**CSCI 1480   
University of Central Arkansas  
Lab 2 Out-of-Class Assignment**

**Chapter 9**

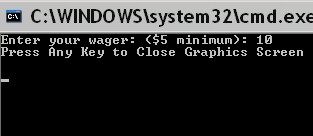
**Topics: Introduction to Pointers**

**Problem 1 (20 points) Name lab2out.cpp**

Write a C++ program that implements a basic 2-player card game. One player represents you while the 2nd player represents the dealer. First, your program will allow the user to enter a wager amount. The wager is always entered as an integer value and **must be greater than or equal to $5**. You will use the getWager function to accomplish this – see the required functions section:

The getWager function will perform all data validation

An input example the wager amount is **greater than or equal to $5** is shown below:



Once the user has entered a wager **greater than or equal to $5**, your program will perform the card game. Your game will utilize the following rules:

**Rules**

. The game is played with a standard deck of cards. There are 4 suits - hearts (♥), diamonds (♦), clubs (♣) and spaids (♠). There are 13 possible cards associated with each suit. These cards, and the amount of points associated with each card are shown below in Table 1:

**Table 1**

Card No Card Value Points File Names

2 2 2 h2.bmp,s2.bmp,d2.bmp,c2.bmp

3 3 3 h3.bmp,s3.bmp,d3.bmp,c3.bmp

4 4 4 h4.bmp,s4.bmp,d4.bmp,c4.bmp

5 5 5 h5.bmp,s5.bmp,d5.bmp,c5.bmp

6 6 6 h6.bmp,s6.bmp,d6.bmp,c6.bmp

7 7 7 h7.bmp,s7.bmp,d7.bmp,c7.bmp

8 8 8 h8.bmp,s8.bmp,d8.bmp,c8.bmp

9 9 9 h9.bmp,s9.bmp,d9.bmp,c9.bmp

10 10 10 h10.bmp,s10.bmp,d10.bmp,c10.bmp

11 Jack 10 h11.bmp,s11.bmp,d11.bmp,c11.bmp

12 Queen 10 h12.bmp,s12.bmp,d12.bmp,c12.bmp

13 King 10 h13.bmp,s13.bmp,d13.bmp,c13.bmp

14 Ace 11 h14.bmp,s14.bmp,d14.bmp,c14.bmp

**Note**: The ***Card No*** column is only included as a way to reference each card using an unique integer number

The cards are stored in 52 different bitmap files listed above. Each file has one of the following prefixes:

h - hearts

s - spaids

d - diamonds

c – clubs

The dimension for each card’s image is 71x100.

2. There will be only 2 participants in our game - 1 player (you) and the dealer.

:

3. The deck is divided exactly in half between you (the player) and the dealer. Your 26 cards are selected from the **hearts** and **diamonds** suits. The dealer’s cards are selected from the **spaids** and **clubs** suits.

4. Only 2 cards – one from **hearts** and one from **diamonds** – are randomly selected for you.

5. Only two cards – one from **spaids** and one from **clubs** – are randomly selected for the dealer.

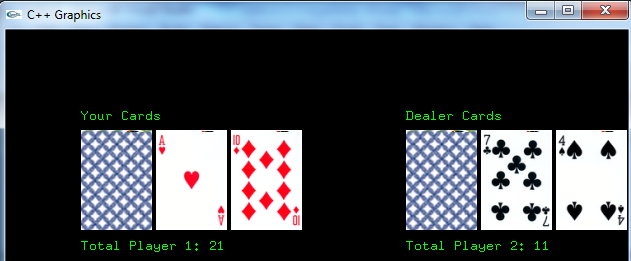
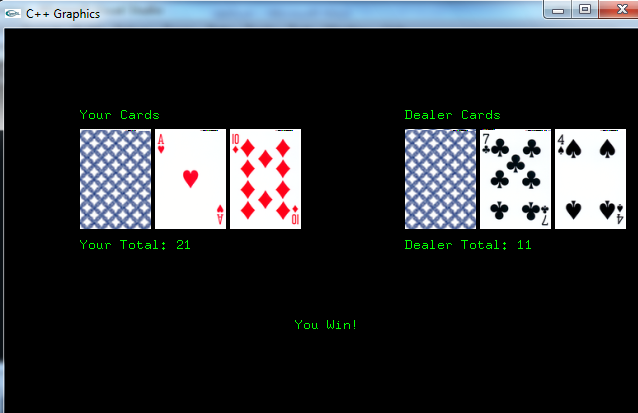
The rules 1 thru 5 are implemented in the deal function – See the Required Functions section for more details.

