Programming Assignment 4: Blackjack

COP 3014 - Spring Term 2010
Point Value: 100 points
Project Due Date: Wednesday March 24, 2010

IMPORTANT NOTE: This program is longer and more complex than previous assignments. Be sure to begin work on it as early as possible and attend lectures and recitations for important tips.

Learning Objectives

- To write a program making use of multiple functions in a well-structured design
- To gain experience writing and calling functions to perform various tasks, including void and value-returning functions
- To understand and correctly use call by value and call by reference parameters
- To gain a deeper knowledge of programming more complex tasks in C++
- To utilize pseudo-random number generation to simulate a game of chance

Problem Statement

For this assignment, you will write a program which simulates a simplified version of the game *Blackjack*, or *21*. The game uses the following rules:

(1) There is one person who is the permanent dealer. For this simulation, assume there is only one other person in the game, the "player." At the start of every hand the player decides how many "chips" he wants to bet on that hand.

(2) The dealer then deals out 4 cards. The 1st card goes to the player, the 2nd to the dealer, the 3rd to the

player, and the 4th to the dealer.

(3) Each card is worth a certain number of points. Cards with numbers on them (2-10) are worth their face value. That is, a 3 is worth 3 points, a 9 worth 9 points, and so on. Jacks, Queens, and Kings are worth 10 points each. Aces are always worth 11 points. The value of a hand is simply the sum of the values of each card. Thus if you held a 6 and a Queen, your hand would be worth 16 points, A 4, a 5 and an Ace would be worth 20 points, and so on.

(4) The purpose of the game is to have a hand that is worth more points than your opponent's. However, you may NOT exceed a total of 21 points. If you do go over 21, you lose immediately (*go bust*).

(5) After the initial 4 cards have been dealt, it becomes the player's turn to decide if he is satisfied with his hand. If he believes he has enough points to win, he says "I stand". If he wishes, however, he may ask the dealer to give him an additional card by saying, "Hit me." His score is then the sum of all 3 cards. Again he has the option of standing or requesting still another card. The player may continue to request as many more cards as he wishes, until he is satisfied, or until his total becomes > 21, at which point he loses instantly.

(6) If the player stands before he exceeds 21, it then becomes the dealer's turn. The dealer MUST continue to draw cards until his total is greater than 16. (If his initial total is > 16, he stands at once). If the dealer draws a card that brings his total above 21, the player wins. If however the dealer stands with a number in the range 17-21, the winner is the person with the highest total. In the event of a tie, the bet is a draw and no chips are awarded.

In your program simulate a blackjack game for a total of 20 hands. Keep track of how many hands the dealer wins, how many hands the player wins, and how many hands are ties, and print out this information at the end of each hand. Also keep track of the total number of chips that the player has won or lost. The player determines the amount that is bet on each hand as follows: the player bets 1 chip on the very first hand. After that, the size of his bet is determined by how he did on the previous hand. If the player won the previous hand, he will bet 3 chips. If he tied the previous hand, he will bet 2 chips. If he lost the previous hand, he will bet only 1 chip.

The output of your program should provide a card-by-card description of the game. For example:

```
             *********************************
                    B L A C K J A C K
                    S I M U L A T O R
             *********************************

                        Hand 1
                        ------
The amount bet is: 1 chip(s)

The initial deal is:    Player               Dealer
                        ------               ------
                        2-spades             K-diamonds
                        Q-spades             2-clubs

The player has a total of 12, and decides to hit.
The player receives 6-hearts.

The player has a total of 18, and decides to stand.

    ------------------------------------------
```

```
The dealer has a total of 12, and must hit.
The dealer receives 2-hearts.

The dealer has a total of 14, and must hit.
The dealer receives 3-spades.

The dealer has a total of 17, and must stand.

The player wins this hand.

Overall scores:   Dealer = 0    Player = 1    Ties = 0
The player has won a total of: 1 chip(s)

==============================================================

                        Hand 2
                        ------
The amount bet is: 3 chip(s)

The initial deal is:    Player                Dealer
                        ------                ------
                        10-diamonds           9-hearts
                        K-clubs               7-hearts

The player has a total of 20, and decides to stand.

    ------------------------------------------

The dealer has a total of 16, and must hit.
The dealer receives 8-hearts.

The dealer has gone bust.

The player wins this hand.

Overall scores:   Dealer = 0    Player = 2    Ties = 0
The player has won a total of: 4 chip(s)
```

DEALING THE CARDS

To simulate dealing cards, you are required to use the following function to generate a pseudo-random integer, which will be in the range 1 through max.

