*Blackjack Development – Testing and Driver Documentation*

**Initial Functions:** These functions make up a version that is barely playable.

**shuffleDeck** – creates a vector of 52 nonrepeating numbers from 1:52. Uses randperm to achieve this

Driver – tests in random variables to see if its repeated, display ‘found’ if found within the deck array. No test should be found twice. No test greater than 52 or less than 1 should be found.

Graphical user interface, application, Word

Description automatically generated

**cardFinder** – assigns a card; suit and number (or face) to a reference card based within the 52-card deck.

Driver – tests in random variables to see if the card outputted is the expected card.

Graphical user interface, text, application

Description automatically generated

**firstTurn (later removed)**– draws 2 cards for dealer and player, also sets the current card to be drawn as the fifth

Driver – tests two decks to see if the dealer and player hands are properly drawn and the current card to be selected ends at the fifth



**Hit (later called addCard)** – adds a card to chosen hand (can be used for dealer or player)

Driver – Tests by adding a third card to a hand of 2 cards



**Prototype 1 Driver** – Collates previous functions into a single program with some added output to make the game playable.

Testing - Impossible to use with test cases as the parameters will change for each game and is played against the computer which doesn’t reveal its cards until the end. Cases in which no Ace is drawn results in proper win status for both dealer and player. Cases in which an Ace IS drawn results in proper win status if both the dealer and player treat their aces as 1. However, there is a paradox in the fact that the dealer must choose to treat an initial Ace as 11 regardless so therefore is not truly correct.

|  |  |  |
| --- | --- | --- |
| **Input (Hit or Stand)** | **Expected Output** | **Actual Output** |
| Hit until bust | Bust | Bust |
| Hit (higher than dealer) | Win | Win |
| Hit (lower than dealer) | Loss | Loss |
| Hit (same as dealer) | Tie | Tie |
| Stand (higher than dealer) | Win | Win |
| Stand (lower than dealer) | Loss | Loss |
| Stand (same as dealer) | Tie | Tie |
| Hit or Stand (with drawn ace) | Win / loss | Incorrect Win / Loss |

Issues

1. No betting system
2. No score system
3. Aces can only be used as 1 instead of the desired 1 OR 11 and are shown as 1 of *suit*
4. Cluttered script that could use more function

**Prototype 2 Driver –** Uses foundation of previous prototype and begins to fix ‘Ace’ issues. Aces are now properly outputted as “Ace” instead of 1. Aces still cannot be chosen as 1 or 11 when adding the sum of the dealer or the player.

Testing – Tested through the program multiple times to see whether 1s were shown as Aces. Everything seems to work. Previous issues from the last prototype persist.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input (Reference Card)** | **Expected Output** | **Actual Output** | **Expected Sum** | **Actual Sum** |
| 1 | Ace of Clubs | Ace of Clubs | 1 || 11 | 1 |
| 14 | Ace of Spades | Ace of Spades | 1 || 11 | 1 |
| 27 | Ace of Hearts | Ace of Hearts | 1 || 11 | 1 |
| 40 | Ace of Diamonds | Ace of Diamonds | 1 || 11 | 1 |

Issues

1. No betting system
2. No score system
3. Aces can only be used as 1 instead of the desired 1 OR 11
4. Cluttered script that could use more functions

**Prototype 3 Driver –** Creates an ‘aceConversion’ function that converts aces to 11s when applicable. The player gets to choose to convert aces to 11 and the dealer automatically converts aces to 11 if their sum is less than 21. Dealer aces past the initial hand are treated as ones.

