CS 103 Lab - TwentyOne

1 Introduction

In this assignment you will use write program to implement the basic rules of the card game, Twenty-One (aka Blackjack). This program will require you to utilize the following knowledge and skills:

- for and while loops
- conditionals
- functions
- arrays and passing arguments

2 Background

The game of Twenty-One is a popular card game that can have many players competing against a dealer. In our version, we will only allow ONE player vs. the dealer. While this game usually contains betting, we will leave out that aspect and just focus on implementing the rules of play and determining the winner.

3 The Rules

3.1 The Goal

The goal of the card game is for the player's cards to total more than the dealer's without going over a total of 21 (aka a "bust"). Non face-cards have the value shown on the card (i.e. 2-10) while Jacks, Queens, and Kings have a value of 10. Aces are special in that they have a value of 11 by default unless that value would cause the player to bust (i.e. have a total greater than 21) in which case the Ace may count as just 1.

3.2 Setup

In this implementation we will only using 1 deck of (52) playing cards. The deck will be shuffled and then two cards will be dealed to the player and dealer in an alternating fashion (i.e. first one card to the player then to the dealer, then a second card). While both of the player's cards are dealt face up (visible), the dealer's first card is kept hidden, while their second is visible. The player uses this information to help inform their choices.

3.3 The Player

Gameplay starts with the player who must decide whether to "stand" (not ask for another card) or "hit" (ask for another card in an attempt to get closer to a count of 21, or even hit 21 exactly). Thus, a player may stand on the two cards originally dealt him/her, or he/she may ask the dealer for

additional cards, one at a time, until the player either decides to stand on the total (if it is 21 or under), or goes "bust" (if it is over 21). In the latter case, the player loses immediately and the game is OVER (i.e. the dealer need not play or take any cards).

3.4 Handling Aces

In 21, ace cards can be counted as EITHER 11 or 1. The combination of an ace with another card is known as a "soft hand," because the player can count the ace as a 1 or 11, and either draw cards or not. For example with a "soft 17" (e.g. an ace and a 6), the total is 7 or 17. While a count of 17 is a good hand, the player may wish to draw for a higher total. If the player draws a card that causes a bust by counting the ace as an 11, the player is allowed to count the ace as a 1 instead and continue playing by standing or hitting (asking the dealer for additional cards, one at a time). Going back to our example, if the player hits and receives a 5 (e.g. their hand contains an ace, 6, and 5), the player can now let their Ace be a 1 for a total of 12 and continue gameplay. If the player were to hit and receive a second ace, then it too can count as a 1. Thus, any ACE can be counted as either 11 or 1, whichever helps make the sum total of the player's hand closer to 21 without going over. In your code, you should count an ACE as 11 unless doing so causes the sum to be over 21 in which case you can recalculate the sum, making the ACE worth 1. Again, if there are 2 or more aces in a hand, one of them can count as 11 while the others can be counted as 1 each.

3.5 The Dealer

If the player did not "bust" the dealer now plays. His/her face-down card is turned up. If the total is 17 or more, the dealer must stand. If the total is 16 or under, the dealer must take a card. He/she must continue to take cards until the total is 17 or more, at which point the dealer must stand. If the dealer has an ace, and counting it as 11 would bring his total to 17 or more (but not over 21), he must count the ace as 11 and stand. The dealer's decisions, then, are automatic on all plays, whereas the player always has the option of taking one or more cards.

4 Representation

To represent the 52 cards of a deck we will just use integers in the range 0 to 51 (which we can store in an array). Each card will be assigned an ID from 0 - 51 in the following order: 0-12 = 2 -> A of hearts, 13-25 = 2 -> A spades, 26 - 38 = 2 -> A diamonds, 39 - 51 = 2 -> A of clubs. This is more explicitly shown in the table below. (Note: H = Hearts, S = Spades, D = Diamonds, C = Clubs)

Card	Integer ID	Card	Integer ID	Card	Integer ID	Card	Integer ID
2-H	0	2-S	13	2-D	26	2-C	39
3-H	1	3-S	14	3-D	27	3-C	40
4-H	2	4-S	15	4-D	28	4-C	41
5-H	3	5-S	16	5-D	29	5-C	42
6-H	4	6-S	17	6-D	30	6-C	43
7-H	5	7-S	18	7-D	31	7-C	44
8-H	6	8-S	19	8-D	32	8-C	45
9-H	7	9-S	20	9-D	33	9-C	46
10-H	8	10-S	21	10-D	34	10-C	47
J-H	9	J-S	22	J-D	35	J-C	48
Q-H	10	Q-S	23	Q-D	36	Q-C	49
K-H	11	K-S	24	K-D	37	K-C	50
A-H	12	A-S	25	A-D	38	A-C	51

