MCD4720 - Fundamentals of C++

Assignment 2 and 3 - Trimester 2, 2020

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: Farkle (Part A: Project Plan)**

  Due Date: ~~August 1, 2020,~~ August 8, 2020, 11:55PM (Week 6.2)

  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: Farkle (Part B: C++ Project Implementation)**

  Due Date: September 4, 2020, 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 10 days late.

  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:

  "***YourFistNameLastName_A2***.docx" or "***YourFistNameLastName_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 "***YourFistNameLastName_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: Farkle (Part A)

You are to implement a computer-based variant of the dice game Farkle. This is a dice game for any number of players. Each player is racing to reach a target score by removing scoring combinations of dice from a set of 6 dice. The first player to reach or exceed the target score is declared the winner! Any number of players may play, however, for the basic assignment you only need to implement one.

You can watch a video on how to play here: https://www.youtube.com/watch?v=jBKdOF9tuog or even play an online version here: http://www.playonlinedicegames.com/farkle

In your version, for the basic assignment you only need to implement a 1-player game.

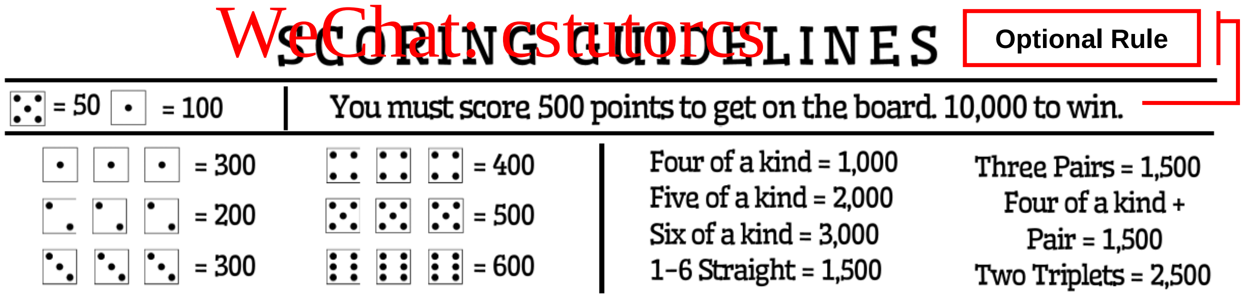
For Assignment 2 (Part A) of the assignment you will focus on the planning of the project. In Part B you will focus on creating the various interactive objects in the game and program the player interactions.

Basic Game Play:

In this program, you will control a set of six dice. On your turn, you must make as many scoring combinations (as outlined below) as you wish to score points towards the winning target score.

The basic game play is as follows:

- One player is playing with themselves each time must track their own score and display their name and game score when required
- On your turn:
  - Roll all six dice and set aside at least one die of a scoring value, (as shown below): Note you should follow the below scoring guidelines even the YouTube and the online game said something else.


SCORING GUIDELINES
Optional Rule
⚄ = 50 ⚀ = 100
You must score 500 points to get on the board. 10,000 to win.
⚀ ⚀ ⚀ = 300
⚁ ⚁ ⚁ = 200
⚂ ⚂ ⚂ = 300
⚃ ⚃ ⚃ = 400
⚄ ⚄ ⚄ = 500
⚅ ⚅ ⚅ = 600
Four of a kind = 1,000
Five of a kind = 2,000
Six of a kind = 3,000
1-6 Straight = 1,500
Three Pairs = 1,500
Four of a kind + Pair = 1,500
Two Triplets = 2,500

  Combinations only count when made in a single throw.

  - You may now decide whether to score your points and end your turn or you can re-roll the remaining dice.

    If you choose to roll again, you can keep rolling the dice, until you choose to stop or you roll no scoring values. Rolling no scoring values is called a FARKLE.

    If you roll a FARKLE, you score NO POINTS for this round and your turn ends. Pass the dice to the next player.

    If you set aside all 6 dice, you may re-roll all 6 dice and continue adding to your score, following all the rules above.

- The first player to score 10,000 or more points wins the game. However, for this assignment set a 5,000-point target score.