Tests – Game is in a playable state with proper win status for every test case now. The player ace conversion is a bit wonky because there is no clear way to automatically decide whether the player should convert an ace to 11 so they are instead given the choice to convert all the aces they draw if they choose to do so. This mechanic is not saved as its own separate function as it cannot be called for both the dealer and the player as they both require different parameters. Aces properly count as 11 when summing hands and still show as Aces when initially drawn.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input (Reference Card)** | **Expected Output** | **Actual Output** | **Expected Sum** | **Actual Sum** |
| 1 | Ace of Clubs | Ace of Clubs | 1 || 11 | 1 || 11 |
| 14 | Ace of Spades | Ace of Spades | 1 || 11 | 1 || 11 |
| 27 | Ace of Hearts | Ace of Hearts | 1 || 11 | 1 || 11 |
| 40 | Ace of Diamonds | Ace of Diamonds | 1 || 11 | 1 || 11 |

|  |  |  |
| --- | --- | --- |
| **Input (Hit or Stand)** | **Expected Output** | **Actual Output** |
| Hit until bust | Bust | Bust |
| Hit (higher than dealer) | Win | Win |
| Hit (lower than dealer) | Loss | Loss |
| Hit (same as dealer) | Tie | Tie |
| Stand (higher than dealer) | Win | Win |
| Stand (lower than dealer) | Loss | Loss |
| Stand (same as dealer) | Tie | Tie |
| Hit or Stand (with drawn ace) | Win / loss | Win / Loss |

Issues

1. No betting system
2. No score system
3. Cluttered script that could use more functions

**Prototype 4 Driver** – Creates a betting and player balance system. By adding this mechanic, the game can now be replayed until the player is broke. Betting calculations are seamlessly added into the final calculation if statements.

Tests – All bets result in proper result if a proper bet is used. Non integer bets are correctly calculated but not formatted nicely. Non proper bets can be used and unfortunately this results in bad bet calculations in the final if statements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input (Bet)** | **Input (Player Balance)** | **Expected Output ( Player Balance – bet)** | **Actual Output ( Player Balance – bet)** | **Expected Output ( Final Balance) ( Win )** | **Actual Output (Final Balance) ( Win )** |
| 50 | 500 | 450 | 450 | 550 | 550 |
| 100 | 500 | 400 | 400 | 600 | 600 |
| 550 | 500 | Invalid bet | -50 | Error | 1050 |
| -50 | 500 | Invalid bet | 550 | Error | 450 |

Issues

1. Improper bets can be placed resulting in unwanted output
2. No score system
3. Cluttered script that could use more functions

**Prototype 5 Driver** – Creates a score system that updates the ‘high score’ as the highest balance the player achieves. Small update but still important in making the game ‘playable’

Tests – Tested a multitude of starting bets ranging from 0-1000 that were all successfully saved to the high score variable. As the high score cannot be predicted, other tests were impossible to predict. Otherwise, this scoring system works flawlessly.

|  |  |  |
| --- | --- | --- |
| **Input (initial Balance, hit until bust)** | **Expected Output ( high score )** | **Actual Output ( high score )** |
| 1 | 1 | 1 |
| 100 | 100 | 100 |
| 1000 | 1000 | 1000 |

Issues

1. Improper bets can be placed resulting in unwanted output
2. Scoring system cannot properly be tested without a way to review the balance over time
3. Cluttered script that could use more functions

**Prototype 6 Driver** – Streamlines the previous prototype into a more readable script by creating more functions (which have already been tested in previous prototypes) and collating them into easy-to-read sections. These functions include: hitPlayer and hitDealer (collated addCard and cardFinder together for player’s and dealer’s subsequent turns), playerAceConversion and dealerAceConversion (converts selected aces to 11 within the hand sum), initialOutput and finalOutput (outputs the initial and final dealer and player hands) and finally, winCalculator (decides who wins the game and how much the player receives back (or loses) in bets )

Tests – Game runs the same as it did before but is much easier to read and fix for future updates to functions and main code. No need to test any specific cases as this prototype only compares functionality of this prototype vs the previous prototype.

|  |  |  |
| --- | --- | --- |
| **Prototype 5 – Arbitrary output** | **Prototype 6 – Arbitrary output** | **Equal?** |
| A | A | Yes |
| B | B | Yes |
| C | C | Yes |

Issues

1. Improper bets can be placed resulting in unwanted output
2. Scoring system cannot properly be tested without a way to review the balance over time

**Prototype 7 Driver** – Collates even more functions together, mainly the initialisation function, to further streamline the code. An initalBet function was created to fix the improper bet issue and now correctly works as intended. Bets greater than the player balance or less than 0 are considered improper bets and now prompts the user to enter a valid bet.