5 Requirements

1. **Shuffling**: *For each new hand*, you should initialize the deck of 52 cards to have values 0-51 in that order (i.e. cards[i] = i;) and then shuffle that deck using the Fisher-Yates / Durstenfeld shuffle algorithm (shown below) so that we all get the same card ordering:

```
// To shuffle an array a of n elements (indices 0..n-1):
for i from n−1 downto 1 do
     j ← random integer such that 0 ≤ j ≤ i
         (i.e. use a modulus operation on the random number)
     exchange a[j] and a[i]
```

   As a check to ensure you are shuffling correctly, if you were to seed the random number generator with a value of 37 (which is the seed used in the first sample run shown below) and then printed out the contents of the deck after shuffling (for debug purposes), then they should be:

```
6 14 2 39 24 23 32 4 12 49 27 36 21 42 10 8 38 51 46 47 22 5 37 41 16 15
43 40 3 31 44 34 17 18 9 26 1 35 19 11 30 48 13 20 25 33 29 28 0 45 50 7
```

2. Initially, the dealer's first card should not be shown to the user. Instead display a ?.
3. The player should be presented with the choice to type h to hit or s to stay. However, if the player has 21 from the initial 2 cards of their hand, the dealer should immediately start playing (i.e. no choice should be given to the player to hold or stay).**Also, any character other than h or s should cause the program to immediately quit.**

4. You must generate the following output based on the results of the game:
    - Output Player busts on a separate line if the player goes over 21 *OR* output Dealer busts on a separate line if the dealer goes over 21.
    - Also, output Win, Tie, or Lose indicating if the player won, tied, or lost to the dealer, respectively, followed the player score and the dealer score separate by a space (all on one line).
5. When the hand ends the player should be allowed to play again (without restarting the program or re-seeding the random number generator) by typing y to play another hand or n to quit. **If the user types any character other than y the program should immediately quit.**
6. Playing another hand should cause the deck to be reinitialized to 0-51 (i.e. cards[i] = i;) and then shuffled (but do **NOT re-seed** the random number generator...just start again by calling shuffle().)
7. If the player busts, the dealer does not play and the desired output should be displayed.

6 Additional Background

1. **Command line arguments**: One other way we can provide input to our programs other than cin is via command line arguments. In this method, we provide input at the command line when we start our program (the information comes after the name of the executable). Then our program can access that info and use it. We will use this technique for the seed value to use. So now you will start your program by typing a seed value after the name of the executable. For example:

    ```
    $./twentyone 37
    ```

    or

    ```
    $./twentyone 20134
    ```

    In the first example the program will use 37 as its seed and in the second program 20134 will be used as the seed. We will learn more about how to use these command line arguments shortly in our class, but for now we have provided all the code to deal with them in the skeleton file. You need not change/adjust that code. Simply realize you must type in the seed value at the command line when you start the program.

2. **Using arrays to lookup values**: One application of arrays is to pre-define values your program may want to use so you don't have to hard code a bunch of if statements but instead just look up the desired value from an array. We have provided some arrays in our skeleton that can be used to easily determine the

value, type, and suit of a card. **You do not have to use these arrays, though they will make your life easier if you do.** To use them, you must generate the appropriate index to the array and then you can retrieve the value at that index. For example,

```
const char* type[13] =
  {"2","3","4","5","6","7","8","9","10","J","Q","K","A"};
```

can be used to print out the value of a card by writing:

```
  cout << type[i];
```

Your job would be to produce i appropriately (i.e. i will need to be 0 to print out 2, i will need to be 1 to print our 3, ..., and i will need to be 12 to print out A). Since we use the values 0-51 for card values you can do some math on that value to convert it to the appropriate index needed for the array.

7 Sample Runs, Input, and Output

Here are five sample runs of the program: the emphasized text is the part that the user typed in and the $ is the command-line prompt.
Most output is up to you to decide how you want to phrase and display. However you must:

1. **Show the cards in the proper format (e.g. 9-H, J-C, A-D**
2. **Show the Player busts, Dealer busts, and Win/Tie/Lose player-score dealer-score messages**. These outputs are shown in italics on the sample runs below.

