

## **ENEE114 Fall 1999 Homework #3, Due Saturday October 9, 11:59PM**

Disclaimer:

The following problems are intended as illustrations of C programming concepts, in particular, functions, arrays, and file i/o, and should not be interpreted as an inducement to gamble or visit Las Vegas (Vival!

(Material below excerpted from "<http://gamblingtimes.com/gtbasbj.html>")

### **BLACKJACK**

*There are many basic strategies for blackjack, depending on the casino's rules and the number of decks used. The basic strategy outlined here is based on the four-deck game as played in Las Vegas. The object of the game is to beat the dealer with a total equal to or less than 21, without going over 21 or bust.*

#### *Rules of the Game*

*Before any cards are dealt, the player must wager. He does this by placing his bet in the designated space in front of his table position. The dealer then deals two cards to each of the players, and two to himself (one of the dealer's cards is dealt face up and one is dealt facing down). Face cards (kings, queens and jacks) count as 10, ace counts as one or 11 (as the player chooses) and all other cards are counted at their face value.*

*BLACKJACK---If the player's first two cards are an ace and a 10 or face card, he wins. However, if the dealer also has a blackjack, it is a standoff, as are all ties or pushes. A winning blackjack pays the player 3 to 2.*

*HIT or STAND---Hit means to draw another card (which the player signifies by scraping the table with his cards or a similar hand motion). Stand means no more cards (which the player signals by placing his cards under his wager or moving his hand in a horizontal direction. If the player hits and busts (goes over 21), he immediately turns his cards over and his wager is lost.*

*DOUBLE DOWN---The player is allowed to double the bet on his first two cards and draw one additional card only to improve his hand.*

*SPLITTING PAIRS---If the first two cards a player is dealt are pair, he may split them into two separate hands, bet the same amount on each and then play them separately. Aces receive only one additional card. After splitting, A-10 counts as 21 and not as blackjack.*

*INSURANCE---If the dealer's up card is an ace, the player may take insurance, a bet not exceeding one-half his original bet. If the dealer's down card is a 10 or any face card, the player wins 2 to 1. Any other card means a win for the dealer.*

*SURRENDER---Where permitted, a player may give up his first two cards and lose only one-half his original bet.*

*The dealer must draw on 16 and stand on 17. In some casinos, the dealer is required to draw on soft 17.*

### The homework problems:

Problem 1 (25%): blackjack building blocks: card, dealer, and player functions to support the game.

Problem 2 (50%): interactive blackjack: integrate functions from problem 1 into a working game.

Problem 3 (25%): blackjack strategy simulation: use problem 2 to test the blackjack strategy of the reference above.

### Suggested approach:

Write programs for problems 1-3 for the basic game ( no double down, splitting pairs, insurance, or surrender) and get them working, then go back and add the complications.

### **Problem 1:** (hw3p1.c; hw3p1.log) blackjack building blocks

Build and test the functions below. You may wish to further decompose some of these into subsidiary functions, and, if there are such subsidiary functions which can be reused, you will be rewarded gradewise.

```
void newdeck(int decks[]); // initialize the four decks of cards
void newchips( void); // i/o to player: sell the player some chips
void shuffledeck(int decks[]); // randomize cards (1000 pairwise exchanges)
int getplayerbet( void); // i/o to player:
void playhand( int decks[], int topcard); //play one hand of the game
    char getplayeraction( void); // i/o to player: (\0=bad answer; hit,
//stand, double down, split,
//insurance, surrender, quit
void hitplayer( int hand[], int topcard); // deal one card to the player
void show_playerhand( int hand[]); // update players card display
void show_dealerhand( void); // update dealers card display
void playdealerhand( void); // play dealers hand
void hitdealer( int topcard); // one card to the dealer
```

For the functions above (and others you add) assume the following data/ data structures/operations

```
int decks[4*52]; //array which holds a set of numeric values 0..207 which are equivalent
//to the 208 cards in four decks. The cards map to the integers 0..207
//by deck (1,2,3,4), by suit ( Club, Diamond, Heart, Spade), by value
//(1..10,Jack,Queen,King,Ace).
int topcard; //pointer to next card to play in deck (0 to start).
int hand[]; //array holding values copied from decks[]
'shuffle' --use rand() function to choose two cards in decks[] and swap them.
```

The demonstration of the above:

Build a "robot blackjack program" using the above functions and starting with a new deck:

1. 'Player' (e.g. algorithm) bets "\$1"
2. Player stops if "blackjack" on initial deal, o.w. hits one card (might bust).
3. Dealer hits until 17 or bust.
4. All hands are displayed after each card; game winner and amount after each game.
5. Play continues until deck is all played; player winnings (losses) are printed at end.

**Problem 2:** (hw3p2.c; hw3p2.log) Turn problem 1 into a 'real' (interactive) game of blackjack; start a 'new' deck whenever there are fewer than twenty cards in the old deck.

**Problem 3:** (hw3p3.c; hw3p3.log) Turn problem 2 back into a robot problem where the 'robot' plays the blackjack strategy as shown at "<http://gamblingtimes.com/gtbasbj.html> and in the table (bjstrategy.txt) provided. Make sure your solution uses the text table by reading it as a file, as we would like to test alternative strategies by modifying the table.

**Blackjack strategies/strategy table (for Problem 3):**

Borrowing from the table at the website referenced above, we are going to describe blackjack strategies in a text table whose entries describe various player actions as a function of what's in your hand and what's showing in the dealers hand. The format for one row of this table is:

Format of one record in the table:

<type>,<value or card>,<strategy for dealer upcard=2>,....,<strategy for dealer upcard=Ace>\n

where:

<type>: the type of hand: Hard, Soft, Pair

<value>: depends on <type>.

For type=Hard, value = card total;

for type=Soft, value = other (non-Ace) card;

for type=Pair, value = value of one card (in pair).

<strategy for dealer upcard> = hit, stand, double down, split, insurance, surrender

Sample record (fourth line of table at website):

**"H, 12, h, h, s, s, s, h, h, h, h, h\n"**