Ruby Casino's Blackjack

By: Nicolas Machado PID: 6398035

EEL 2880 C Programming for Embedded Systems

Introduction:

My project is a blackjack game made in programming language C. I love playing video games, but most of the games me and my friends play are made as huge projects from big companies like Epic Games or Ubisoft. These companies have whole teams that are assigned to create and maintain the games with constant updates. I wanted to make a game, but to make one of a scale like that I would need more people and a large number of resources, like engines and others. So, I decided to make a my favorite card game since they are simple enough to be made with just a programming language.

I have always loved blackjack and I got the inspiration from playing blackjack in the video game Grand Theft Auto V (GTA V). The casino in GTA was called “Diamond Casino” so the name that I picked was inspired of off that. I wanted to be able to create an experience like that in the form of a program in C.

Description:

1. The Programs starts with a main() function that prints some of the terminology of the game, also prints a message that the game is starting and ending, and has a playGame() function which displays the main game.
2. Inside playGame() function we have about all the other functions created in my program.
3. When the game starts function randomNum(), fillDecks(), oneMoreCardPlayer() twice, printPlayerCards(), oneMoreCardDealer(), and printDealerCards() are functions that are instantly being called.

- randomNum() gives a random number between 1 and 13 which give the probability of getting certain cards for the player and dealer.
- fillDecks() already predetermines the first 10 cards for both the player and the dealer. This function makes it easy for cards to be called on.
- oneMoreCardPlayer() when called on gives the player one more card. This happens again later on when the player Hits or inputs H/h.
- oneMoreCardDealer() is the same things as the player it just gives a new card to the dealer.
- printPlayerCards() and printDealerCards() just print the cards on the output console.

4. The game will being using a big while loop that ends when Boolean expression gameFinished = true.
5. The player's and the dealer's move will be shown as while loop as well starting with the player.
    - The player's while loop will end when playerFinished = true which will happen depending on the players input.
    - The player's while loop has functions getPlayerHand(), which uses functions like onMoreCardPlayer() and printPLayerCards() to get more cards when the player want to HIT or just prints current cards when the player wants to STAND, and checkIfOver21Player(), which checks if the player's cards have a value over 21.
    - In getPlayerHand() the player will decide whether he wants to HIT(h) or STAND(s) inputting 'h' or 's'. The player can keep on hitting until checkIfOver21Player() detects that the player cards went over 21 and determines

not only that playerFinished = true , which will end the player's turn, but that gameFinished = true, which will exit the whole game since the player loses when they go over 21.

6. After all that, there is an if statement that if gameFinished = true, the program will print the results on a result.txt file and will finish the game. This will only be called before checking the dealer when the player goes over 21 during his turn.
7. Now for the dealer; the dealer plays automatically after the player. The dealer will also always hit if his value is lower than the player's if the player didn't go over 21 during his turn. The dealer has a while loop that will end when dealerFinished = true.
    - Inside the while loop there are functions like getDealerHand(), which uses functions like oneMoreCardDealer() to get more cards when the player has higher value and function printDealerCards() to print all the cards the dealer got.
    - The dealer will play according to what the player got.
8. After the player and the dealer, their scores will be compared inside function compare().
    - If the player has less value of cards than the dealer, then the dealer will stop getting cards and he will win.
    - If the dealer got more cards than the player but when over 21 then the player automatically wins.
    - And if, like mentioned before, the player goes over 21 then the dealer won't even have to play and will be declared the winner.

This is without mentioning all the for loops, if else if statements, and switch case statements used

Instructions:

- The player will be given the information below about the terminology of the game.

```
Terminology:
*HIT(H) means that the player will get one more card.
*STAND(S) means that the player does not want to get more cards and the dealer will play.
*BUST means that the player of the dealer has gone over 21.
```

- The player will start with 2 cards will only know 1 of the dealer's cards. The player will also decide if to "Hit" or "Stand" depending on the total value of their cards. The player will also try to not go over 21.

```
Your hand:
 3 5
Dealer's hand:
 10 ?
Do you want to HIT or STAND?(H/S):
```

- In this example, let's say the player decides to hit. His hand will be displayed again but with the extra card they requested.

```
Your hand:
 3 5
Dealer's hand:
 10 ?
Do you want to HIT or STAND?(H/S):  h
Your hand:
 3 5 10
Do you want to HIT or STAND?(H/S):
```

- Now lets say the player decided to Stand since they do not want to go over 21.

