**Blackjack Simulation – CS 4830**

**Katherine Lemmert**

**Due date: 12/14/2018**

**1. Introduction**

For my simulation, I have decided to simulate the simple card game called Blackjack. It is a casino game that is very easy to learn and play due to its simple rules.

Blackjack is a game where you have the player and the dealer. The player is given two cards and the dealer is given two cards. The player will go first and flip the two cards to see what cards they have. Each card has its own value:

Ace = 1\*

Two = 2

Three = 3

Four = 4

Five = 5

Six = 6

Seven = 7

Eight = 8

Nine = 9

Ten = 10

Jack = 10

Queen = 10

King = 10

\*Ace = 11 if running the strategy simulation

Depending on what the card values add up to, the player can either draw or hit. They are other moves in Blackjack, but for this simulation, I wanted to keep it relatively basic. I have 3 versions for the game.

Normal is where the player and dealer hit until they reach the total score of 17 or over.

There is a version where the player stops hitting at 18 instead of 17. This relates to one of my what-if question: What would the percentage difference be if the player was slightly riskier by stopping their hits at a score of 18 instead of 17 like the dealer?

And finally, there is a strategy version where the player will follow a specific strategy when playing against the dealer. This will answer my other what-if question: What percentage might the player have of winning if they followed to a strict strategy when playing against the dealer?

**1.1 What-If Question #1**

*What would the percentage difference be if the player was slightly riskier by stopping their hits at a score of 18 instead of 17 like the dealer?*

Normally, players don’t want to play too risky when it comes to trying to win a game, especially if it involves betting money. According to the dealer rules at most casinos, the dealer will stop hitting for another card once they reach the score of 17 or higher. This is because it is the safest path to go. It’s high enough that can help them win, and low enough to not bust all the time if they hit at 16. However, how much would a player’s chance of winning change if they were to stop hitting at the score of 18+ instead of 17 like the dealer? By changing their choice to hit or not at a number 1 higher than the safe zone of 17, will that help the player win more or will it hurt the player instead? By learning this, it will help determine whether it is worth taking the risk to player a bit riskier and, in real life, earn some more money.

In order to see the code of how I implemented this in my simulation, please go to the Blackjack.cpp file and look at line 198.

**1.2 What-If Question #2**

*What percentage might the player have of winning if they followed to a strict strategy when playing against the dealer?*

Blackjack can be considered a game of luck since the player will be given random cards and those cards help determine whether the use has a good chance of winning or not. However, there are strategies that can either help a user gain the advantage or keep them from losing money.

In this simulation, I have implemented a strategy the player will take to see if the player will gain any kind of advantage or not. The dealer will play as normal, but the player will either hit or stand depending on what value their cards add up to and what the dealer’s first card is. In my normal simulation, the dealer does not show their cards, but casinos vary in how the dealers play and this will take on the simulation of the dealer showing 1 card and not showing their other card. Also, due to the strategy given, the Ace’s value is 11 points instead of 1 for this type of game. Below is a chart that explains when the player will hit or stand based on the card values:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
| 5 | H | H | H | H | H | H | H | H | H | H |
| 6 | H | H | H | H | H | H | H | H | H | H |
| 7 | H | H | H | H | H | H | H | H | H | H |
| 8 | H | H | H | H | H | H | H | H | H | H |
| 9 | H | H | H | H | H | H | H | H | H | H |
| 10 | H | H | H | H | H | H | H | H | H | H |
| 11 | H | H | H | H | H | H | H | H | H | H |
| 12 | H | H | S | S | S | H | H | H | H | H |
| 13 | S | S | S | S | S | H | H | H | H | H |
| 14 | S | S | S | S | S | H | H | H | H | H |
| 15 | S | S | S | S | S | H | H | H | H | H |
| 16 | S | S | S | S | S | H | H | H | H | H |
| 17 | S | S | S | S | S | S | S | S | S | S |
| 18 | S | S | S | S | S | S | S | S | S | S |
| 19 | S | S | S | S | S | S | S | S | S | S |
| 20 | S | S | S | S | S | S | S | S | S | S |
| 21 | S | S | S | S | S | S | S | S | S | S |

|  |  |
| --- | --- |
| Player hits | H |
| Player stands | S |

*Table 1: Single-Deck Strategy*

This graph and strategy were used from a reference site that can be found in my references page on line 1. To see where how I implemented this in my simulation, please go to the Blackjack.cpp file and look at line 234.