Tests – All final bet calculations and win status outputs work as intended now that only proper bets can be entered by the user.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input (Bet)** | **Input (Player Balance)** | **Expected Output ( Player Balance – bet)** | **Actual Output ( Player Balance – bet)** | **Expected Output ( Final Balance) ( Win )** | **Actual Output (Final Balance) ( Win )** |
| 50 | 500 | 450 | 450 | 550 | 550 |
| 100 | 500 | 400 | 400 | 600 | 600 |
| 550 | 500 | Invalid bet | Invalid bet | Error | Error |
| -50 | 500 | Invalid bet | Invalid bet | Error | Error |

Issues

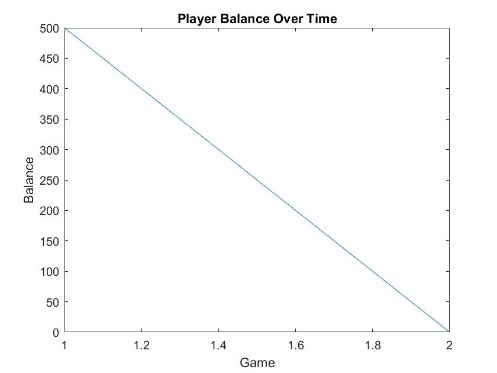
1. Scoring system cannot properly be tested without a way to review the balance over time

**Prototype 8 Driver** – Stores the players Balance in a vector after each ‘game’ and outputs the vector as a graph that can be viewed at the end of the game. This acts to test the scoring system and to show the viewer their progress as the game went on. An output of the rules were also added to the initialisation function to guide new players when they first play the game.

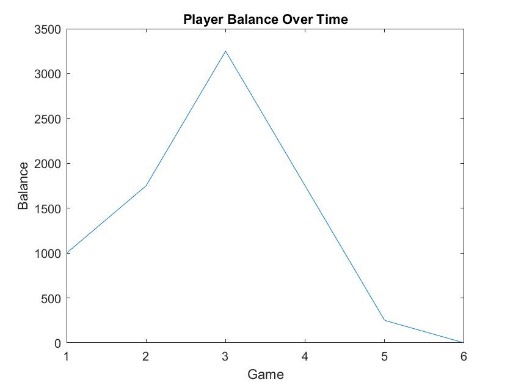
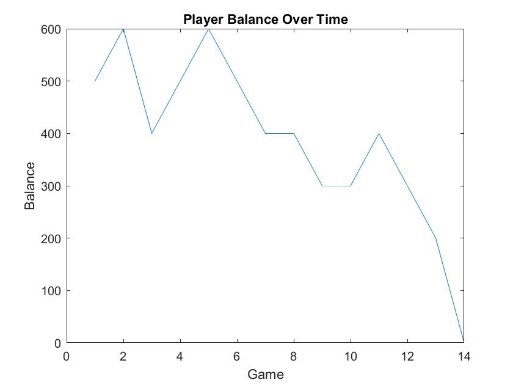
Tests – Scoring system works as expected now that it can be cross referenced with an easy-to-read graph that plots the player’s balance over time. Plotting the balance over time works as expected.

|  |  |
| --- | --- |
| **Observed Highest Balance** | **Actual Highest Balance** |
| 550 | 550 |
| 500 | 500 |
| 3250 | 3250 |
| 600 | 600 |



Issues

* Reusability of functions



**Protype 9 Driver** – A major update to make functions more reusable. Player / Dealer loops have now been scrapped and addCard is now the main function which can add a single card to a hand at a time. Output functions have been scrapped and replaced with a single function (currentStatus) that outputs the dealer and player’s hand based on different Boolean variables. Player ace conversion and dealer ace conversion have now been merged and the functionality has changed – all aces drawn as the first 2 cards are always converted into 11s and are now shown appropriately in each respective output. The player has no choice to convert aces anymore. This has less functionality than before but makes the function much less confusing and more akin to how blackjack is supposed to be played. The add card function also reshuffles the deck once the current card reaches at least the 48th card in the deck. A player choice function has been made to identify correct player choices now, works like the initialBet function. Game works almost the same as before with some minor tweaks to formatting.