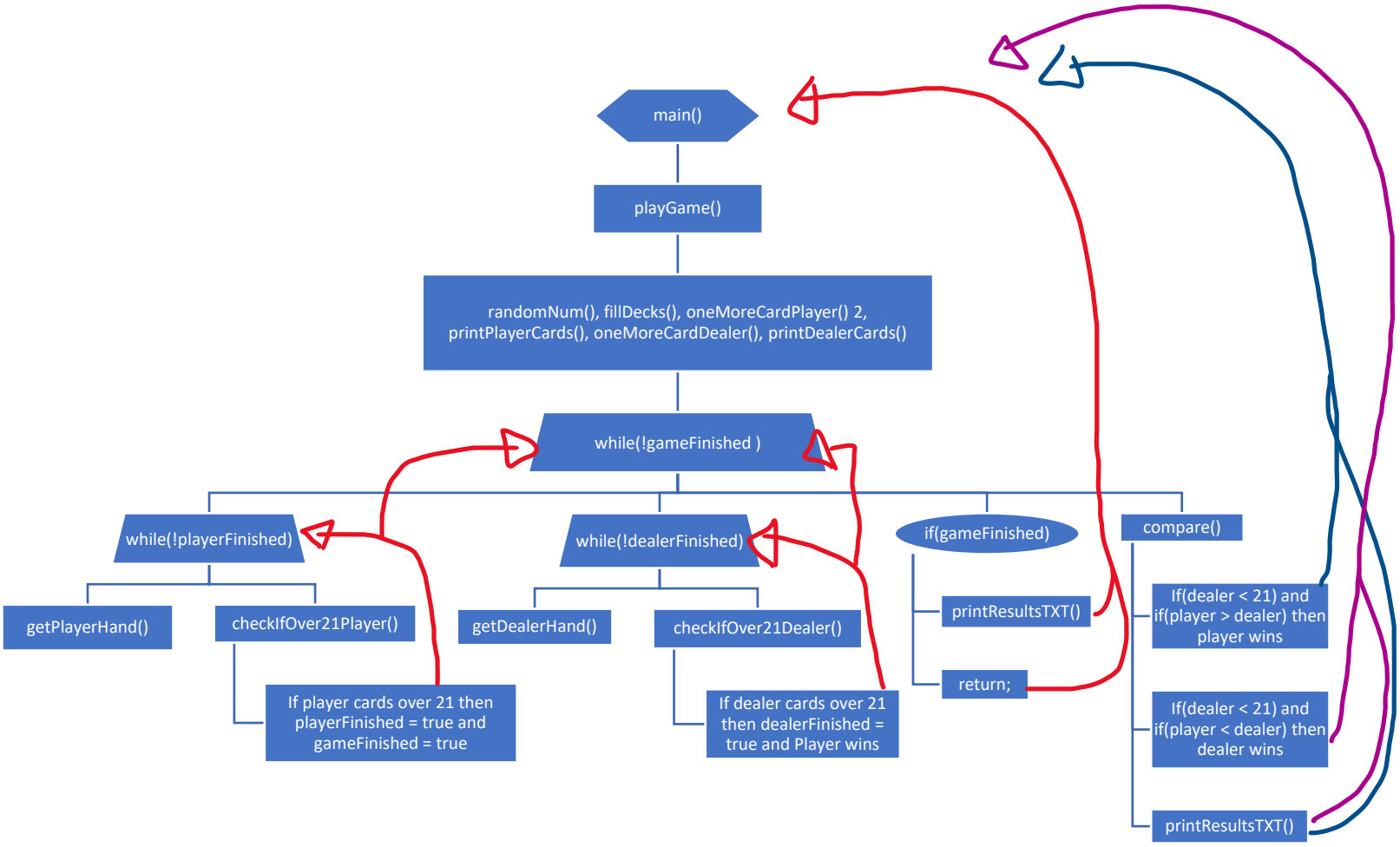
```
Your hand:
 3 5 10
Dealer's hand:
 10 4
Dealer's hand:
 10 4 10
BUST!!! The dealer went Over 21. You win.

Game finished!
```

- Since the player decided to stand his move was over and it is then the dealer's turn to play according to what the player got. The dealer started with 10 and 4 totaling to an amount of 14. 14 is less than 18 so the dealer will hit again. The dealer got another 10, giving him a new total of 24 which is over 21. Remember, going over 21 means they lost, and it works the same way with the player.
- Now that the match is over, the player's and the dealer's hand along side their total value of cards will be displayed on a result.txt file that the program will create for the player looking like this.

Ruby Casino Blackjack > results.txt

```
1  RESULTS:Your hand:
2   3 5 10
3  Dealer's hand:
4   10 4 10
5  BUST!!! The dealer went Over 21. You win.
6
7  Players's score: 18
8  Dealer's score: 24
9
```

Flowchart:



I did not know how to make a good flowchart. Also, adding every loop and if statement inside most functions would be too much for just Word Flowchart function

Conclusion:

I learned many things while doing this project. I learned how having a lot of time can be used to optimize code. I did my project in like 5 different days and I started it without using any arrays and now at the end I had arrays inside arrays inside loops and if statement; I did not think

I would be able to do that. I also learned how to organize my project effectively with the use of functions (void example(){}). With the use of functions, I do not have to read the whole code again to understand what is happening, I can just read the reasonable name I gave the function and there (Example: oneMoreCardPlayer() gives the player one more card when called). My brother was the one who helped me to organize my code with functions and also taught me about the Boolean expression library (true or false expressions). I would say that the most difficult part after organizing the code was making most of the loops and using the variables correctly because changing the order of how the functions were being called inside the playGame() would completely mess with how some variables would be updated. All in all, organization is key and I hope that in the future I remember to plan before starting to code; even just setting up functions that resemble the overall layout of how the program works without putting anything inside those functions would help tons.