**1.3 Setup and usage**

There are 2 parts to this simulation.

1. The zipped file simulation – a C++ based project created in Visual Studio 2017.
2. The report (a word document and a pdf file version)

This report is meant to describe the simulation and display the statistics I was able to gather to answer my what-if questions. I wrote my code in C++ as that is a language I am familiar with.

In order to run and use the simulation, please unzip the file where you’d like it to be. Then, open Visual Studio. This simulation was created in Visual Studio 2017, but it should be able to work on other versions. First:

Open simulation: File → Open → Project/Solution

Open the Blackjack.sln file, and the code should pop up. You may need to retarget the solution in order to make it compatible with your Visual Studio version.

Retarget solution: Project → Retarget Solution

Then, you can run the local windows debugger button to run the program. The will run the games and write the output of data to a text file that can be found in the Blackjack folder.

**2. Road Map**

This section will provide a guide to where each part of the requirements listed for the project grading is in the code.

**2.1 Event Model**

There are several events in the Blackjack simulation. These events are:

* Main event (line: 323 in Blackjack.cpp)

This event controls all the other events and runs the program.

* Play game (line: 395 in Blackjack.cpp)

This event will play the game, calling functions needed when performing the simulation actions as well as resetting the game when the next game is being played. It also determines the winner of the game and returns who won to the main function.

* Write file (line: 363 in Blackjack.cpp)

This event will take the data collected from the simulation and append it to a text file that can be accessed to later usages. It waits to see what type of game data it should print out, and when it is determined, will write to the file that specific data.

This simulation runs, at a default, 100,000 games per game type. Main event will trigger whether to play the games, end playing the games, or write to a file. Play game event will trigger a cleanse at the start to create the player objects, create a new deck, and shuffle the new deck. It will then go through and play the game until a winner has been determined. In the write file event, the simulation will determine which game type was played and will append data to the file with those game statistics.

**2.2 Models**

**2.2.1 Deterministic Mechanistic**

A deterministic mechanistic model is basically, a “non-random how”. This means that the model that can be determined through parameters and a condition that must be met. There are no random variables in this type of model. An example of this would be a mathematical function which has inputs that go into an equation and you will get the same result with the same inputs every time.

A very simple example of this in the simulation is when I am checking the bounds of the score each player has gotten in the game. The player and dealer can not go over the score of 21 or they will “bust” aka lose the game. I used a very simple if-else statement that checks the score and then returns a Boolean that indicates if the player is within range or not. Below this line displays the code that shows deterministic mechanistic:



*Deterministic Mechanistic code (line 123 in Blackjack.cpp)*

Above shows the very simple equation where, any score 21 or below will return 1, indicating that the player is within bounds and hasn’t busted. If the score is above 21, then the function will return -1 to indicate that the player or dealer has busted.

**2.2.2 Deterministic Empirical**

A deterministic empirical model is basically calculating a formula based on observation. In other words, based on the data observed, non-random equations will be used to calculate whatever is being calculated. For this, an example could be from a snippet of code from the Blackjack simulation when it is playing through the strategy type game. Based on whatever the dealer’s revealed card is, the player will make the decision whether to hit or stand until the player reaches the score of 17 or higher. The result will always be the same, the player will stop hitting once the score is 17 or higher. However, when they stop hitting is determined on what value the dealer’s card is. On the next page will show the code that displays deterministic empirical:



*Deterministic Empirical code (line 257 in Blackjack.cpp)*

**2.2.3 Stochastic Mechanistic**

A stochastic mechanistic model is a mathematical equation that has at least 1 random variable. That means that different outputs can be made using the same type of equation. A good example of this in the simulation would be the shuffle function. That function will take the deck of cards and use a random number generator to generate a random variable to help place the cards in a different spot. That variable is an example of stochastic. This equation will provide a different outcome every time, yet the same equation to solve where the cards are placed is used, meaning it’s also mechanistic. On the next page will show the code that displays stochastic mechanistic:



*Stochastic Mechanistic code (line 48 in Blackjack.cpp)*

As stated in the code, using the random number generator will give an index between 0 and the card deck size. The equation will always give a number between those numbers, but the number inside that parameter is random.

**2.2.4 Stochastic Empirical**

A stochastic empirical model basically means there is a random element based off an observation. A good example for that the dealer cheats during random games. It would be bad if the dealer a lot of money, so if the player went to a crooked casino, the dealer would cheat at random games to stay ahead of the player and not lose money. The stochastic part would be the dealer choosing when to cheat. It would be at random times so the player won’t suspect the dealer cheating all the time. However, based on the card values the dealer has, the dealer can slip a card in the deck like an Ace (if it’s valued at 1) or a 2 to help raise the score without busting. In this simulation, the dealer will slip in a 2 if the score is 19 or less. The dealer determines when and how long to cheat by using a random number generator. There is the use of rejection sampling (that makes it empirical) by deciding to cheat when the dealer is winning less than 40% of the time. Below this line is the code I used to show stochastic empirical:



*Stochastic Empirical code (line 514 in Blackjack.cpp)*

When the dealer cheats and for how long are completely random. However, the dealer will use rejection sampling by choosing whether to cheat in the first place by determining if the winning percent is low enough. Otherwise, the dealer will play like normal.

**2.3 pRNG**

A pRNG is a pseudo-random number generator that uses an empirically derived PDF with either a rejection sampling or inverse transform sampling. In other words, this is a stochastic empirical model. This model is already explained and used in the stochastic empirical section on page 9. Since this pRNG does not count against the item requirement, I will be using that as the pRNG.

**3. Text File**

In the Blackjack simulation, an output text file of the games data is being put in there. This file can be imported into an Excel file. The first part of the text file:

Game 0

Game 1

Game 2

That represents the timestamp, or the game type that is being recorded. On that line will contain what type of game along with that games data. Game 0 represents a normal simulation of the game. Game 1 represents the first what-if simulation which is: *What would the percentage difference be if the player was slightly riskier by stopping their hits at a score of 18 instead of 17 like the dealer?* Game 2 represents the second what-if simulation which is: *What percentage might the player have of winning if they followed to a strict strategy when playing against the dealer?*

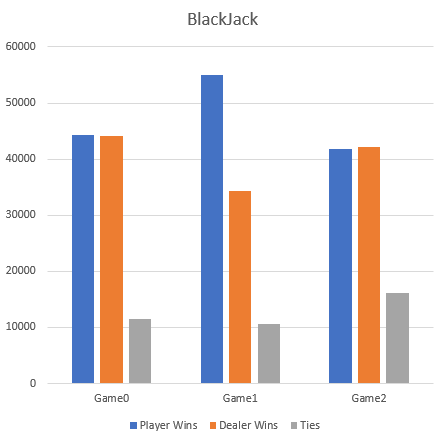
The first column of numbers represents the amount of times the player has won. The second column of numbers represents the amount of times the dealer has won. And the last column represents the number of ties each the player and dealer had throughout the games. These numbers will total up to 100,000 each line, as there were 100,000 games simulated to get this data.

**4. Analysis**

**4.1 What-If Question #1**

*What would the percentage difference be if the player was slightly riskier by stopping their hits at a score of 18 instead of 17 like the dealer?*

In order to determine whether it is advantageous to be risker in play, I decided to make a bar chart to not only see the results comparing the games, but also see the difference between the two.