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.

A Typical Player Turn:

During a player's turn, the logic is as follows:

1) First, the player must roll all 6 dice to produce 6 random numbers from 1-6.
2) Next, they set aside at least one die that scores points (a single 1 or 5, or 3-of-a-kind). The points for which are stored in a running total for the player's turn. These dice cannot be rolled again, until after all 6 dice are set aside (see Step 4).
3) The player is presented with 2 options – roll the dice or score their points.

   If they choose to score, the running total is added to their player score and their turn ends.

   If they roll again, only those dice not set aside are rolled. Steps 2 and 3 may be repeated as often as the player wishes, until they use all 6 dice (Step 4) or roll a Farkle (Step 5).
4) If the player sets aside all 6 dice, they may choose to start again with all 6 dice and continue scoring as before. Steps 2 and 3 may be repeated as often as the player wishes.
5) A Farkle is a roll with no scoring values – the player cannot take a single 1 or 5, or 3-of-a-kind. When this happens, the player ends their turn immediately and does not score the points rolled during that turn. Their saved score is not changed.

Here is an example roll for a player's turn:

- On their first roll, the player rolls a 1, 2, 3, 3, 5, 5.
- They can score 100 points for the 1, or 100 points for both the 5s, or 50 points for 1 of the 5s, or 200 points for the 1 and both 5s.
- They choose to set aside just the 1 for 100 points and roll the 5 remaining dice again.
- This time they roll a 2, 3, 6, 6, 6. They decide to take the 600 points for the 3-of-a-kind in 6s.
- Their running total is now 700 points. This is a good score so they opt to save their points and end their turn.

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 description of "How to Play"**

This will be the information you display on the opening screen of your game to inform the player what they need to do in order to play and win the game. Do not just copy the game play description above, make your own 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 game board
      - 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.

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

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

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: Farkle (Part B)

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

Your completed Farkle dice 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 a player's details including their name, score and turns taken.
- **Die class:** holds the current die value, a visual representation of the die and whether it can be rolled (or not).
- **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 roll or score the dice rolled and see appropriate feedback as a result
- continue rolling dice until they win or lose the game
  - a win = reaching or exceeding 5,000 points
  - a loss = rolling 3 Farkles in a row
- quit the game at any time – during or after a game

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

- be set to a random number from 1 to 6 with each roll
- if the die is selected, it should be locked and unable to roll again
- be able to reset the die value to 0 at the end of a turn when the player scores their points
- display a rolled value and a locked version (eg: [1] = a rolled 1 and [x] = a locked die)

✓ The **Game Application** must do the following:

- display the “how to play” information at the start of the game
- create a player and a set of 6 dice
- 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 roll the dice or score the points
- ask for and allow the player to select one or more of the rolled dice
- display the updated dice after each roll entered by the player
- terminate the game (player wins) when the player has reached or passed the target
- terminate the game (player loses) when the player has rolled 3 Farkles in a row
- 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:

“***A3-YourFistNameLastName***.docx” or “***A3-YourFistNameLastName***.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 selects a skill level (eg: Rookie, Roller and High Roller) which modifies the game parameters – the target score (5,000, 10,000 and 20,000) and the minimum points required to roll before saving points (250, 500 and 750 respectivaly), etc. **[5]**
- If a player rolls three Farkles in a row (on three consecutive turns) they lose 1000 points from their saved score (can go negative). Once they lose the points the counter for Farkles is reset to zero. If a single player then rolling 3 Farkles in a row will lose the game if their score becomes negative. **[2]**
- Include multiple human players making the game playable by 1 to 4 people. At the beginning of the game the player order must be randomised. Each player must track their own score

and display their name and game score when required. One player is randomly chosen to go first. Players take turns in clockwise order. The game is played in rounds in which each player has a turn to roll the dice and score points. **[10]**

