**Project 1**

Title:

**Blackjack**

Course:

**CSC-5-46023**

Due Date:

**31 July 2014**

Instructor:

**Dr. Mark E. Lehr**

Author:

**Daniel Boebinger**

**Table of Contents**

Section Page

Rules -------------------------------------------------------------------------------------------------- 3

Stats -------------------------------------------------------------------------------------------------- 5

Summary -------------------------------------------------------------------------------------------- 6

Sample Inputs and Outputs ------------------------------------------------------------------------ 7

Flowchart -------------------------------------------------------------------------------------------- 8

Psuedocode ------------------------------------------------------------------------------------------ 9

Concept Checklist ---------------------------------------------------------------------------------- 14

Future Improvements ------------------------------------------------------------------------------ 16

References ------------------------------------------------------------------------------------------- 17

**Rules**

The objective of the card game “Blackjack” or “21” is to get the highest total value without going over twenty one. In this version of 21, there is one human player and one computer player that acts as the dealer. Therefore, the objective in this case is to beat the dealer. If the human player has a value over twenty one, it is called a “bust” and it is an automatic loss. This is true even if the dealer also “busts.” In the case that the player and dealer have the same value of cards, it counts as a draw and is called a “push.” In this version, the player’s original bet is returned.

To determine the value of a player’s hand, each card is given a specific value and the summation of all cards is the player’s hand. The numbered cards are worth their face value. The Jack, King, and Queen are worth ten points each. The Ace has a variety of values in this version of 21. The ace counted as either eleven or one. The Joker card is not used in this game.

The game 21 is started with the dealer dealing the player two cards, and then dealing him/herself two cards also. The player is able to see what cards he/she has but the dealer only reveals one of his/her cards. In this version of 21, the player then has a choice to “hit,” “stay,” “surrender,” “double down,” “split pairs,” or buy “insurance.” If the player chooses to surrender, the player forfeits his/her cards and is given half of the original bet back. If the player chooses to double down, the player has an additional bet up to the value of the original bet and receives one and only one additional card. If the player chooses to split pairs, the pair of cards is split into two separate hands and the player has to have another bet of equal size for the new hand. If the Dealer shows an Ace, the player can buy insurance up to half the original amount of the bet. If the dealer has Blackjack, the player gets double the insurance bet but losses the original bet. If the player chooses to hit, the dealer deals the player an additional card and the value of this card is added to the player’s total value of his/her hand. The player can continue to hit as long as his/her total value does not go over 21. When the player does not wish to hit, the player can stay and keep his/her current hand. After the player’s turn has ended, as long as the player did not bust, the dealer reveals his/her cards. The dealer is required to hit until the dealer’s total value of cards remains under seventeen. Sometimes the dealer hits at seventeen and this is called a “soft 17.” In this version of 21, this is not the case.

Some additional rules are that if player has an Ace and a ten value card (Blackjack), the player’s payout is 1:1. If the player or the dealer has Blackjack, they win, unless they both have Blackjack and in this case it is a push. The player can only split his/her pair once and there is no double down after a split.

**Stats**

Using Basic Strategy:

Overall Player Win Percentage = 48%

Overall Dealer Win Percentage = 52%

Player Blackjack Probability = 1 out of 21 hands

Player Bust Percentage = 16%

Dealer Bust Percentage = 28%

**Summary**

This project is based on the card game “21” or “Blackjack.” This version of the game is for one human player and one computer player that is also the dealer. There is a betting system and the player starts with $50 and the minimum bet is $5. There is only one player not counting the dealer, ties are counted as a push and the player gets his/her money back, and Aces are valued as high or low.

In the game 21, the statistics are in the dealer’s favor because not only does the human player go first, but statistically, the player only has a forty-eight percent chance of winning. The player also only has a one in twenty-one chance to get “blackjack,” and the player has a sixteen percent chance to bust and automatically lose the game.

