Appendix B
Technical Description of Code

Function and Variable Descriptions
The simulation was scripted in MATLAB software. There is one main script with 11 functions to help run different varieties of hands. Global variables were not used in the program, this caused the main parameters to be passed into, and out of each function as arguments. These parameters included: the deck, the count, the player's cards, the dealer's cards, and the outcome for each hand. The following subsections describe the variables and functions within the simulation.

Variables
The following parameters were the main variables in the simulation. Each variable description goes into detail about what the variable represents and how it is used in the simulation.

maxbet and countthreshold
Maxbet and count threshold are the two parameters of interest for this study. The count threshold is the value the count needs to exceed for the player to increase the wager on each hand. Max bet is the amount that the wager is increased to. The count threshold took on all integer values in between 10 and 20; while maxbet was studied at values including: 10, 25, 50, 75, 100, and 200. We believed these parameters were realistic based on how often the count would actually reach that threshold, and the amount of money the player started with.

pcards, p2cards, dcards
Pcards is a size ten vector whose indexes represent cards slots 1-10 for the player. If there is no cards in a particular slot; which is often the case for higher number slots, the value is 0. If a card is placed in at an index; which is always the case for at least the first two cards, the number in that index represents the value of a single card. Dcards works the exact same way, except it represents the dealer's hand. The first slot of the dealer's hand is the up card. P2cards is only used in the event that the player splits a pair, and represents the second hand for the player.

deck
The deck is also a size ten vector. Each index represents a card value 1-10 (aces in the deck are represented as 1). The number in each slot represents the number of cards that are left in the deck with a value that is equal to that index. For example, a brand new shuffled single deck of cards would have a value of 4 for each index 1-9; to represent the four cards in a deck that have a value equal to that index. Then a value of 16 would be in the 10 index to represent all cards that have a value of 10 including: 10's, Jacks, Queens, and Kings. This setup allowed the deck to be turned into a vector of probabilities that sum to one with the MATLAB command ">>p=deck/sum(deck);" which was necessary for the deal function. After a card is drawn, the value in that index is decremented by one. To reshuffle the deck, we set a value of 20 cards to be the minimum cards left. Before each hand, if the total number of cards left is under that minimum, then all the values at each index are reset

(4*(number of decks) for values 1-9, 16*(number of decks) for 10's).

count
The count is a single number variable based off the Hi-Lo card counting system. It is only changed in the deal function; and the only time that it is used is to check which bet to place (maximum or minimum). Each card value 2 through 6 increments the count by 1, tens and aces decrement the count by 1, and cards 7, 8, and 9 are neutral.

bet
This variable represents the amount that is wagered on a single hand. In this simulation the player's strategy is to bet the minimum amount at the table ($5.00) until the count reaches a certain threshold. After the variable 'count' exceeds that threshold the bet is increased to a parameter 'maxbet' which varies depending on the trial trial. Of course, the player cannot wager more than the amount of money that is left, this is accounted for by the player leaving the table when $500.00 are left.

outcome
The variable outcome represents a value that would be multiplied by the variable 'bet' to correctly adjust the amount of money the player has. For example, when the player simply wins or loses a hand, outcome is set to equal 1 or -1 respectively. If the player doubles down the outcome is always multiplied by 2. Blackjacks have a return of 3:2, therefore when the player receives a blackjack outcome is set to equal 1.5. When the player splits a pair, the two outcomes are determined individually and then summed to be one single outcome.

Functions
Eleven functions and/or scripts were used to run the simulation. Not every file was used on each hand, but all were necessary to cover the variety of different scenarios that could take place on any hand. Each file is described in detail below.

game.m
This file is the main script that is called from the MATLAB terminal. It first initialized all vectors that were used to collect data. It then entered a *for loop* from 1 to 1000 to have 1000 games in each trial. Within the *for loop* the following things occurred: the deck is created, the count and initial money values are set, and both the player's and dealer's hands are initialized. After the pre-game values were set, the script entered a *while loop* that simulated hands of blackjack until the variable 'money' was below 500 or above 1500 (representing the player leaving the table after a profit or loss of 500). The first check within the *while loop* is to determine if the cards are below some reshuffle point, in that case the values in the variable 'deck' were reset. Each iteration through the *while loop* represented one hand. First, the player placed their bet. Next, two cards are dealt to the player and dealer. Then the function 'blackjack.m' is called to determine the strategy and outcome. All the data is collected in the previously initialized arrays and displayed for the user to analyze.