- Display the dice using ASCII art, showing the faces as pips not as numbers. **[5]**
- Allow the game to be saved and restored at the player's request. **[5]**
- Ask the player if they want to be promoted to the next skill level after winning 5 games. If they accept their skill level is increased by one (to the maximum level permitted in your game). Also, if the player loses 5 games in a row they are automatically demoted one skill level (to the minimum level permitted in your game). **[5]**
- When the first player reaches or exceeds the target score, all other players (in a multi-player game) have one final turn to try and beat that score. In this case, the player at the end of the round with the highest score wins. **[5]**
- Allow the game to be played with computer players. The computer player should be able to make reasonably intelligent decisions for choosing scoring combinations and when to save their points. This AI can be a set of rules that the computer player checks before making a roll or when to save their score. On a computer player's turn, the process should be automated so that the human player can watch the results. **[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 should be able to select the level of difficulty and number of their opponents at the beginning of the game. **[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. Die 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 dice **[5]**
   2.2. Implementation of a clear and uncluttered User Interface display **[5]**
   2.3. Implementation of the basic game mechanics (roll and score) **[5]**
   2.4. Implementation of basic scoring combinations (as shown on table) **[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. Implement difficulty levels which may be selected by the player **[5]**
   4.2. Rolling 3 Farkles in a row (on three consecutive turns) lose 1000 points **[2]**
   4.3. Include multiple human players making the game playable by 1 to 4 people **[10]**
   4.4. Display the dice using ASCII art, showing the faces as pips not as numbers **[5]**
   4.5. Allow the game to be saved and restored at the player's request **[5]**
   4.6. Ask the player if they want to be promoted to the next skill level after winning 5 games **[5]**
   4.7. All players (in a multi-player game) have an equal number of turns **[5]**
   4.8. Allow the game to be played with computer players **[10]**
   4.9. Implement a more sophisticated AI for the computer players **[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.

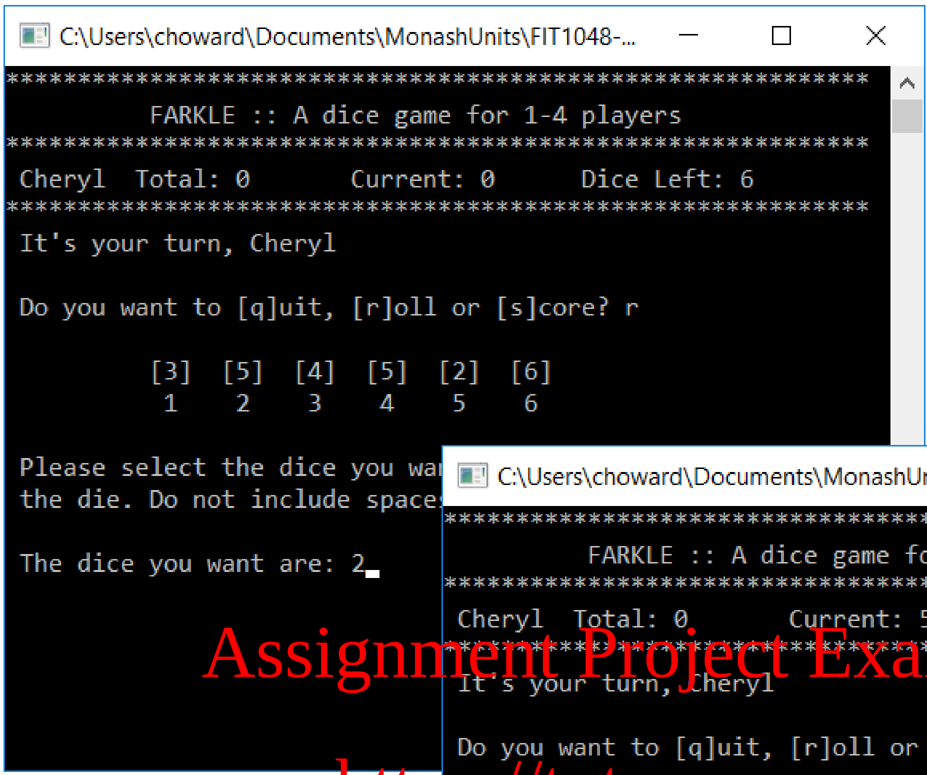
Note: Your tutor may ask you to program part of your assignment in class to determine that you have completed the work yourself. Failure to do this to an acceptable level will result in you being referred to the Subject Leader for plagiarism.