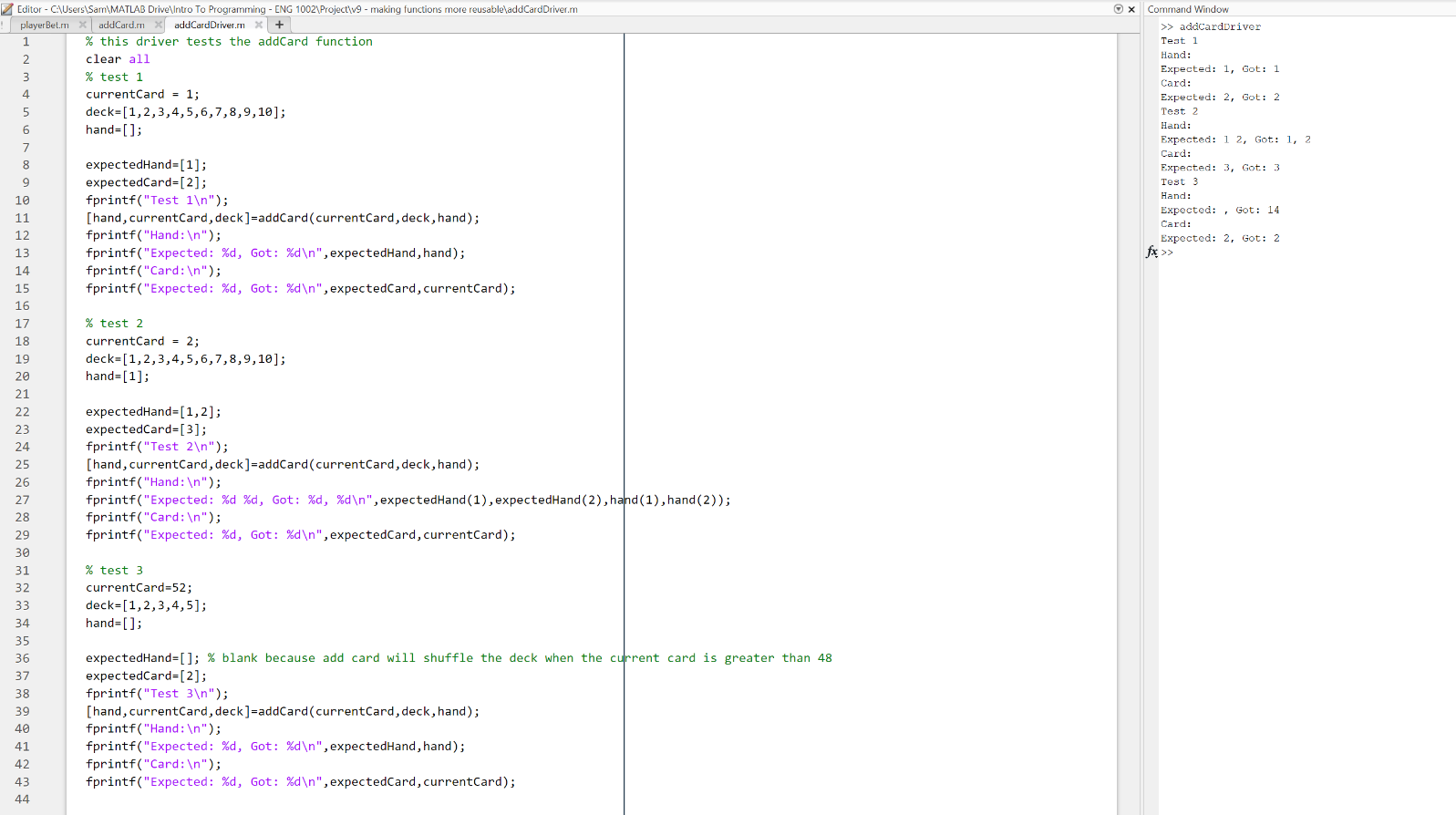
Tests – A driver function has been created for each applicable function in addition to the prototype driver to test for new functionality. I am happy with the state of the current program, these drivers will test each function and small tweaks will be made, if any, to fix any issues for the final version of the game.