blackjack.m
This is the first function called by *game.m*. It used the array of the player's cards to first

determine if the player has a pair, soft hand, or hard hand. Then it makes the appropriate function call to determine which strategy to use based on the player's hand.

hard.m, soft.m, pair.m

These functions each contain a matrix with the rows representing the player's hand, and the columns representing the dealer's upcard. Each index in the matrix contains a number that represents the different possible initial strategies for blackjack: split, stand, double down, and hit. Each of these functions are called with the appropriate input to represent the player's hand and the dealer's up card. The dealer's upcard of an ace is considered an 11 in the matrix; so if that is the case, the previous value of 1 is set to 11 before the strategy is determined. Each of these files are open sourced code from MATLAB found online.

hardhand.m

The function *hardhand.m* is called from *blackjack*.m if the player does not have an ace or a pair. It immediately entered a *while loop*. The loop only breaks after the variable 'outcome' is initialized to be returned (which is guaranteed to happen the first time through the loop). This is to allow for the method to skip the rest of the code as soon as an outcome is decided. The first check is to see if the dealer has a blackjack; in that case the outcome is set to -1 and the *while loop* is broken. Next, the basic strategy is determined by calling *hard.m* with the player's hand and the dealer's upcard. Then the player is dealt to accordingly. The next check was to determine if the player had busted (reached a value of over 21); in that case, the outcome is set to -1 and the *while loop* is broken. The three final steps are: dealing to the dealer, checking if the dealer busted, and finally comparing the player's hand to the dealer's hand. In each case, the appropriate outcome is set and the *while loop* is broken. All outcomes are multiplied by a variable representing whether or not the player has 'doubled down' before they are returned.

softhand.m

The *softhand.m* function works in the same way as *hardhand.m.* The main difference is accounting for the ace card, which can count as an 11 or a 1. There is also a check to determine if the player has a blackjack (which can only happen if the player has an ace). Blackjacks have a 3:2 payoff, unless the dealer also has blackjack, in that scenario the outcome is a push.

pairhand.m

This function determines the appropriate strategy in the same fashion as *hardhand.m* and *softhand.m*. If the basic strategy says to not split the pair, the hand is played out as if it is a hard hand. Since basic strategy says to always split aces, we can assume if the player does not split, the hand does not contain any aces. If basic strategy says the player should split, a new array is created to represent the player's new hand. Then the player's first card of the new hand is initialized to second card of the first hand. After that a new card is drawn for the second card of both hands. The *playpair.m* function is then called to determine the outcome.

playpair.m

The *playpair.m* function assists *pairhand.m* in calculating the results of each hand. This

function was needed to compare the player's two hands against the dealer's one hand. Each hand is played out with the same function checks in both hardhand.m and softhand.m. If the outcomes for both hands have not yet been determined by blackjacks or busts, the dealer is then dealt the remaining cards for the hand. Each of the player's hands are compared to the dealer's hand individually and an appropriate overall outcome for the hand is determined.

deal.m

This function draws one card from the deck and gives it to either the dealer or the player. The card is then removed from the current deck and the count is adjusted accordingly. A card is drawn randomly from the deck of cards that have not yet been played using the principles from the discrete inversion algorithm. The amount of cards for each number 1, 2, 3 … etc. that are left in the deck are converted into probabilities by dividing each one by the total number of cards left in the deck. A uniform random number between 0 and 1 is then generated and the discrete inversion method is used to chose a random card.

dealto.m

This function draws cards for the player or dealer until the player has a hand that is over 16 (or over a soft 17) as per basic strategy and rules for the dealer at the table. Before a new card is dealt, there is a check to see if an ace exists in the hand; and if so, does setting the ace to be equal to 11 give a hand total greater than soft 17 and less than 22. If that is the case, the ace is set to be equal to 11 and the dealing stops. Otherwise, cards continue to be dealt until the sum of the hand exceeds 16.