UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

School of Computer Science & Applied Mathematics

Introduction to Algorithms & Programming (COMS1018A/1022A) Assignment 2

Due Date: 8 June 2020 23h59

1 Introduction

Like the labs, assignments are marked automatically online. This means that your program must produce *exactly* the expected output for it to be considered correct. For each task, submit the appropriate `.cpp` file to the Moodle marker. Successfully completing all five tasks will earn you 100%.

Again, please be aware of the policy surrounding plagiarism (see the course outline for a full description). Code will be scrutinised, and anyone found to have cheated (including the person from whom the code was copied) will immediately receive 0, and may be subject to disciplinary sanctions.

2 Background

As mathematicians, we all know that the odds in any gambling game are stacked against us. No casino would ever offer players a game where they are able to come out ahead in expectation. Gambling should thus be considered a form of entertainment, where players are under no illusion that they will most likely lose money in the long run. In my opinion, there are better things to spend your money on. Like this 53-inch Teddy Bear (`http://costcocouple.com/53-inch-plush-teddy-bear/`). But that's just me.

Your friend[1] Kiernan, however, is convinced otherwise. He wants to visit his local casino and win money by playing *Blackjack*. Before visiting the casino, he'd like to get some practice in, and has asked you to write a program that will allow him to play hands of Blackjack on his computer.

3 The Plan

Kiernan is ... well ... let's just say he can't count to twenty with his shoes on. You therefore decide to implement a simplified version[2] of Blackjack, the rules of which are described below.

[1]More of an acquaintance, really.

[2]The rules of this version mirror standard Blackjack, but the player only has two actions available to him: hit or stand.

The Rules

Setup

The game uses a single deck of standard playing cards, and is contested between two people: the *dealer* and the *player*. At the start of each round, the player receives two cards, while the dealer receives a single card. The object of the game is to beat the dealer by having a hand of cards with a value greater than the dealer's, but not more that 21.

The Cards

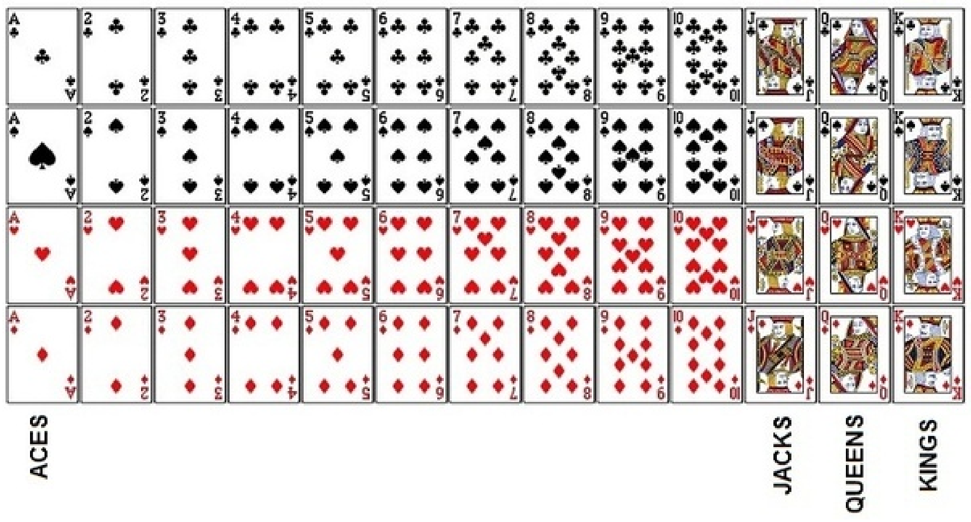

Figure 1: Standard deck of cards. The first row are clubs, the second spades, the third hearts and the fourth diamonds.

The game is played with a standard deck of cards. There are 52 cards in total, divided into 4 *suits*: clubs, spades, hearts and diamonds. Each suit contains thirteen cards: ace, king, queen, jack, and the cards 2 — 10.

Scoring

The value of a person's hand is calculated as the sum of the individual cards: aces count as either 1 or 11, tens and face cards (kings, queens and jacks) are worth 10 points, while the remaining cards are scored according to their face value.