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:

Sample Screenshots to give you some ideas for your layout and interface design.

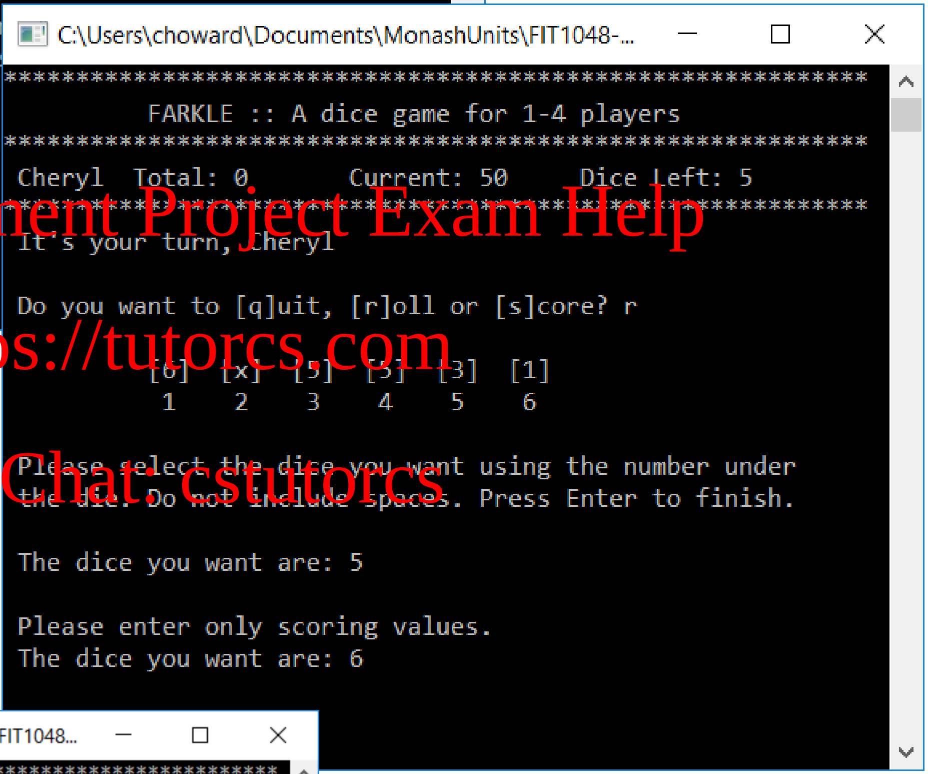

C:\Users\choward\Documents\MonashUnits\FIT1048-...

```
*****************************************************************
          FARKLE :: A dice game for 1-4 players
*****************************************************************
 Cheryl  Total: 0       Current: 0      Dice Left: 6
*****************************************************************
 It's your turn, Cheryl

 Do you want to [q]uit, [r]oll or [s]core? r

          [3]  [5]  [4]  [5]  [2]  [6]
           1    2    3    4    5    6

 Please select the dice you wa
 the die. Do not include space

 The dice you want are: 2_
```

Displaying information to the player is important.

This shot shows player's saved total, running total and how many dice to roll.

