Trevor Oakes, Nate Miller, Ford Hirsch

Prof. Krish

ACS310

9 May 2019

Blackjack Implementation Via Python

After the completion of project 1, my partners and I realized that we had created a functional game of blackjack, but this implementation lacked many facets that make the game of blackjack more enjoyable. After consulting amongst ourselves and with traditional game rules, we decided that we must implement the final variables and outcomes that make blackjack that more enjoyable. These new additions that make the game complete are “splitting” hands, insurance abilities, and finally the addition of the infamous “five card charlie.” The insurance function simply asks the user if he would like insurance for the current hand he is playing. This capability allows players to cut down on their losses if the dealer hits 21. Otherwise, insurance forces them to play with their current bet. As previously mentioned, we also implemented the capability of “splitting” hands. Splitting hands provides blackjack players with the ability to split and play two hands if their first two cards are the same number or suit. If a player elects to split, they then must double down on their bet to cover their new (second) hand. Splitting simply gives the player an option to play two hands for the current round, and the possibility of accruing more funds from a single round of play. Lastly, we implemented the infamous, and quite rare, five-card-charlie. The five-card-charlie in blackjack simply accounts for if a player gets five cards under 21, then they automatically hit 21 and win the round. Some of these additions were quite simple to implement, and others provided us with a bit of a headache when proceeding with implementation. For the insurance capability, we simply added an input to ask the user if they would like insurance. This would then check the dealers deck to see if the dealer had 21. If he did, then the dealer takes half of the users bet. Otherwise, the user is forced to play with the bet they currently hold. Split was perhaps the most complicated to implement. For splitting we of course had to prompt the user with the option if the users first two cards were of the same number or value. For split we had to add an entirely different “add card” function so the user was able to play off of two separate hands. We also had to add an entirely new “hold bet” function for this as well as it was difficult to apply the original methods to two hands, at the same time. We also became a bit redundant with the base code in our main function so that the code would run all over again if the user decided to split his hand. The five-card-charlie ended up being by far the easiest in implantation as we just added a simple counter to our “addCard” function. This allowed the program to check the users card count, and provided the program with conditionals for if the user reached a total of five cards and it was under 21. The new features definitely increased the aesthetic appeal of the game, as it provided a more accurate representation of a true game of blackjack. The new implementations did not provide us with more efficiency though, as more detailed code and more thorough functions provided us with increased complexity. Run-time was hardly effected though, and the game still runs incredibly smooth.