This version of the game was designed to be user friendly by setting default decisions and validating the user’s input. This game also outputs what steps are happening and it always tells the user the most recent card or cards that are dealt and calculates the total value of the user’s hand. Once the user is done playing, he/she can open the results file and see the total wins, losses, and win percentage. The player’s score, or money, is also shown on the leaderboard if he/she has one of the top ten scores.

Overall, this project uses a large majority of the concepts learned so far and satisfies all of the requirements. Some future improvements for this project include but are not limited to having more than one player, split pairs as many times as possible, and provide even more additional blackjack rules and options.

**Sample Inputs and Outputs**

Player’s Initials:

Input – “AB” or “ABCD”

Output – “You can only enter three characters”

Player’s Cards:

Output – “Card 1 = Ace”, “Card 2 = 3”, “The total value of your cards is 14”

Dealer’s Cards:

Output – “Card 1 = Jack”, “Card 2 = ?”

Hit Option:

Input – “H” for hit

Output – “You Draw an additional card”, “Card = 2”, “The total value of your cards is

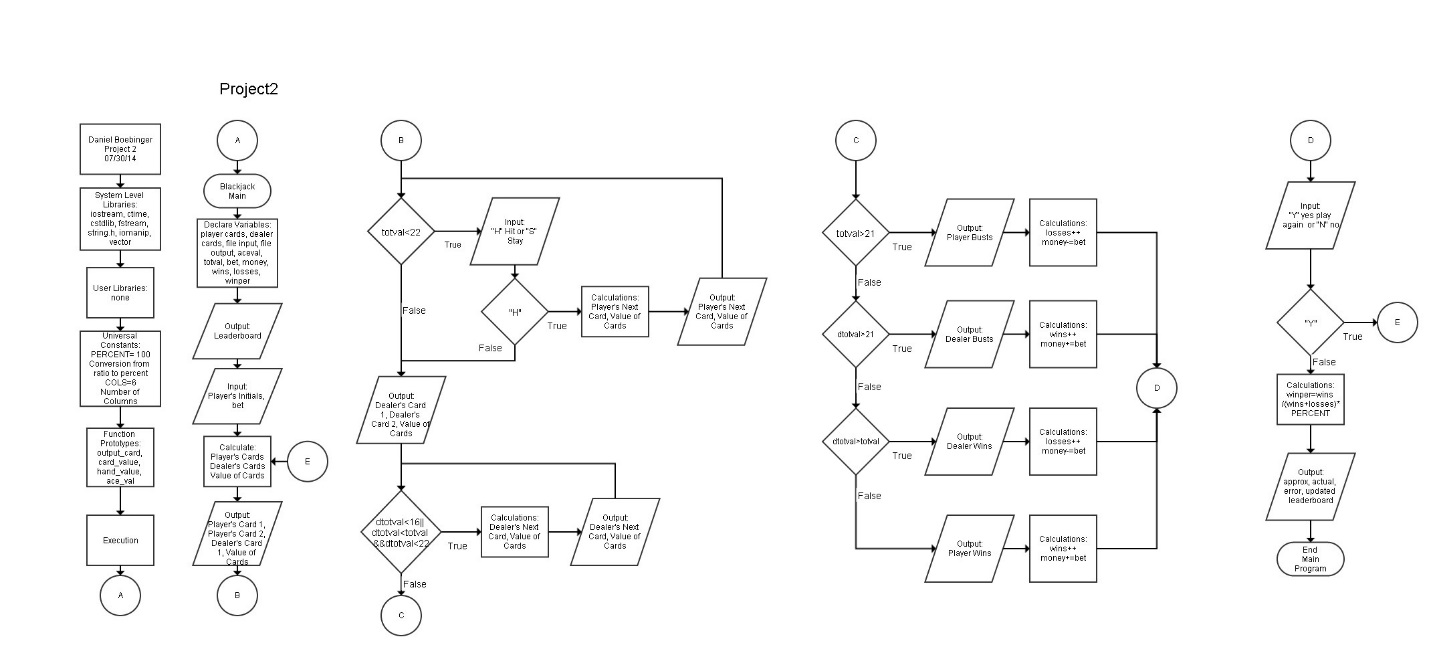
16”

