**Project 1**

Title:

**21**

Course:

**CSC-5-46023**

Due Date:

**21 July 2014**

Instructor:

**Dr. Mark E. Lehr**

Author:

**Daniel Boebinger**

**Table of Contents**

Section Page

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

Stats -------------------------------------------------------------------------------------------------- 4

Summary -------------------------------------------------------------------------------------------- 5

Sample Inputs and Outputs ------------------------------------------------------------------------ 6

Flowchart -------------------------------------------------------------------------------------------- 7

Psuedocode ------------------------------------------------------------------------------------------ 8

Future Improvements ------------------------------------------------------------------------------ 19

References ------------------------------------------------------------------------------------------- 20

**Rules**

The objective of the card game “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, there is no betting system so it counts as a loss to the player.

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 player can choose to have the Ace as a value of eleven, one, or to have 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” or “stay.” 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.

**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. The only differences between this game and the true game 21 are that there is no betting system, there is only one player not counting the dealer, ties are counted as a loss, and Aces could be valued as only 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.

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 incorporating a betting system, functions, arrays, and provide additional blackjack rules and options.

**Sample Inputs and Outputs**

Ace.txt:

Input – “High”, “Low”, or “Either”

Output – “Aces are High”, “Aces are Low”, or “Aces go Either Way”

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 or Stay Option:

Input – “H” for hit

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

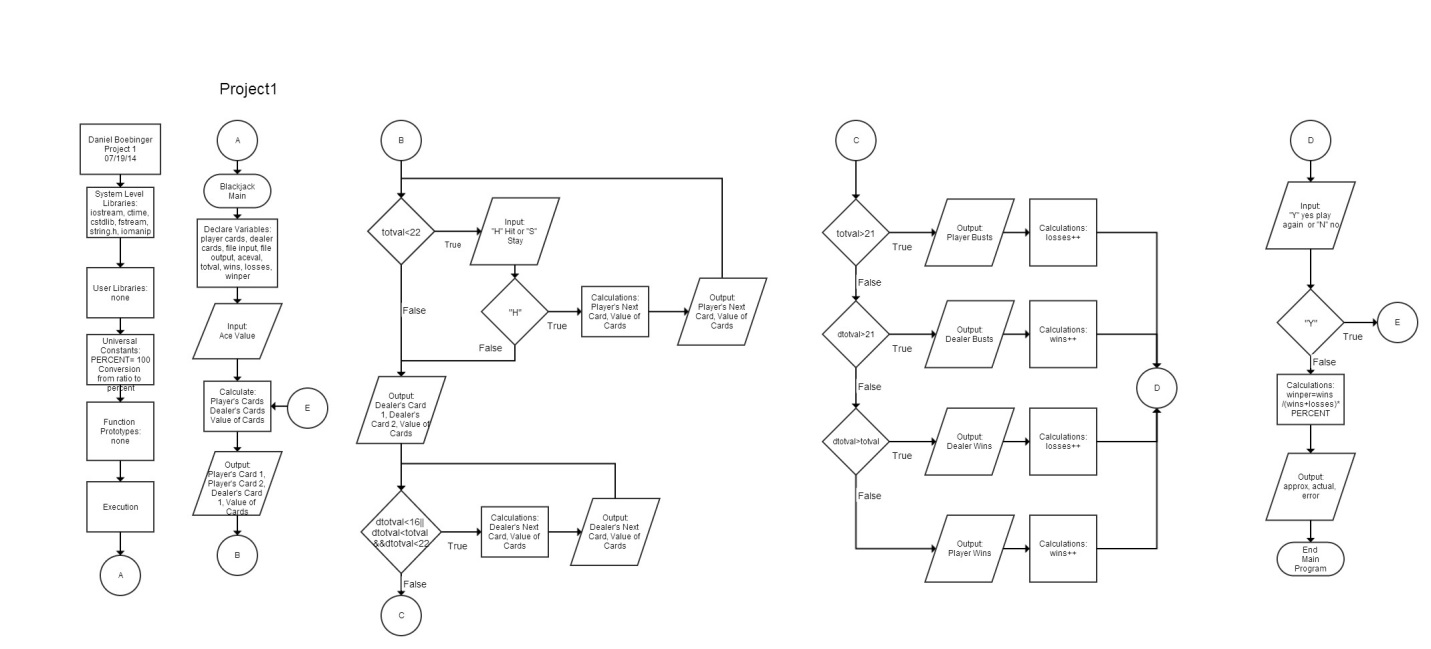
16”