Scoring is slightly more involved when a hand contains an ace. In this case there are two possible values, since an ace can be treated as either 1 or 11. If both possible values do not exceed 21, then the hand is said to contain a *usable* ace. If the larger of the values is more than 21, then only the lower value is considered. For example, a hand consisting of an ace and a six has a value of either 7 or 17, while a hand consisting of an ace, a six and a ten has a value of 17 only.

A special case occurs when the first two cards of either contestant add up to 21. In such a case, the person is said to have *Blackjack* and immediately win the round. A hand with more than two cards that add up to 21 is **not** *Blackjack*.

Play

At the beginning of the round, the player's cards are examined. If they add up to 21, the player has *Blackjack* and immediately wins the round. Otherwise the player is given two options: to request another card (*hit*) or to stick with his current hand (*stand*). The player keeps making decisions until he chooses to stand, or the value of his hand exceeds 21. If his hand exceeds 21, he is said to have *busted*, and the dealer wins the round.

If the player completes his turn without busting, the dealer then takes his turn. The dealer first draws a card. If this results in the dealer achieving *Blackjack*, then he immediately wins the round. Otherwise, the dealer keeps drawing cards until the value of his hand is greater than 16, at which point he stops drawing cards. If the dealer has a usable ace, then he stops if his higher score is greater than 16.

If the dealer busts, or has a value less than that of the player, then the player wins. If the dealer has a higher value, the player loses. And if both have the same value, they tie (*push*).

4 The Plan (continued)

Like any good programmer, you're not going to write this as one big system, so you break it down into a series of manageable tasks. For each of the tasks, you plan to represent a card as a string in the form `<rank><suit>`. `<rank>` is either `2-10`, `J` (jack), `Q` (queen), `K` (king), or `A` (ace), while `<suit>` is either `c` (clubs), `d` (diamonds), `h` (hearts), or `s` (spades).

For example, the "ten of hearts" is represented by `10h`, while the "ace of clubs" is `Ac`.

5 The First Task (25 marks)

Write a program that takes in a single card and outputs its value according to the rules of Blackjack. For aces, the program should output both possible values.

5.1 Input

A single line representing a card, in the form described by Section 4.

5.2 Output

The value (or values) of the card.

5.3 Example Input-Output Pairs

Sample Input #1

```
2d
```

Sample Output #1

```
2
```

Sample Input #2

```
As
```

Sample Output #2

```
1 or 11
```

Sample Input #3

```
Qh
```

Sample Output #3

```
10
```

6 The Second Task (30 marks)

Now write a program that calculates the value of two cards. If the hand's value is 21, then the program should output `Blackjack!` Otherwise, the value of the hand should be displayed. If the hand contains a usable ace, both possible values must be output.

6.1 Input

Input consists of two cards on individual lines.

6.2 Output

The value (or values) of the cards, or `Blackjack!`.

6.3 Example Input-Output Pairs

Sample Input #1

```
Ah
Kd
```

Sample Output #1

```
Blackjack!
```

Sample Input #2

```
As
Ac
```

Sample Output #2

```
2 or 12
```

Sample Input #3

```
Ac
4d
```

Sample Output #3

```
5 or 15
```

7 The Third Task (20 marks)

Because the program will be used for practising, a useful feature would be to suggest an action to the player. Write a program that accepts the dealer's single card, and the player's two initial cards, and outputs a suggested action according to the tables below.

If the player has a usable ace

Player	**Dealer's Card**									
	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**A**
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
17	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit
19+	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

If the player does not have a usable ace

Player	**Dealer's Card**									
	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**A**
4-8	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
9	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
10	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
14	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
17+	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

7.1 Input

Input consists of three cards, each on their own line. The first card belongs to the dealer, while the remaining two belong to the player.