Determining Winner:

Output – “You Bust”, “Dealer Busts”, “Dealer Wins”, or “You Win”

Results.txt:

Output – “Wins = 7 Losses = 9 Win Percentage = 43.75%”



**Flowchart**

***Pseudocode***

*Execution Starts Here!*

*Declare and Initialize Variables*

*Inputs and Constants*

*Input from file*

*Number*

*Leader Board with initials and scores from file*

*Size set to 4 for the 3 initials and null terminator*

*Initials of User*

*Outputs*

*User's Money (Buy-in at $50)*

*Player's Cards*

*Player's Second hand if splitting cards*

*Dealer's Cards*

*Size of vector for player's cards*

*Size of vector for player's second hand*

*Size of vector for dealer's cards*

*User's Bet in dollars*

*An additional Bet if needed*

*Insurance Bet if needed*

*Player's Total value of all cards*

*Player's Total value of all cards for second hand*

*Dealer's total value of all cards*

*Player's Decision*

*Number of wins*

*Number of losses*

*Winning percentage*

*Output to file*

*Set the Random Seed*

*Input Values*

*Open the Leader Board file*

*Test File For Open Failures*

*Input Leader Board*

*Output Game Title*

*Output Current Leader Board*

*Input Player's Initials*

*Output Pre-Game Instructions*

*Set loop=true*

*Do: Game Loop*

*Output Start of New Game Information*

*Input Player's Bet*

*Initialize Hands and Values*

*Determine Values for Card 1 and 2 for Player*

*Calculate Size of Vector for Player's Cards*

*Output Player's Cards*

*Determine Value of Player's Cards*

*Determine Values for Card 1 and 2 for Dealer*

*Calculate Size of Vector for Dealer's Cards Minus One*

*Output Dealer's Card 1*

*Output Dealer's Card 2 as Unknown*

*Calculate Actual Size of Vector for Dealer's Cards*

*Calculate total value of Player's cards*

*Determine Best Ace Value for Player's Cards*

*Ask Player for First Decision*

*Evaluate Player's Decision and Change Bet Value If Needed*

*Player Hit Loop For First Hand*

*While Total Value of Player's Card is Less than 22 and Player Chooses Hit*

*Determine Value of Card for Player*

*Calculate Size of Vector for Player's Cards*

*Output Player's Cards*

*Determine Value of Player's Cards*

*Calculate total value of Player's cards*

*Determine Best Ace Value for Player's Cards*

*Output Value of Player's Cards*

*If Total Value of Player's Cards is less than 22*

*Player Hit Loop For Second Hand if Needed*

*While Total Value of Player's Card is Less than 22 and Player Chooses Hit*

*Determine Value of Card for Player*

*Calculate Size of Vector for Player's Cards*

*Output Player's Cards*

*Determine Value of Player's Cards*

*Calculate total value of Player's cards*

*Determine Best Ace Value for Player's Cards*

*Output Value of Player's Cards*

*If Total Value of Player's Cards is less than 22*

*Else*

*Exit Hit Loop*

*Calculate Total Value of Dealer's Cards*

*If Both Cards are Valued as 10*

*Total Value of Dealer's Cards is 20*

*Else if Both Cards are valued as 11*

*Total Value of Dealer's Cards is 12*

*Else if One Card is Ace and The Other is Valued as 10 and Ace Value is High or Either*

*Total Value of Dealer's Cards is 21*

*Else if Card 1 is Valued at 10*

*Total Value of Dealer's Cards is 10 plus Card Number*

*Else if Card 2 is Valued at 10*

*Total Value of Dealer's Cards is 10 plus Card Number*

*Else*

*Total Value of Dealer's Cards is Card 1 Number plus Card 2 Number*

*Dealer's Turn in relation to Player's First Hand*

*If Player Cards has Blackjack*

*output Blackjack*

*If Dealer Cards has Blackjack*

*output Blackjack*

*If dealer has same value as player*

*Output Push*

*If Dealer’s cards are greater than player’s cards*

*}else if (dtotval>totval&&dtotval>16){*

*Output Dealer as Winner*

*if Total Value of Dealer's Cards is Less than Total Value of Player's Cards and Both are Less than 22*

