ACM 11: Blackjack Project

This is one possible topic for the ACM11 MATLAB project. Difficulty rating: easy to medium. 100 points. 10/30/08.

Background. In this assignment, you will implement a simple version of Blackjack, where a human user can play against a computer "dealer". By the end of the project, you should be familiar with writing functions, using loops, and asking for user inputs. Please see `http://en.wikipedia.org/wiki/Blackjack` for background and rules on Blackjack.

Basic Version

For the project, use standard Blackjack rules with the following modifications and clarifications:

1. The player begins the game with a specified amount of chips; the dealer is assumed to have an infinite supply of chips. At each round, the player specifies an arbitrary number of chips to bet. All bets are made in whole numbers of chips (e.g. no .3 chip bets).

2. At every round, the objective of the player is to make the sum of the cards in his/her hand as close to 21 as possible without exceeding 21 (exceeding 21 is a "bust").

3. At the end of every round, the winner is the person with the highest sum of cards not exceeding 21. The winner gets back their bet as well as an equal amount from the opposing player; the loser forfeits his/her bet. In the event of a tie between a user and dealer, each user gets his/her money back.

4. Cards 2-10 are worth face falue; face cards (Jack, Queen, King) are worth 10 points. The Ace is worth 11 points unless this would cause a bust, in which case the Ace is worth 1 point; alternatively, your program could ask the user to decide which value to use for an Ace when the situation arises.

5. At every round, the player and dealer are both dealt two cards each (we can assume an infinite deck of cards, so this is "with replacement"). The player is allowed to see one of the dealer's cards. The player can choose to either hit (receive one more card) or stand. The player may hit repeatedly until they have 5 cards, at which point they must stand. After standing, it is the dealer's turn. The dealer "hits" automatically until the dealer's hand is 17 or higher (or until a bust).

6. The order of play is important! If a player busts, he/she loses, since the player played first. If the player does not bust but the dealer does, then the player wins. If the player has a blackjack (total of the cards is 21) and the dealer does not, then the player wins back his/her money plus 150% of the bet; if the dealer also has a blackjack, it is a tie. If the dealer has a blackjack and the user does not, then the user only loses his/her bet (not 150% of the bet).

7. The program should keep track of the statistics of play, as well as the chip count. The game is over when the user specifies, or when the user has no chips left.

8. The program should prompt the user for input, such as the amount of money to bet, and whether to hit or stand. The program must be robust to input! If the user specifies an invalid action, or an invalid amount of chips, the program should send a warning and ask for the input again.

9. The main function, `BlackJack.m`, should first ask the user how many chips to play with (default: 100). Inside this function, you may want subfunctions, such as `Card2String` (coverts a "numeric" card to a nicely formatted string that the user can understand, such as "Queen"), and `ScoreHand`, which scores the hands and determines a winner.

10. To prompt the user for input, you may uses either `input` (which displays a string and then waits for a keyboard response), or `inputdlg` which will ask for information from a popup box.

11. Other helpful MATLAB functions: `disp` or `fprintf` for displaying information, `nargin` for checking number of input arguments, `while`, `if`, `elseif` for logic operations, `rand` for random number generation, and `num2str` to convert a number to its string representation.

Advanced Version

Here are some variations to add (difficulty: medium to medium-hard). You are not required to do any of these, but they may be very interesting!

1. Allow the user to double-down and/or split.

2. Use a finite deck of cards, so that cards are dealt without replacement. Use a deck of 5 cards, and "reshuffle"' the deck when only one deck of cards remains.

3. Create a nice graphical interface (using `GUIDE`) for the program.

4. Create a program that plays a computer dealer against a computer player. Use "basic strategy" for the computer player (see the Wikipedia article for a table of basic strategy).

5. After implementing (4), play several thousand games and keep track of the statistics (don't worry about chips, just record the win/loss record). What are the dealer's odds?

6. After implementing (4), play with an infinite deck, but bias the draws toward high cards. What are the dealer's odds now?

7. Bonus: After implementing (4), play with a finite-sized deck, and allow the computer player to count the cards. Use a simple card-counting scheme, and bet accordingly. Can the player expect to make money? What are the odds, what is the expected earnings/loss, and what is the variance on the earnings/loss?

Special thanks to Esther Wang for describing the project and to Misha Dhar for the project idea.