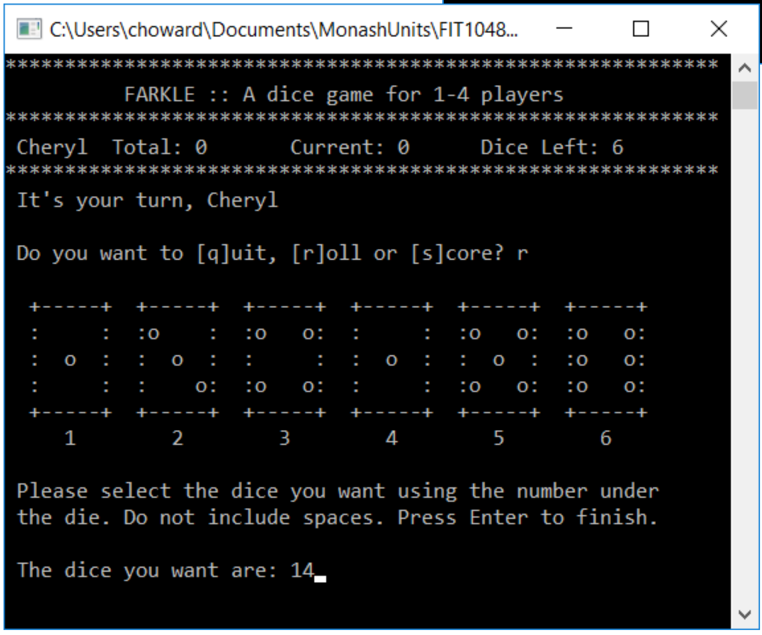
Note the use of prompt messages to help the player know what to do.


C:\Users\choward\Documents\MonashUnits\FIT1048-...

```
*****************************************************************
          FARKLE :: A dice game for 1-4 players
*****************************************************************
 Cheryl  Total: 0       Current: 50     Dice Left: 5
*****************************************************************
 It's your turn, Cheryl

 Do you want to [q]uit, [r]oll or [s]core? r

          [6]  [x]  [5]  [5]  [3]  [1]
           1    2    3    4    5    6

 Please select the dice you want using the number under
 the die. Do not include spaces. Press Enter to finish.

 The dice you want are: 5

 Please enter only scoring values.
 The dice you want are: 6
```

This shot shows the rolled dice and the locked die the player selected on their last roll. The dice values are in the [ ] to make them stand out from the selection ones.

Data validation is used to ensure the player enters


C:\Users\choward\Documents\MonashUnits\FIT1048...

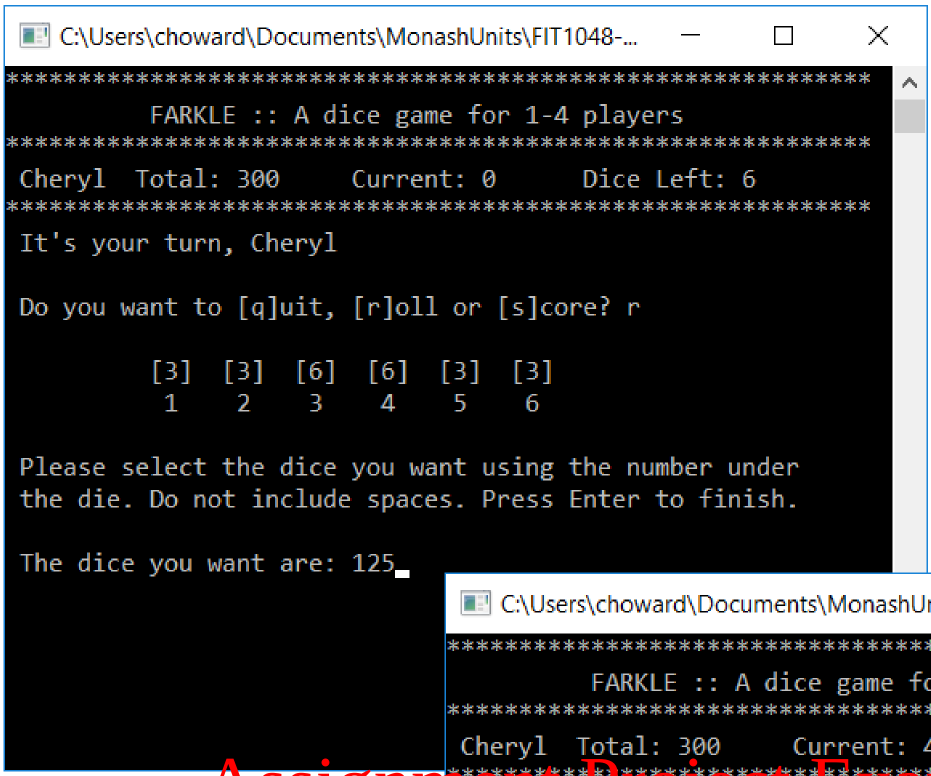
```
*****************************************************************
          FARKLE :: A dice game for 1-4 players
*****************************************************************
 Cheryl  Total: 0       Current: 0      Dice Left: 6
*****************************************************************
 It's your turn, Cheryl

 Do you want to [q]uit, [r]oll or [s]core? r

  +-----+  +-----+  +-----+  +-----+  +-----+  +-----+
  :     :  :o    :  :o   o:  :     :  :o   o:  :o   o:
  :  o  :  :  o  :  :     :  :  o  :  :  o  :  :o   o:
  :     :  :    o:  :o   o:  :     :  :o   o:  :o   o:
  +-----+  +-----+  +-----+  +-----+  +-----+  +-----+
     1        2        3        4        5        6

 Please select the dice you want using the number under
 the die. Do not include spaces. Press Enter to finish.

 The dice you want are: 14_
```