*Dealer Hit Loop*

*While Total Value of Dealer's Cards is less than Total Value of Player's Cards or 16, and Less than 22*

*while ((dtotval<totval||dtotval<=16)&&dtotval<22){*

*Determine Value of Dealer's Card*

*Calculate Size of Vector for Dealer's Cards*

*Output Dealer's Cards*

*Determine Value of Dealer's Cards*

*Calculate total value of Dealer's cards*

*Determine Best Ace Value for Dealer's Cards*

*Output Total Value of Dealer's Cards*

*Determine winner in accordance to first hand*

*If the hands Tie*

*Output Push"<<endl;*

*If Total Value of Dealer's Cards is Over 21*

*Output That Dealer Busts and Player Wins*

*if Total Value of Dealer's Cards is Over Total Value of Player's Cards*

*Output That Dealer Wins*

*Else*

*Output That Player Wins*

*Else*

*Output That Player Busts*

*Repeat for Player's Second Hand if there is a second hand*

*If player chose to Surrender*

*Output Surrender*

*End of Game Loop*

*If player does not have enough money*

*Output You do not have enough money to play again!*

*Else*

*Ask to Play Again*

*If Player Chooses Yes Continue Game Loop*

*Default End Game Loop*

*While: Player Wants to Play and has enough money*

*Calculate Win Percentage*

*Update Leader Board*

*Enter User's Name and Score to Leader Board*

*Enter Name to Leader Board*

*Enter Score to Leader Board*

*Search and Sort Leader Board*

*Output Sorted Leader Board to file*

*Output the results to file*

*Output Total Wins*

*Output Total Losses*

*Output Win Percentage*

*Inform Player Results were Outputted to File*

*Exit Stage Right!*

*Close Files*

**Concept Checklist**

|  |  |  |
| --- | --- | --- |
| Used in Code | Concept | Example in Code |
| X | cout | Line 67 |
| X | cin | Line 90 |
| X | endl | Line 67 |
| X | #include | Line 8 |
| X | short | Line 54 |
| X | int | Line 44 |
| X | float | Line 56 |
| X | char | Line 36 |
| X | character strings | Line 38 |
| X | bool | Line 106 |
| X | Math Expressions | Line 698 |
| X | Type Casting | Line 698 |
| X | Naming Constants | Line 20 |
| X | Combined Assignment | Line 91 |
|  | Format Input | NONE |
| X | Format Output | Line 782 |
| X | File Input | Line 64 |
| X | File Output | Line 771 |
| X | Relational Operators | Line 70 |
| X | if | Line 170 |
| X | if/else | Line 66 |
| X | if/else if | Line 116 |
| X | switch | Line 682 |
| X | Menus | Line 177 |
| X | Logical Operators | Line 122 |
| X | Validating User Input | Line 88 |
| X | String Function | Line 91 |
| X | Increment and Decrement Operators | Line 70 |
| X | while | Line 338 |
| X | do-while | Line 88 |
| X | for | Line 70 |
| X | 1-D Array or Vector | Line 41 |
| X | 2-D Array | Line 36 |
| X | Passing 1-D Array or vector between functions | Line 25 |
| X | Passing 2-D Array between functions | Line 28 |
| X | Pass by value for a function | Line 24 |
| X | Pass by reference for a function | Line 27 |
| X | Defaulted parameters for a function | Line 26 |
| X | Return primitive data types for a function | Line 26 |
| X | Sorting | Line 731 |
| X | Searching | Line 731 |

**Future Improvements**

* Multiple players
* Split as many pairs as possible
* Additional blackjack rules and options
* Shorten overall code length

**References**

Dr. Mark Lehr

Savitch, Walter. *Problem Solving With C++*. 8th Edition. 2012 Pearson Education, Inc.

<http://www.goldentouchcraps.com/tamburin_005.shtml>