*intialisation* – no need to test any variables. Passes the eye test, if all other functions work, this one must too.

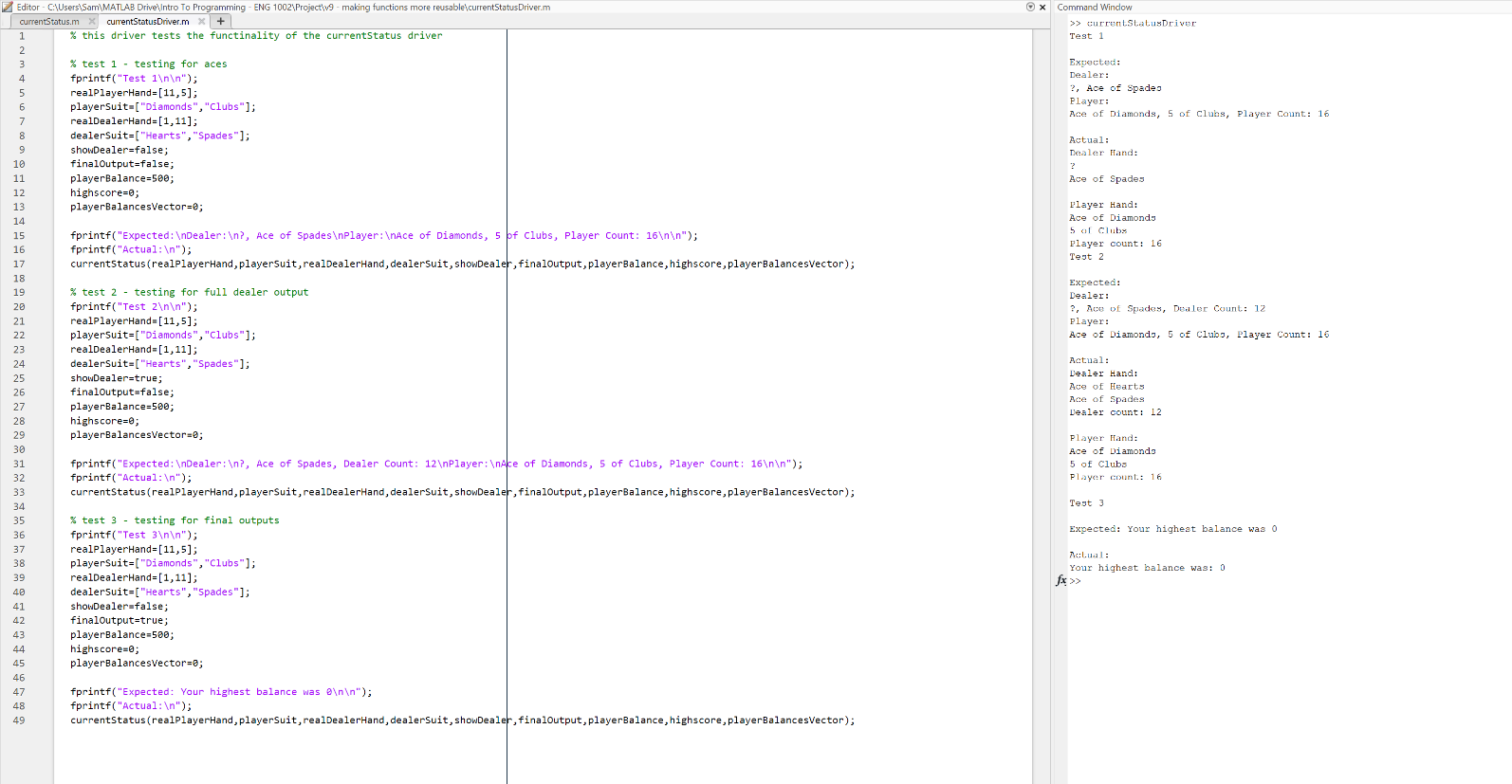
*playerBet* – tested in prototype 7 driver thoroughly.