This shot shows the rolled dice using ASCII art to display the dice, as an alternative to the above interface.

Using this type of display will require putting some effort into formatting the individual dice faces then combining them into a suitable display.

The selection of the dice is the same as in the above example.

To score full marks for this "extra" you must display the dice across the screen, not down the screen below each other.

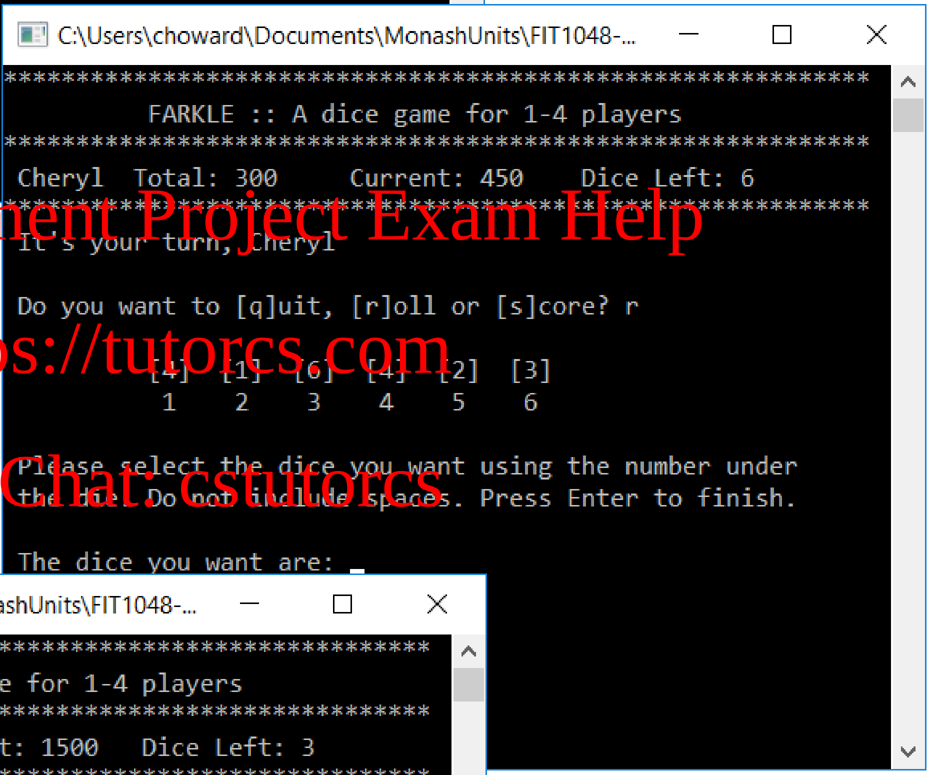


C:\Users\choward\Documents\MonashUnits\FIT1048-...

```
*************************************************************
          FARKLE :: A dice game for 1-4 players
*************************************************************
 Cheryl   Total: 300     Current: 0      Dice Left: 6
*************************************************************
 It's your turn, Cheryl

 Do you want to [q]uit, [r]oll or [s]core? r

          [3]  [3]  [6]  [6]  [3]  [3]
           1    2    3    4    5    6

 Please select the dice you want using the number under
 the die. Do not include spaces. Press Enter to finish.

 The dice you want are: 125
```

This shot shows the player's saved score and displays their next roll.

