Program Design

## Input

1. In detail indicate what the inputs to this program are?

* **The number of players playing**
* **The amount of money the user want the players to start with (The Buy In Amount)**
* **Ask the user what they want to do for each turn. (1. Check/Call 2. Bet/Raise 3. Fold)**

1. How do you validate the input.

* **The number of players playing should be between 2-4 players**
* **The amount of buy in should be between 100-1000**
* **The user can only choose a choice of 1-3 for each turn**

1. Pseudocode: Briefly explain how this section will look like

* **Ask user to input number of players playing and the amount of buy in**
* **And for each player’s turn show them a menu of 3 choices and ask user to input**
* **Validate the input.**

## Processing

1. What is this program calculating?

* **Poker Game**

1. Can we divide the calculations to the smaller “modules”

* **Shuffle Deck**
* **Deal Cards**
* **Evaluate hand**
* **Check For Pair**
* **Check For Two Pair**
* **Check For Three Of A Kind**
* **Check For A Straight**
* **Check For A Flush**
* **Check For A Full House**
* **Check For A Four Of A kind**
* **Check For A Straight Flush**
* **Check For Royal Flush**
* **Check For High Straight**
* **Find Winner**
* **Compare hands**
* **Find the better hand**
* **Sort**
* **Best High Card**
* **Best Pair**
* **Best Two Pair**
* **Best Three Of A Kind**
* **Best Straight**
* **Best Flush**
* **Best Full House**
* **Best Four Of A kind**
* **Best Straight Flush**
* **Find pair**
* **Find threes**
* **Find Fours**
* **Find Double Pair**
* **Find Straight**
* **Find Straight Flush**
* **Check if Bet Can Continue**

1. Pseudocode: Briefly explain how this section will look like – Each module should have its own section.

**CHECK IF BET CAN CONTINUE**

* Make a loop 4 times
* Check if the Player Playing[loop counter] is true
* If it is check again if Money[Loop counter] is greater than 0
* If it is increment the number of players in the round
* Outside the loop check if the number of players in the round is greater than 1
* If it is return true
* If not return false

**FIND WINNER**

* Find the value of each players hand that is still playing that round
* The players that is not playing cards value will be given -2
* Then compare player 1 and player 2 card values
* Then compare the one with a higher card with player 3
* Then compare again the higher card with player 4
* If players with same card value happens, the number of players tied will be incremented to one
* And assign the first and second player tied to the players number that is tied
* But if there is player with a higher card, the number of players tied will be change to 0 again

**COMPARE HANDS**

* Check player 1 card value and player 2 card value that is being compared
* If player 1 card is greater winner = Player Number 1
* If player 2 card is greater winner = Player Number 2
* If both is equal call the function find the better hand to find the winner

**SORT**

* Make a loop for 7 times
* Make a inner loop for 7 times
* Then check if array[inner loop counter][0] is greater than array[inner loop counter+1][0]
* If it is, swap their values

**FIND THE BETTER HAND**

* First check if the players card value
* If -1, better card = -2
* If 0, better card = call function best high card
* If 1, better card = call function best pair
* If 2, better card = call function best double pair
* If 3, better card = call function best three of a kind
* If 4, better card = call function best straight
* If 5, better card = call function best flush
* If 6, better card = call function best full house
* If 7, better card = call function best four of a kind
* If 8, better card = call function best straight flush
* If 9, better card = -1

**SHUFFLE DECK**

* First, make a deck of 52 cards into a 2D array.
* The array will be having 4 rows and 13 columns.
* Each column will represent the card number and the row will represent the suit
* Where 11 is for Jack, 12 is for queen and 13 is for king
* The deck will be shuffled before each round
* First, make a loop from 1-52
* Make a loop until the row and column randomized in the deck array is not equal to 0
* Then randomize a number from 1-4 for the suit and 1-13 for the card value
* If the number is not equal to zero store the loop counter in it.
* The number stored in the array is to represent the cards position on the deck
* Where 1 is the card on top of the deck and 52 is the card on the bottom of the deck.

**DEAL CARDS**

* First deal 2 cards to every player that is going to play.
* Each player’s hand will be a 2D array too
* The array will be 2 row and 2 columns
* Make a loop twice as the card is going to be dealt by giving 1 card to each player first
* And after each player gets a card, give another card to the player in the same order
* The card given to each player will be determined by the deck array.
* Player 1 will receive the first card and it will be given by finding card number 1 that is stored in the deck
* Then check the row and column the number 1 is stored in the deck array.
* The row number will be the player’s card suit
* While the column number will be the player’s card value
* It will then deal 5 cards to the board
* First, the card after the last card given to the players will be set aside.
* This card is usually called burn
* It will then deal 3 cards by storing it in a 2D array called board which is called the flop.
* It will be having 2 rows and 5 columns
* Where the first row is to store card value
* And second row to store card suit
* Then burn another card and deal another card that is called the turn,
* Then deal another card that is called the river.