Sample 1: Player busts twice.

```
$./twentyone 37
Dealer: ? 2-C
Player:  8-H 4-H
Type 'h' to hit and 's' to stay:
h
Player: 8-H 4-H K-S
Player busts
Lose 22 5

Play again? [y/n]
y
Dealer: ? J-H
Player: 8-S 3-D
Type 'h' to hit and 's' to stay:
h
Player: 8-S 3-D A-S
Type 'h' to hit and 's' to stay:
```

```
h
Player: 8-S 3-D A-S Q-C
Player busts
Lose 22 20

Play again? [y/n]
n
```

Sample 2: Player wins and then loses based on score.

```
$./twentyone 29
Dealer: ? 9-C
Player: K-D J-D
Type 'h' to hit and 's' to stay:
s
Dealer: Q-H 9-C
Win 20 19

Play again? [y/n]
y
Dealer: ? 10-C
Player: 6-C 6-H
Type 'h' to hit and 's' to stay:
h
Player: 6-C 6-H 4-H
Type 'h' to hit and 's' to stay:
s
Dealer: J-H 10-C
Lose 16 20

Play again? [y/n]
n
```

Sample 3: Dealer busts.

```
$./twentyone 1411
Dealer: ? J-S
Player: 3-D 4-H
Type 'h' to hit and 's' to stay:
h
Player: 3-D 4-H K-H
Type 'h' to hit and 's' to stay:
s
Dealer: 3-C J-S 2-D A-C 9-D
Dealer busts
Win 17 25

Play again? [y/n]
n
```

Sample 4: Player doesn't need to choose to hit or stay since they already have 21. Also, dealer stays at 17 or more even though it means it will lose.

```
$./twentyone 20132
```

```
Dealer: ? 3-D
Player: 10-S A-S
Dealer: 2-H 3-D 3-S 9-D
Win 21 17

Play again? [y/n]
n
```

Sample 5: Tie.

```
$./twentyone 4
Dealer: ? 9-C
Player: 7-D 5-C
Type 'h' to hit and 's' to stay:
h
Player: 7-D 5-C 7-C
Type 'h' to hit and 's' to stay:
s
Dealer: 4-C 9-C 6-S
Tie 19 19

Play again? [y/n]
n
```

Sample 6: Player no longer needs to choose if they reach 21 after hitting.

```
$./twentyone 7
Dealer: ? 3-C
Player: A-H 3-D
Type 'h' to hit and 's' to stay:
h
Player: A-H 3-D 7-D
Dealer: 4-S 3-C 10-H
Win 21 17

Play again? [y/n]
n
```

Sample 7: Aces are counted correctly to give best score without busting. In this run we stay and get a score of 18.

```
$./twentyone 11
Dealer: ? 10-D
Player: 7-D A-S
Type 'h' to hit and 's' to stay:
s
Dealer: 2-S 10-D 7-H
Lose 18 19

Play again? [y/n]
n
```

Sample 8: In this run we choose to hit and thus the Aces needs to be counted as 1.

```
$./twentyone 11
Dealer: ? 10-D
Player: 7-D A-S
Type 'h' to hit and 's' to stay:
h
Player: 7-D A-S 7-H
Type 'h' to hit and 's' to stay:
s
Dealer: 2-S 10-D 3-S J-D
Dealer busts
Win 15 25

Play again? [y/n]
n
```

8 Q&A

Q: Do we have to worry about users typing in characters other than h or s to hit or stay?

A: No. You may assume that only h or s will be entered, not H, S, or any other character.

Q: Do we have to worry about users typing in characters other than y or n when we ask if they want to play again?

A: No. You may assume that only y or n will be entered, not Y, N, or any other character. **However, your program should quit if ANY character other than 'y' is entered**. If you submit your program and our checker returns a timeout error, it is likely because you have failed to meet this requirement.

Q: How can I use printHand to print the dealer's initial cards since I need to display a ? for their first card?

A: Don't use printHand() for that task. Remember after printing out the dealer's hand initially with the first card hidden, subsequent display of the dealer's hand should show all the cards, including the first. For that you can use printHand()

9 Skeleton:

We have provided a skeleton (starter code) in Vocareum.

10 Readme:

Answer the experimental questions that are listed in the readme file.
Open “readme.txt” in Vocareum to read the questions and answer them.
Please write your answers in the same file “readme.txt”

11 Style:

You must follow all of the code style guidelines posted here.
http://bytes.usc.edu/cs103/coursework/style/

12 Submit:

Use Vocareum to submit your code and readme file. It will run some basic tests to check your formatting. To check it more exhaustively, it would be a good idea for you to try running it on additional test cases.