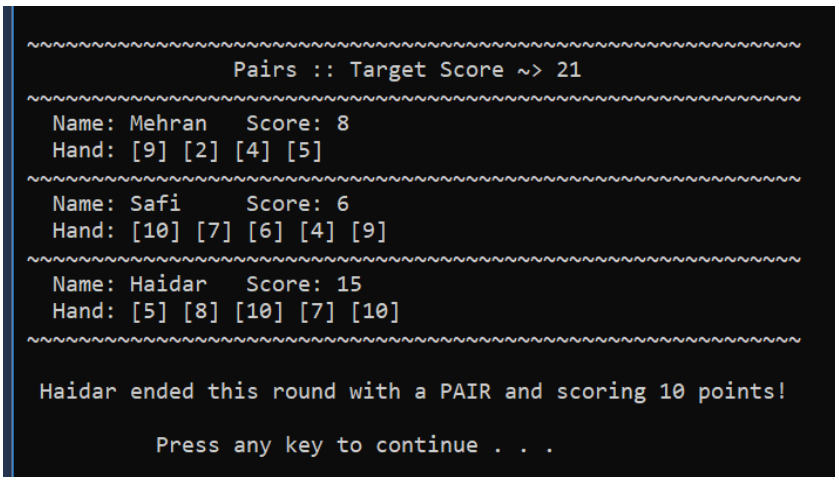
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 7
  Hand: [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 0
  Hand: [8] [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 2
  Hand: [8] [7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Haidar, will you [H]it or [F]old: H
```

▪

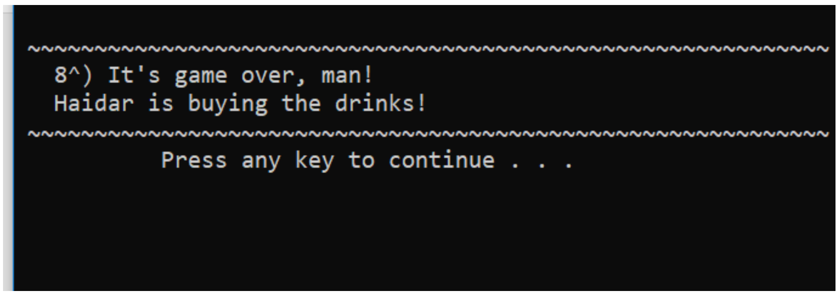
▪

▪

▪



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Pairs :: Target Score ~> 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Mehran   Score: 8
  Hand: [9] [2] [4] [5]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Safi     Score: 6
  Hand: [10] [7] [6] [4] [9]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Name: Haidar   Score: 15
  Hand: [5] [8] [10] [7] [10]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Haidar ended this round with a PAIR and scoring 10 points!

          Press any key to continue . . .
```



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  8^) It's game over, man!
  Haidar is buying the drinks!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          Press any key to continue . . .
```