**Evaluate Hand**

* First check if the players card contains royal flush, if it does return 9
* If not, check if it contains straight flush, if it does return 8
* If not, check if it contains four of a kind, if it does return 7
* If not, check if it contains full house, if it does return 6
* If not, check if it contains flush, if it does return 5
* If not, check if it contains straight, if it does return 4
* If not, check if it contains three of a kind, if it does return 3
* If not, check if it contains two pairs, if it does return 2
* If not, check if it contains pairs, if it does return 1
* if high card, return 0

**Check For Pair**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored, check if there is a 2 in the card counter
* If there is, the cards have a pair

**Check For Two Pair**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored, check if there is a 2 in the card counter
* If there is, increment the number of pairs.
* Then Check if the number of pairs is greater than or equal to two
* If it is, return true
* If not return false

**Check For Three Of A Kind**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored, check if there is a 3 in the card counter
* If there is, return true
* If not return false

**Check For A Straight**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* First check if the cards in array card counter position 0,9,10,11 and 12 is not equal to zero
* If it is, return true
* Then make a loop from 0-8
* Check if the card counter(loop counter) is not equal to zero
* If it is, make a loop four times to check if the next four cards is not equal to zero
* If it is, return true

**Check For A Flush**

* Add the board cards and the player cards to a suit counter
* The card counter is a 1D array with a size of 4.
* Increment the number in the suit counter for every suit of the player and board
* After all suit has been stored, check if there is a 5 in the suit counter
* If there is, return true
* If not return false

**Check For A Full House**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored
* Check if it has a pair and a three of a kind
* If yes, return true

**Check For A Four Of A Kind**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored, check if there is a 4 in the card counter
* If there is, return true
* If not return false

**Check For Straight Flush**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored
* Check if it has a straight and a flush
* If yes, return true

**Check For Royal Flush**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored
* Check if the card has 10,J,Q,K,A
* Check if it has a straight and a flush
* If yes, return true

**Check For High Straight**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored,
* check if the cards in array card counter position 0,9,10,11 and 12 is not equal to zero
* If there not, return true
* If not return false

**Best high card**

* Make a loop from 4 times from 3-0
* Check if the both of them has an ace
* The one with ace will be the winner
* If they both has no ace
* Check player1[loop counter] > player2[loop counter]
* If it is the winner will be player 1
* If player1[loop counter] < player2[loop counter]
* It will be player 2
* If none of the conditions match, they tied.

**Find Pair**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored,
* check for the number 2 in the array and return the position it is stored at.

**Find three**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored,
* check for the number 3 in the array and return the position it is stored at.

**Find Four**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored,
* check for the number 4 in the array and return the position it is stored at.

**Find two pair**

* Add the board cards and the player cards to a card counter
* The card counter is a 1D array with a size of 13.
* Increment the number in the card counter for every card of the player and board
* After all card has been stored,
* check for the number 2 in the array and assign the position of the number to Pair 1
* Increment the number of pairs too
* Then for another number 2 and then check if the number of pairs is 1
* If it is assign the position of that number to Pair 2

**Best pair**

* Call the function find pair for both players
* Check whose pair is greater the one with the greater pair wins
* If the pair is the same,
* Call the function best high card

**Best double pair**

* Call the function find two pair for player 1
* Call the function find two pair for player 2
* First check for each player strongest pair
* Then compare their strongest pair
* The one with the stronger pair wins
* But if it is the same check their weaker pair
* The one with the greater weaker pair wins
* If the weaker pair is also the same
* Check for their high card
* Then compare their high card
* The one with greater high card win or if it’s the same it is a tie

**Best Three Of A Kind**

* Call the function find three for both players
* And then check whose three of a kind is greater
* The one with greater wins
* If it is the same, check the remaining two cards and compare who has the greater cards
* If the cards are the same they are tied

**Best Four Of A Kind**

* Call the function find four for both players
* And then check whose four of a kind is greater
* The one with greater the greater one wins.

**Best Flush**

* It is the same like finding the best high card
* But you must check the suit of the high card
* If it matches the suit of the flush then it is the highest card in the flush
* After finding the highest card with the same flush, compare the two cards
* If the cards is equal, check the next highest card with the same suit
* If all 5 cards are equal, they are tied

**Best Straight or straight flush**

* First call the function check for high straight for both players
* If both of them has it they are tied, if only one of them has it the one with it wins
* Then make a loop from 0-8
* Check if the card counter(loop counter) is not equal to zero
* If it is, make a loop four times to check if the next four cards is not equal to zero
* If it is, the last card would be a straight to that card
* Then take the last card and compare it with the other players last card. The one with the greater card has a higher straight.
* If the card is equal, then it’s a tie.

## Output

1. What will this program display to the users?

* **Output the player’s card, the board cards, the player’s money and what the player before them has done.**
* **Output a menu that give the player choices on what to do**
* **Output the winner of the round and game**

1. What kind of formatting this output will use (Currency with $ …)?

* **Output the player’s money with currency**

1. Pseudocode: Briefly explain how this section will look like

* **For every player’s turn output their cards and the board cards according to the round turn**
* **If the round turn is 0 which is called pre-flop, no board cards is shown**
* **If the round turn is 1, the first three card is shown**
* **If it is 2, the first four card is shown**
* **After the third turn, all cards will be shown**
* **Make 2 string array called suit and value**
* **The suit array will hold {hearts, diamonds, clubs and spades}**
* **The value array will hold the 13 strings from aces to kings**
* **To output a player cards, make a loop 2 times**
* **Output value[ Player Hand[loop counter][0] ] “of” Player Hand[ card[loop counter][1] ]**
* **To output the board cards make a loop a number of round turn times**
* **Output value[ board[loop counter][0] ] “of” board[ card[loop counter][1] ]**