*Bar Chart of the 3 games types comparing wins/loss/ties*

This bar chart shows all 3 game types, but the types we want to look for are Game0 and Game1 in order to answer the question. Just from looking at the graph, there is a significant difference to how many times the player wins against the dealer in the riskier game compared to the normal game. The ties are around the same amount. However, this isn’t enough to see if it would be it was worth playing this way than it is the normal way.

For my null hypothesis, the dealer wins 30% or less. If we reject this null hypothesis, then this is an advantage to the player. In order to figure that out, I must find the percentage won of from game1.

34312 / 100000 = .34312 \* 100 = 34.312%

Since 34% is larger than 30%, we are rejecting the null hypothesis. That means, there is at least 70% of the player winning or getting ties by using the riskier method. Now, to compare that to the normal games, we would need to do the same calculation.

44180 / 100000 = .4418 \* 100 = 44.18%

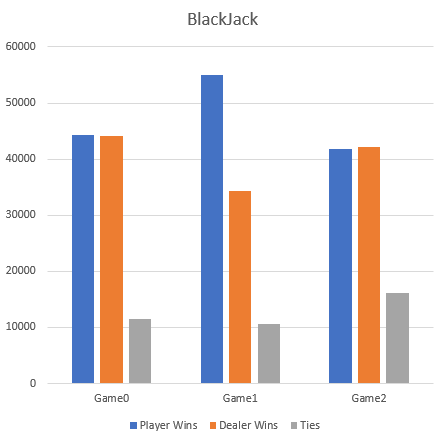
That is approximately 44% of the dealer winning, meaning the player is winning or getting ties by at least 56%. If we compare the two games:

66% - 56% = ~10% (since we are ignoring the decimals)

There is a 10% difference of the player winning compared to the two games. **In conclusion, the player is winning 10% more by playing riskier than if they were to play like the dealer.**

**4.2 What-If Question #2**

*What percentage might the player have of winning if they followed to a strict strategy when playing against the dealer?*

 Since we want to see what percentage the player may have of winning if they follow a strict strategy, we don’t need to do a whole lot of analyzing. However, by using a bar chart, we can help make quick decisions as well as compare with the normal game version to see why some people might implement this type of strategy. Below is the bar chart used:

*Same Bar Chart of the 3 games types comparing wins/loss/ties*

As you can see, this bar chart shows that there is a pretty equal chance of winning or losing against the dealer (although slightly less chance of winning through these calculations). However, bar chart is not enough to make any kind of conclusion or a way to answer the question. So, we must find the percentage of how much the player wins. We can do this by taking the number of times the player has won compared to the number of games.

41737 / 100000 = .41737 \* 100 = 41.737%

So, the player will win approximately 42%. But, we need to also take into calculations of the ties as the player wins through those as well. So, to redo the calculation:

14737 + 16168 = 57905 / 100000 = .57905 \* 100 = 57.905%

With this, we can answer the question that the player will win around 58% of the time. That makes this strategy used a relatively safe path if the player doesn’t want to lose a lot of money but gain a little.

However, another question comes to mind which is how does this compare to the normal games? By doing the same calculations:

44241 + 11579 = 55820 / 100000 = .5582 \* 100 = 55.82%

In the normal type of game, the player wins or ties around 56%. That percentage is extremely close to the percentage for the strategy game type. In this case, it wouldn’t really matter if the player played normally or with this specific strategy. However, the strategy could be a good idea for players who are new and do not know when to hit or stand. **In conclusion, if the player plays by this specific strategy, they will win ~58% with wins and ties.**

**5. References**

This site shows the strategy I used. It is the Blackjack basic strategy – single deck – European style where dealer hits on soft 17, double after split not allowed, and there is no surrender.

https://wizardofodds.com/games/blackjack/strategy/calculator/