

**Computer Science 215**  
**Tools and Techniques for Software Development**  
**Summer 2002**  
**Project 3 – Blackjack**

**Due: Monday, 6/10/2002 1:00 p.m.**

For this assignment, you will write a program for the card game Blackjack. The program should be menu-driven as follows:

**Please select one of the following options:**

- 1 Play**
- 2 Set seed for deck**
- 3 Shuffle cards**
- 4 Read in a loaded deck**
- 5 Print stats**
- 6 Quit**

**Option:**

Upon start-up, the program should initialize the deck of cards and the statistics, and allow the user to enter options until Quit is selected. Numerical error checking for the menu item is required (i.e., make sure the value is in range). Information for each of the options follows after a brief description of the rules of Blackjack.

In Blackjack, a player tries to get a hand that has a total value closer to 21 than the dealer without going over 21. The player can opt to draw additional cards and stop whenever desired, while the dealer (or in our case, the computer) must play by a strict set of rules. The value of the hand is evaluated according to the following rules:

cards from 2 through 9 are valued as indicated  
10, Jack, Queen, and King are valued at 10  
the Ace can count as 1 or 11, whichever makes the best hand

The suits on the cards have no meaning in the game.

To illustrate how the Ace works, assume that the player is dealt (Ace, 6). This hand can be either 7 or 17. If the player stops there, it will be 17. Let's assume the player draws another card to the hand and now has (Ace, 6, 3). The total hand is now 20, counting the Ace as 11. If instead, the player had drawn a third card, which was an 8, the hand would be (Ace, 6, 8) which totals 15. Notice that now the Ace must be counted as 1 to avoid going over 21.

The player will be playing against the dealer, whose hand will be determined by the rules set forth below. If there's a tie on total hand values, the player wins.

A description of the menu options follows:

**1 Play** – When this option is selected, the user enters a playing mode. The player and dealer are each dealt 2 cards from the current deck in alternating fashion (i.e., player gets 1 card, dealer gets 1 card, player gets 1 card, dealer gets 1 card). The sum of the card value for each hand is printed in parentheses, followed by the cards themselves in sorted order (first by value, then suit; e.g., **6 H 6 S A C**):

**Option: 1**

**PLAYER'S DRAW**

**Player: (16) 6 H 10 H**

**Dealer: (12) 2 H K C**

**Would you like another card? y**

Note that the first set of hands is preceded by the title **PLAYER'S DRAW**, indicating that the player may now draw cards to improve his hands. If player responds with **Y** or **y**, the next card from the deck is drawn, displayed on the screen, and added to the player's hand. This interaction continues until the player answers with **N** or **n**, or until the player's hand is  $\geq 21$ . If the player or dealer draws a hand totaling 21 on the first draw, **BLACKJACK!!** should be printed at the end of the **Player:** line, and the program should skip asking the player if he'd like another card. A sample session follows, continuing from the previous example.

**Card drawn: 4 D**

**Player: (20) 4 D 6 H 10 H**

**Dealer: (12) 2 H K C**

**Would you like another card? n**

**DEALER'S DRAW**

**Card drawn: A D**

**Player: (20) 4 D 6 H 10 H**

**Dealer: (13) 2 H K C A D**

**Card drawn: Q S**

**Player: (20) 4 D 6 H 10 H**

**Dealer: (23) 2 H Q S K C A D**

**FINAL HANDS: Player 20 Dealer 23**

**You're a winner!!!**

\*\*\*\*\*

Current statistics:

Number of games: 1

Number of wins : 1

Winning % : 100.0

\*\*\*\*\*

Would you like to play again? (y/n) y

Once the player has finished drawing cards, play enters into **DEALER'S DRAW**. The dealer continues to draw until the total value of the dealer's hand is greater than the player's or greater than 16. If the player's hand is greater than 21, or the dealer's hand is greater than the player's hand when the player finishes adding cards, the dealer stands with his original hand and wins. For this case, **DEALER STANDS** must be printed on the screen. If the dealer's hand is greater than 21, the player wins. Note that the dealer must stop drawing cards if his hand is greater than 16, even if he knows the player will win. Also, whenever possible, an Ace counts as 11 in the dealer's hand, even if it will cause the dealer to stand with a hand that won't beat the player's.