*shuffleDeck* – tested in initial functions, has not changed since.

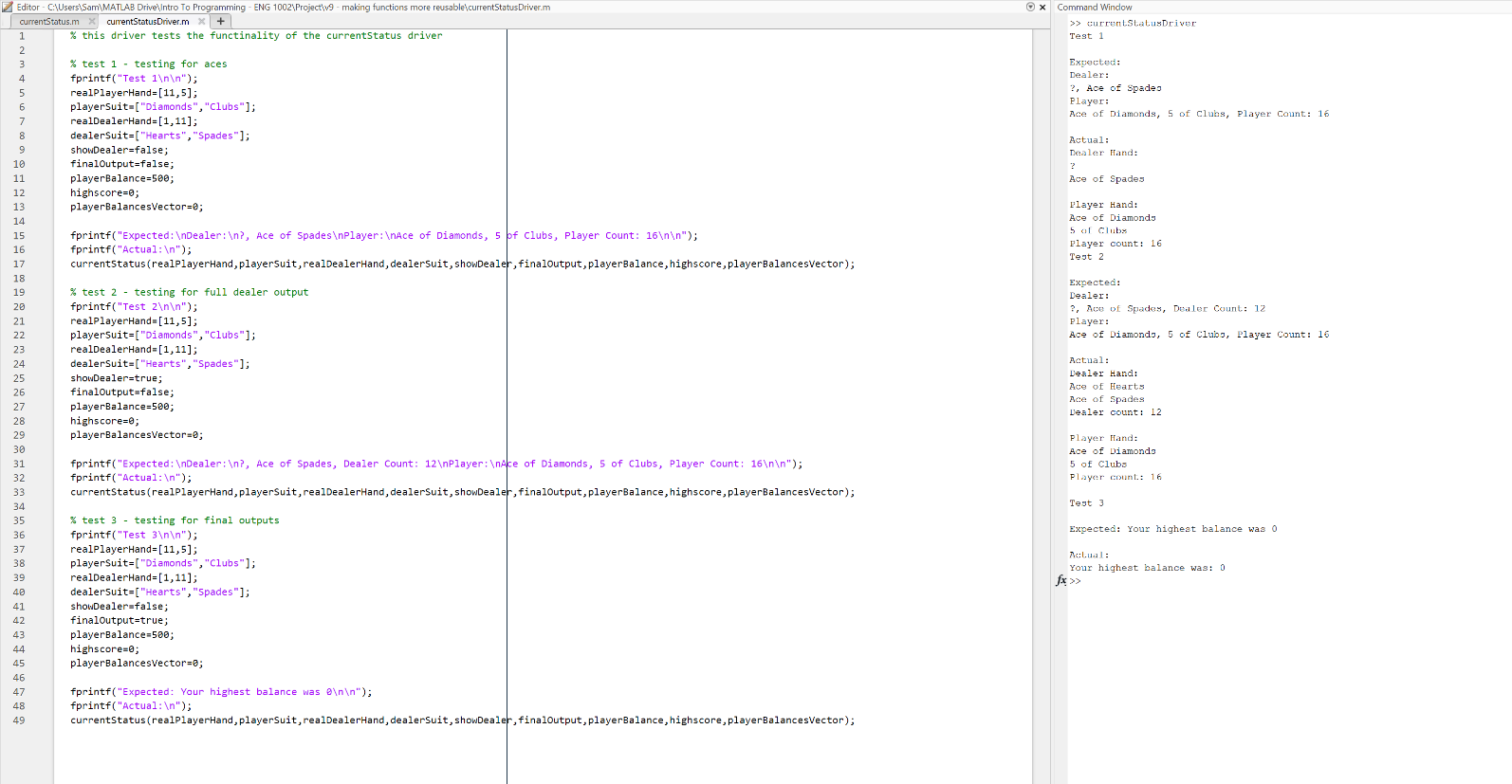
*addCard* - successfully adds a single card to an empty hand. Successfully adds a second card to a hand of 1 card. Successfully shuffles the deck, sets the current card as 1 when the imported parameter current card is greater than 48.



cardFinder – tested in initial functions, has not changed since.