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*

*Ace Value*

*Character length*

*Ace Type, Inputted from file*

*Character string to compare input string to*

*Character string to compare input string to*

*Character string to compare input string to*

*Outputs*

*Player's Card 1 value*

*Player's Card 2 value*

*Player's Additional card value*

*Player's Additional card value*

*Player's Total value of all cards*

*Dealer's Card 1 value*

*Dealer's Card 2 value*

*Dealer's additional card value*

*Dealer's additional card value*

*Dealer's total value of all cards*

*Number of wins*

*Number of losses*

*Winning percentage*

*Output to file*

*Set the Random Seed*

*Output Pre-Game Directions*

*Input Values*

*Open the Input file*

*Test File For Open Failures*

*Input Ace Type*

*Determine Game Type*

*If file reads High*

*Set Ace Value to High*

*Else if file reads Low*

*Set Ace Value to Low*

*Else if file reads Either*

*Set Ace Value to Either*

*Else Default to Either*

*Set loop=true*

*Do: Game Loop*

*Output Start of New Game Information*

*If Ace Values are High*

*Output Ace Value as High*

*Else if Ace Values are Low*

*Output Ace Value as Low*

*Else Default to Either*

*Determine Values for Card 1 and 2 for Player*

*Output Player's Cards*

*Output Player's Card 1*

*If Card 1 is Ace*

*Output Card 1 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 1 is Jack*

*Output Card 1 as Jack*

*Set Jack Value as 10*

*Else if Card 1 is Queen*

*Output Card 1 as Queen*

*Set Queen Value as 10*

*Else if Card 1 is King*

*Output Card 1 as King*

*Set King Value as 10*

*Else*

*Output Card 1 as Number*

*Output Player's Card 2*

*If Card 2 is Ace*

*Output Card 2 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 2 is Jack*

*Output Card 2 as Jack*

*Set Jack Value as 10*

*Else if Card 2 is Queen*

*Output Card 2 as Queen*

*Set Queen Value as 10*

*Else if Card 2 is King*

*Output Card 2 as King*

*Set King Value as 10*

*Else*

*Output Card 2 as Number*

*Determine Values for Card 1 and 2 for Dealer*

*Output Dealer's Cards*

*Output Dealer's Card 1*

*If Card 1 is Ace*

*Output Card 1 as Ace*

*Else if Card 1 is Jack*

*Output Card 1 as Jack*

*Else if Card 1 is Queen*

*Output Card 1 as Queen*

*Else if Card 1 is King*

*Output Card 1 as King*

*Else*

*Output Card 1 as Number*

*Output Dealer's Card 2 as Unknown*

*Calculate total value of Player's cards*

*Initialize Player Decision as Stay*

*If Total Value of Player's Card is Less than 22*

*Ask Player for Hit or Stay Decision*

*Player Hit Loop*

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

*Determine Value for Card3 for Player*

*Output Player's Card 3*

*If Card 3 is Ace*

*Output Card 3 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 3 is Jack*

*Output Card 3 as Jack*

*Set Jack Value as 10*

*Else if Card 3 is Queen*

*Output Card 3 as Queen*

*Set Queen Value as 10*

*Else if Card 3 is King*

*Output Card 3 as King*

*Set King Value as 10*

*Else*

*Output Card 3 as Number*

*Calculate total value of Player's cards*

*If Ace Value is Either Determine Best Ace Value*

*If Card 1 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate new Total Value of Player's Cards*

*If Card 2 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate new Total Value of Player's Cards*

*If Card 3 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate new Total Value of Player's Cards*

*If Card 4 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate new Total Value of Player's Cards*

*Output Total Value of Player's Cards*

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

*Ask Player for Hit or Stay Decision*

*If Player Hits*

*Determine Value for Card 4 for Player*

*Output Player's Card 4*

*If Card 4 is Ace*

*Output Card 4 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 4 is Jack*

*Output Card 4 as Jack*

*Set Jack Value as 10*

*Else if Card 4 is Queen*

*Output Card 4 as Queen*

*Set Queen Value as 10*

*Else if Card 4 is King*

*Output Card 4 as King*

*Set King Value as 10*

*Else*

*Output Card 4 as Number*

*Calculate total value of cards*

*If Ace Value is Either Determine Best Ace Value*

*If Card 1 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate New Total Value for Player's Cards*

*If Card 2 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate New Total Value for Player's Cards*

*If Card 3 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate New Total Value for Player's Cards*

*If Card 4 is High Ace and Total Value of Player's Cards is Over 22*

*Set Ace Value as 1*

*Calculate New Total Value for Player's Cards*

*Output Total Value of Player's Cards*

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

*Ask Player for Hit or Stay Decision*

*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 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*

*If Total Value of Dealer's Cards is greater than or equal to Total Value of Player's Cards*