Why do programmers sometimes use functions other than rand( )? Perhaps you will find yourself using a programming language which has no pseudo-random number generator built-in. In that case a portable function is needed. Or, you may be a stickler for accuracy and decide that the rand( ) function does not work well enough for you. In that case, you may wish to investigate one of the MANY alternatives available.

The following is a simple, basic, stand-alone function which you must use in your program 4:

```
int randmax
(int max)             /* largest possible number to return */
                      /* max should be restricted to the range */
                      /* 1 through 32767 for this function to */
                      /* work correctly */

/* This function computes a pseudo-random number in the range
   1 through max for max > 0, and returns it to the caller. The linear
   congruential algorithm is used. For more information on this
   algorithm, please see an appropriate textbook.

   Seed is used to control the sequence of numbers generated. A
   different initial value for seed would produce a different
   sequence of pseudo-random numbers.
*/

{   /* randmax */

    static int seed = 2;        /* determines sequence of random numbers */
                                /* initialized to 2 here to meet project */
                                /* requirements */

    seed = (seed * 13077 + 6925) % 32768;
    return (static_cast<int> (max * seed / 32768.0 + 1.0));

}   /* randmax */
```

To "deal" a card, simply use this function twice for each card, First, call randmax(13) to generate a number in the range 1...13, and use the convention that 1 -> Ace, 2 -> a two, ... , 10 -> a ten, 11 -> Jack, 12 -> Queen, and 13 -> King.

Next call randmax(4) to generate a number in the range 1...4, and use the convention that 1 -> spade, 2 -> heart, 3 -> diamond, and 4 -> club. Note that the suit of a card has no effect on its value. Note also that we assume the same card can be dealt more than once, with no limit on how many times it can be drawn. This is equivalent to assuming that the dealer has an "infinite" deck, which will simplify the program significantly.

Special note: do not change any of the numeric values in function randmax. Using this function as given will cause everyone to generate the same random number sequence, every time the program is run. This will simplify debugging for you, and also simplify program grading.

If you'd like to simulate dealing different card sequences, just change the seed to some other value while you are testing the program. However, when you turn in the program make sure the seed is set to 2.

Here is a good starting point if you are interested in more information about pseudo-random number generation:

http://www.nist.gov/dads/HTML/pseudorandomNumberGen.html

HOW THE PLAYER DECIDES WHEN TO HIT AND WHEN TO STAND

This is called a "strategy". In a real blackjack game the dealer deals the 2nd card to himself face up, and the player can see it. The player will base his decision to hit or stand based on the value of the dealer's 2nd card (i.e. the 4th card dealt in the hand). If the dealer's 2nd card has a value less than or equal to 6, the player should continue to hit as long as the player's total is less than 13. If the dealer's 2nd card has a value of 7 or higher, the player should continue to hit as long as the player's total is less than 17.

Input

This program takes no input.

Output

Your output must follow the course style guidelines and include a report of the events of the game as it is played, including at least all of the elements described earlier in this write-up. You may look at the sample output provided above for some ideas.

Use Of Functions

Part of your grade on this and **all** future course programming projects will be determined by how well you set up your program in a modular design and utilize functions and parameters appropriately.

Additional Info & Some Hints

To get started: do simpler tasks first, and get them to work before working on the harder tasks. Use the technique of writing, testing and debugging your code with stubs and drivers. Then, add on other tasks one at a time.

What File To Turn In and How to Turn In Your Work using Blackboard

You must turn in your C++ program source file, which must be entitled ***blackjack.cpp***.

Be completely certain you have ***thoroughly*** read the handout entitled ***"Submitting Your Program Assignments Electronically Using Blackboard"*** for complete instructions on how to submit and how to **verify your submission** after you submit it. This will help you to avoid losing large numbers of project points due to improper submission issues. This handout is available on the course web site under "Handouts."

Last Update: February 8, 2010 3:00 PM A. Ford Tyson