After both the player and dealer have drawn, the final totals for the hands are printed, as shown above. If the player wins, the message **You're a winner!!!** is displayed; otherwise, the message **Better luck next time.** is printed.

Statistics are printed after each hand, at which time the cards are shuffled. Note that play continues until the user answers **N** or **n**.

**2 Set seed for deck** – Since your program depends on random numbers for shuffling, we need a way to make it play a different set of games each time the program is run. We can do this by seeding the random number generator an integer value. Note that by seeding the random number generator with a specific value, we will play the same set of games, which may be useful for testing your program. To get the seed, prompt the user for a seed value; otherwise, use the method provided on the website to get a semi-random seed.

**3 Shuffle cards** – This option allows the user to shuffle the deck of cards. Upon program start-up, the cards are not shuffled, so they must be shuffled at least once by selecting this option before play begins. When playing, the cards are shuffled once **after** each hand. You should use the following algorithm to shuffle the cards:

```
for (each index in the deck array)
    get a random number to use as a second index and swap
    the cards at the two indices
swap
```

**4 Read in a loaded deck** – This option will allow you to test certain hands, dealt from a loaded deck. The file should contain 52 lines, each with a numeric value (2..14), a space, and a single character for suit. Use the code from the website to help write this function. A sample deck with a double blackjack is also available.

5 **Print stats** – This option will allow the user to print the statistics as given above, including the two lines of `'*'`s.

6 **Quit** – The program should terminate when this option is selected.

For this program, you **must** use **structs** and **typedefs** for the following types:

**CARD\_T** – a struct containing two fields: an integer **value** and a character **suit**  
**HAND\_T** – a struct containing two fields: an array of 11 **cards** and an integer **num\_cards** indicating the number of cards currently in the hand  
**DECK\_T** – a struct containing two fields: an array of **cards** and a pointer to the current **top** of the deck  
**STATS\_T** – a struct containing the number of wins and the number of games played

Additionally, you **must** define functions with the **exact** prototypes listed below (these will *help* you):

```
void init_cards (DECK_T *deck);          /* initialize the 52 cards */
void init_stats (STATS_T *stats);       /* initialize stats fields */
void play (DECK_T *deck, STATS_T *stats);
void set_seed ();
void read_deck (DECK_T *deck);
void shuffle_cards (DECK_T *deck);      /* shuffles the deck */
void swap (CARD_T *p, CARD_T *q);       /* swaps values of two cards */
void deal_card (CARD_T *hand, DECK_T *deck);
void sort_hand (HAND_T hand);
void print_card (CARD_T card);          /* prints a single card */
int hand_total (HAND_T hand);           /* gets total value of hand */
void print_stats (STATS_T *stats);      /* prints the stats */
```

`init_cards` creates the `deck` of cards as follows:

```
2C 3C 4C ... KC AC 2D 3D 4D ... KD AD 2H 3H 4H ... KH AH 2S 3S 4S ... KS AS
```

`play` is the main playing function; it is not exited until the user enters `n` or `N`.

`set_seed` prompts the user and initializes the random number generator.

`read_deck` reads in the loaded deck and stores it in `deck`.

`deal_card` assigns the current top card on the `deck` to `hand`.

`sort_hand` sorts the cards by value/suit in `hand`.

On the web page, you will find files called `blackjack.h`, `cards.h`, and `cards.c` with some partial code that you need to complete. You must also keep these files separate and compile and link them. The main function, along with functions specific to Blackjack and this program, should be in a program called `blackjack.c`. It should include any `.h` files it needs. All of the functions related to cards in general should be in `cards.c` (see `cards.h` for a list of these functions). Compile the program as follows:

```
cc blackjack.c cards.c -o blackjack
```