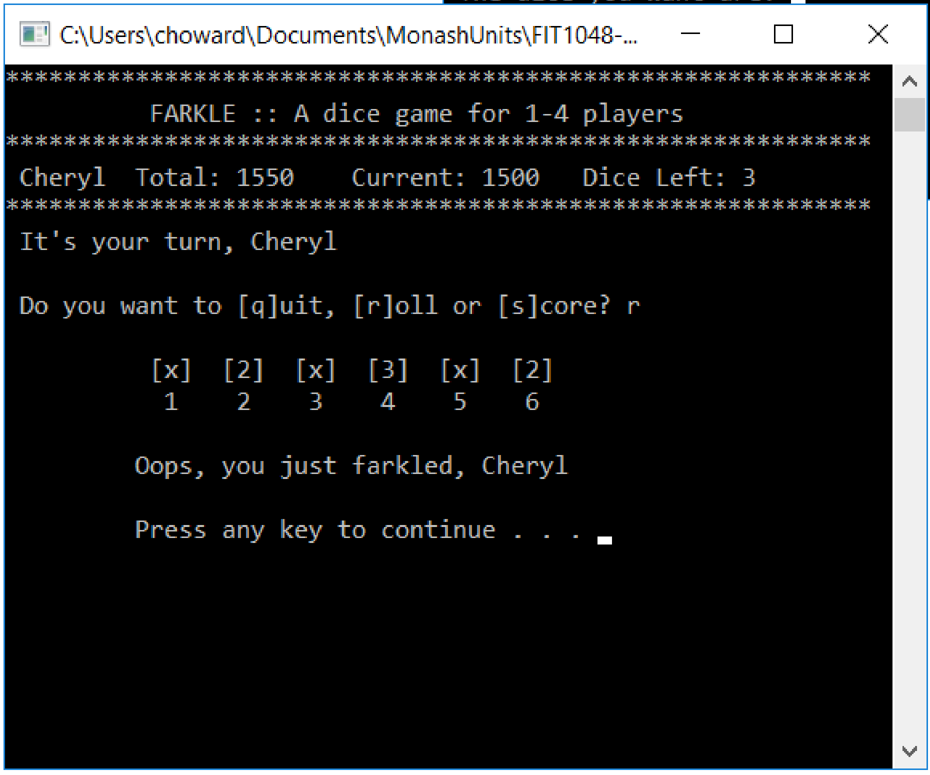
Selecting the dice is as easy as typing in the number of the required dice then processing each character in the string to calculate the players running total, based on the scoring system you have implemented.



C:\Users\choward\Documents\MonashUnits\FIT1048-...

```
*************************************************************
          FARKLE :: A dice game for 1-4 players
*************************************************************
 Cheryl   Total: 300     Current: 450    Dice Left: 6
*************************************************************
 It's your turn, Cheryl

 Do you want to [q]uit, [r]oll or [s]core? r

          [4]  [1]  [6]  [4]  [2]  [3]
           1    2    3    4    5    6

 Please select the dice you want using the number under
 the die. Do not include spaces. Press Enter to finish.

 The dice you want are:
```

This shot shows the player has rolled and scored all 6 dice so has the option to continue rolling with 6 dice.



C:\Users\choward\Documents\MonashUnits\FIT1048-...

```
*************************************************************
          FARKLE :: A dice game for 1-4 players
*************************************************************
 Cheryl   Total: 1550    Current: 1500   Dice Left: 3
*************************************************************
 It's your turn, Cheryl

 Do you want to [q]uit, [r]oll or [s]core? r

          [x]  [2]  [x]  [3]  [x]  [2]
           1    2    3    4    5    6

         Oops, you just farkled, Cheryl

         Press any key to continue . . .
```

This shot shows the rolled dice, all locked dice and that the player did not roll any scoring values – they Farkled!

This ends their turn and they do not score any points – they lose their current running total (in this case 1500 points … this is what happens when you get greedy).