A score is computed based on the points associated for each card. This score is the addition of the corresponding value for each card. The card’s value is shown above in the table under the *card value* column. The score is computed by the computeScore function- See the **Required Functions** for more details.

These cards (including the leftmost *back.bmp*) should be displayed with the appropriate headings as shown below:

(400,90)

(400,100)



(75,90)

(75,100)

The score is also displayed under the cards as shown below:

Your score is compared with the dealers.

If your score is greater than the dealer’s, display “You Win!”

If your score is less than the dealer’s, display “You Loose!”

If your score is equal to the dealer’s, display “Tie!”

The display of the cards, the score, and the informational message (as described above) is performed by the displayResults function – See the **Required Functions** section for more details.

The suggested coordinates for this message is (290,300).

**Note**: You do not have to maintain the *accumulated* amount won (or lost) each time the game is played – simply display the information as shown above for each time the game is played.

**Required Functions**

The following functions are **required** and must be invoked from main **Do Not Change the Prototypes as given below.**

**However**, you are free to implement any **additional** functions you like**.**

void getWager(int\* wager);

Prompts the user for the desired wager and stores in the formal parameter wager (implemented as a pointer). Valid values are greater than or equal to 5.

void deal(int\* pc1\_no, int\* pc2\_no, int\* deal1\_no, int\* deal2\_no);

This function sets the two cards given to the player and the two cards given to the player. **All formal parameters are declared as pointers and are intended to be output parameters (i.e., the function will assign values to these parameters from within the deal function and the assigned values persist once the function completes ).** An explanation for the parameters are given below:

**pc1\_no** - A number randomly generated between 1 and 13 inclusively for the player that corresponds to the ***Card No*** column shown in the Table 1 above. This will be the card selected from the Hearts suit. This value represents the **first** card number dealt to the player.

**pc2\_no -** A number randomly generated between 1 and 13 inclusively that corresponds to the Card No column shown in Table 1 above. This card is selected from the Diamonds suit. This value represents the second card dealt to the player.

**deal1\_no -** A number randomly generated between 1 and 13 inclusively that corresponds to the ***Card No*** column shown in the table above. This card is selected from the Clubs suit. This value represents the **first** card number dealt to the dealer.

**deal2\_no -** A number randomly generated between 1 and 13 inclusively that corresponds to the ***Card No*** column shown in the table above. This card is selected from the Spaids suit. This value represents the **second** card number dealt to the dealer.

int computeScore(int card1\_no,int card2\_no);

The purpose of this function is to take 2 parameters that represent the card numbers (1-13) dealt to a participant and compute (and return) the appropriate score for that participant. For example, suppose **card1\_no** is assigned 1, and **card2\_no** is also assigned 1, the returned score would be 12 (this is beacause of 2 aces). As another example, suppose **card1\_no** is assigned 13, and **card2\_no** is assigned 12, the returned score would be 20

You will need to invoke this function twice in main – once to compute the score for the player, and once to compute the score of the dealer.

void displayResults(int pc1\_no, int pc2\_no, int deal1\_no, int deal2\_no, int player\_score, int dealer\_score);

Displays the cards, scores and informational messages as shown on page 3. The formal paramers pc1\_no, pc2\_no, deal1\_no, and deal2\_no are the same as described for the deal function. The formal parameter player\_score corresponds to the score computed for the player while dealer\_score corresponds to the score computed for the dealer (both by the computeScore function).

**Guidelines for main**

1. Declare variables wager, pc1\_no, pc2\_no, deal1\_no, deal2\_no as integers.
2. Invoke getWager function and pass the address of the wager variable (&wager)
3. Invoke deal function and pass the address of the pc1\_no, pc2\_no, deal1\_no, deal2\_no variable using the & operator (example: &pc1\_no)
4. Invoke computeScore function for players cards.
5. Invoke computeScore function for dealers cards
6. Invoke displayResults function and display the cards for the player and dealer, the score for player and dealer, and the informational message about who won the game.

**Have the program repeat as many times as desired.**

**Submit lab2out.cpp via Blackboard.**

\