7.2 Output

Output either `Hit` or `Stand`, according to the above strategy

7.3 Example Input-Output Pairs

Sample Input #1

```
4d
Ac
As
```

Sample Output #1

```
Hit
```

Sample Input #2

```
Ac
10s
10h
```

Sample Output #2

```
Stand
```

Sample Input #3

```
2d
Ah
6c
```

Sample Output #3

```
Hit
```

Sample Input #4

```
8d
7d
7h
```

Sample Output #4

```
Hit
```

8 The Fourth Task (15 marks)

Naturally, a hand may contain any number of cards. Write a program that takes in a hand consisting of any number of cards and outputs the hand's score, as per The Second Task. If the hand consists of two cards and equals 21, `Blackjack!` should be printed. Otherwise, if the hand's value exceeds 21, the word `Bust!` should be displayed.

8.1 Input

Input consists of of a series of cards, one per line. Input is terminated by a single line containing the word `end`.

8.2 Output

Output either the value(s) of the hand, `Blackjack!` or `Bust!` where appropriate.

8.3 Example Input-Output Pairs

Sample Input #1

```
Ah
Kd
end
```

Sample Output #1

```
Blackjack!
```

Sample Input #2

```
10d
10h
Ac
end
```

Sample Output #2

```
21
```

Sample Input #3

```
Ac
6h
10d
end
```

Sample Output #3

```
17
```

Sample Input #4

```
Ac
6h
10d
5d
end
```

Sample Output #4

```
Bust!
```

9 The Fifth Task (10 marks)

Having completed the previous tasks, creating a Blackjack game is simply a case of putting it all together. Download the source code on Moodle and complete the require functions to create a basic, but fully-working Blackjack game. You may add as many additional functions as required, but do not modify or delete the existing code in any way. There are five functions that need to be implemented to complete the program:

9.0.1 `string outputScore(const vector<string> &hand);`

This function takes in a hand of any size and returns the valuation of the hand, as per The Fourth Task.

9.0.2 `bool isBlackjack(const vector<string> &hand);`

This function takes in a hand of any size and returns whether the hand is a *Blackjack* hand.

9.0.3 `bool isBust(const vector<string> &hand);`

This function takes in a hand of any size and returns whether it is a *bust* hand.

9.0.4 `string getAdvice(const vector<string> &playerHand, string dealerCard);`

This function takes in a hand of any size and the dealer's card, and returns what action should be taken, as per The Third Task.

9.0.5 `int getHighScore(const vector<string> &hand);`

This function takes in a hand of any size and returns its numeric score. If the hand contains a usable ace, then this function returns the larger of its two possible values.

9.1 Input

Input consists of the user responses to the on-screen prompts. For example, when the program starts, the following is displayed

```
(N)ew round or (Q)uit?
```

Input would then be either the character `N` or `Q`. There is no need to worry about handling input — the existing code takes care of it for you.

9.2 Output

Output is handled by the existing program. Ensure your functions do not print anything to the screen, and do not modify existing output in any way.

9.3 Sample Run

Run the program and interact with it. A sample of what it might look like is reproduced below. User input is indicated by red text.

```
(N)ew round or (Q)uit? N
Dealer shows 9s  -> 9
Player shows 3s 4h  -> 7
(H)it, (S)tand, or (A)dvice? H
Player shows 3s 4h 10d  -> 17
(H)it, (S)tand, or (A)dvice? S
Dealer shows 9s Js  -> 19
Dealer wins!
******************************************
(N)ew round or (Q)uit? N
Dealer shows 7s  -> 7
Player shows 9s Ks  -> 19
(H)it, (S)tand, or (A)dvice? S
Dealer shows 7s Qc  -> 17
Player wins!
******************************************
(N)ew round or (Q)uit? N
Dealer shows Ks  -> 10
Player shows 10s 4d  -> 14
(H)it, (S)tand, or (A)dvice? A
Advice: Hit
(H)it, (S)tand, or (A)dvice? H
Player shows 10s 4d 6c  -> 20
(H)it, (S)tand, or (A)dvice? S
Dealer shows Ks 5d  -> 15
Dealer shows Ks 5d 9h  -> Bust!
Player wins!
******************************************
(N)ew round or (Q)uit? Q
```