*currentStaus* – tested using the different Boolean values as well as ace conversion. 

*aceConversion* – tested aces in the first and second spot of hand, tested in both spots of hand and tested in third spot.



*playerChoice* – tested thoroughly in prototype 9 driver.

*winCalculator* – tested thoroughly in prototype 6 driver.

*Blackjack v1.0*

**Preface**: Happy with the state of the game. There is still a lot of room to improve the code – ace conversion still isn’t perfect as the function needs to be called more times than I am happy with and can’t change drawn hit aces to 11 even if it doesn’t push the sum past 21. I cannot figure out a way to implement this. Perhaps shuffleDeck, cardFinder and aceConversion can turned into a single function, but this will require a lot of code rewriting. The previous prototype and function drivers have been used thoroughly to test all the different scenarios and I am confident that the game will handle any legitimate attempt by the user. Formatting is clean. Style has been completely overhauled in the last prototype and I am confident it satisfies all criteria. Satisfactory commenting is frequent throughout the main body as well as in the functions. All final drivers and tests are also well commented.

**Final Thoughts**: More things can be added such as the ability to split or double down but this was far too complex given the scope of the project. A GUI also seems like the next logical step in furthering the development of this game but again, not within the scope of this project. Better function names could also be implemented through the entire development in the future to avoid any discrepancies within the organisation of the project. Perhaps using Git could be beneficial in dealing with updating larger pieces of code.

**Functions:**

*Initialisation:* initialises many variables to be used by other functions. Also prompts the user if they want to read the rules or not.

Testing – Passes the eye test – no need for any drivers as this function needs to work for other functions to work and therefore a single driver can be used to test this and other functions at the same time.

*shuffleDeck:* creates a vector of 52 non repeated integers from 1:52. Imitates a standard, shuffled, 52 card deck.

Testing – Testing completed in the initial function stage of development. Code has not changed since.

*playerBet:* asks player for their bet and determines whether it is valid or not.

Testing – thoroughly tested in prototype 7 driver.

*addCard:* adds a single card to ANY hand of ANY length.

Testing – This function went through 3 different forms, finally evolving into what is currently described. The initial function was tested thoroughly in the initial functions stage of development. The collated version was thoroughly tested in protype 6 and 7 drivers. The final version was thoroughly tested in prototype 9 driver.

*cardFinder:* converts reference cards (1:52) to actual cards with a face and suit value.

Testing – Testing completed in the initial function stage of development. Code has not changed since.

*aceConversion:*  converts aces to 11s for the sake of hand summing, determined using different parameters based on the position of the ace within a hand.

Testing – Went through multiple different versions. Was initially separated into 2 different functions, one for the player and the dealer. Testing for these versions can be found in prototype drivers 2 and 3. The final version combines the previous 2 functions and makes the function much easier to understand. Testing for this version can be found in the prototype 9 driver section of this document.

*currentStatus:* outputs the status of the dealer and the players hand OR outputs the final message to the player based on different Boolean variables.

Testing – This function was not initially a function, so testing was done within its respective prototype drivers. It was then separated into 3 different functions and then into a single function to simplify the code. These tests can be found in prototypes 6 and 9 respectively.

*playerChoice:*  asks player for choice – hit or stand, and determines whether their answer is valid or not

Testing – This function uses the same format as the playerBet function and therefore does not need to be tested.

*winCalculator:* checks for status of player and dealers and outputs winnings accordingly

Testing – testing completed in protype driver 7.

*playeyGameChoice:* does not need to be tested, uses the same structure as playerChoice function with different I/O

*getHint: tested thoroughly within the prototype driver*