*Output Dealer's Cards*

*Output Dealer's Card 1*

*If Card 1 is Ace*

*Output Card 1 as Ace*

*Else if Card 1 is Jack*

*Output Card 1 as Jack*

*Else if Card 1 is Queen*

*Output Card 1 as Queen*

*Else if Card 1 is King*

*Output Card 1 as King*

*Else*

*Output Card as Number*

*Output Dealer's Card 2*

*If Card 2 is Ace*

*Output Card 2 as Ace*

*Else if Card 2 is Jack*

*Output Card 2 as Jack*

*Else if Card 2 is Queen*

*Output Card 2 as Queen*

*Else if Card 2 is King*

*Output Card 2 as King*

*Else*

*Output Card 2 as Number*

*Output Total Value of Dealer's Cards*

*Output Dealer as Winner*

*Add Loss*

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

*Output Dealer's Cards*

*Output Dealer's Card 1*

*If Card 1 is Ace*

*Output Card 1 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 1 is Jack*

*Output Card 1 as Jack*

*Set Jack Value as 10*

*Else if Card 1 is Queen*

*Output Card 1 as Queen*

*Set Queen Value as 10*

*Else if Card 1 is King*

*Output Card 1 as King*

*Set King Value as 10*

*Else*

*Output Card 1 as Number*

*Output Dealer's Card 2*

*If Card 2 is Ace*

*Output Card 2 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 2 is Jack*

*Output Card 2 as Jack*

*Set Jack Value as 10*

*Else if Card 2 is Queen*

*Output Card 2 as Queen*

*Set Queen Value as 10*

*Else if Card 2 is King*

*Output Card 2 as King*

*Set King Value as 10*

*Else*

*Output Card 2 as Number*

*Calculate Total Value of Dealer's Cards*

*Output Total Value of Dealer's Cards*

*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*

*Determine Value of Dealer's Card 3*

*Output Dealer's Card 3*

*If Card 3 is Ace*

*Output Card 3 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 3 is Jack*

*Output Card 3 as Jack*

*Set Jack Value as 10*

*Else if Card 3 is Queen*

*Output Card 3 as Queen*

*Set Queen Value as 10*

*Else if Card 3 is King*

*Output Card 3 as King*

*Set King Value as 10*

*Else*

*Output Card 3 as Number*

*Calculate Total Value of Dealer's Cards*

*If Ace Value is Either Determine Best Ace Value for Dealer*

*If Card 1 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*If Card 2 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*If Card 3 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*If Card 4 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*Output Total Value of Dealer's Cards*

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

*Determine Value of Dealer's Card 4*

*Output Dealer's Card 4*

*If Card 4 is Ace*

*Output Card 4 as Ace*

*If Ace Value is High or Either*

*Set Ace Value as 11*

*Else*

*Set Ace Value as 1*

*Else if Card 4 is Jack*

*Output Card 4 as Jack*

*Set Jack Value as 10*

*Else if Card 4 is Queen*

*Output Card 4 as Queen*

*Set Queen Value as 10*

*Else if Card 4 is King*

*Output Card 4 as King*

*Set King Value as 10*

*Else*

*Output Card 4 as Number*

*Calculate total value of Dealer's cards*

*If Ace Value is Either Determine Best Ace Value for Dealer*

*If Card 1 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*If Card 2 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*If Card 3 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*If Card 4 is High Ace and Total Value of Dealer's Cards is over 21*

*Set Ace Value as 1*

*Calculate New Total Value of Dealer's Cards*

*Output Total Value of Dealer's Cards*

*Determine winner*

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

*Output That Dealer Busts and Player Wins*

*Add Win*

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

*Output That Dealer Wins*

*Add Loss*

*Else*

*Output That Player Wins*

*Add win*

*Else*

*Output That Player Busts*

*Add Loss*

*End of Game Loop*

*Ask to Play Again*

*If Player Chooses Yes Continue Game Loop*

*Default End Game Loop*

*While: Player Wants to Play*

*Calculate Win Percentage*

*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 55 |
| X | cin | Line 234 |
| X | endl | Line 55 |
| X | #include | Line 8 |
| X | short | Line 45 |
| X | int | Line 29 |
| X | float | Line 47 |
| X | char | Line 28 |
| X | character strings | Line 31 |
| X | bool | Line 95 |
| X | Math Expressions | Line 225 |
| X | Type Casting | Line 798 |
| X | Naming Constants | Line 29 |
| X | Combined Assignment | Line 290 |
|  | Format Input | NONE |
| X | Format Output | Line 801 |
| X | File Input | Line 66 |
| X | File Output | Line 802 |
| X | Relational Operators | Line 230 |
| X | if | Line 230 |
| X | if/else | Line 68 |
| X | if/else if | Line 77 |
| X | switch | Line 783 |
| X | Menus | Line 783 |
| X | Logical Operators | Line 238 |
| X | Validating User Input | Line 68 |
| X | String Compare | Line 77 |
| X | Increment and Decrement Operators | Line 495 |
| X | while | Line 238 |
| X | do-while | Line 97 |
|  | for | NONE |

**Future Improvements**

* Betting system
* Multiple players
* Additional blackjack rules and options
* Incorporate arrays and functions to 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>