DEMO (STRAIGHTS)

Command line parameters

To run the executable, there will be 4 formats:

1. No seed for shuffling and no input file for testing:
   ./straights

2. With seed, but no input file for testing:
   ./straights (seed)

3. No seed for shuffling, but with input file for testing:
   ./straights -f (filename)

4. With seed and input file for testing:
   ./straights (seed) -f (filename)

Note:

- If there will be more than 4 arguments, the program will exit immediately.
- If (seed) is invalid or out of range (i.e. an error occurs), the seed will be time based.
- If there is no (filename) or if there is but an error occurs while trying to open (filename), std::cin will be used for input.

When EOF is reached, the program will terminate.

Invite players

Firstly, when the program runs, it asks player 1-4:

Is Player<x> a human (h) or a computer (c)?

>

If human, type “h”, and if computer, type “c”. If another input other than “h” or “c” is given, the program will repeatedly ask until a valid input is given.

We can see this by running:

./straights

```
Is Player<1> a human (h) or a computer (c)?
>h
Is Player<2> a human (h) or a computer (c)?
>hh
Is Player<2> a human (h) or a computer (c)?
>cc
Is Player<2> a human (h) or a computer (c)?
>c
Is Player<3> a human (h) or a computer (c)?
>
```

Shuffling and Dealing

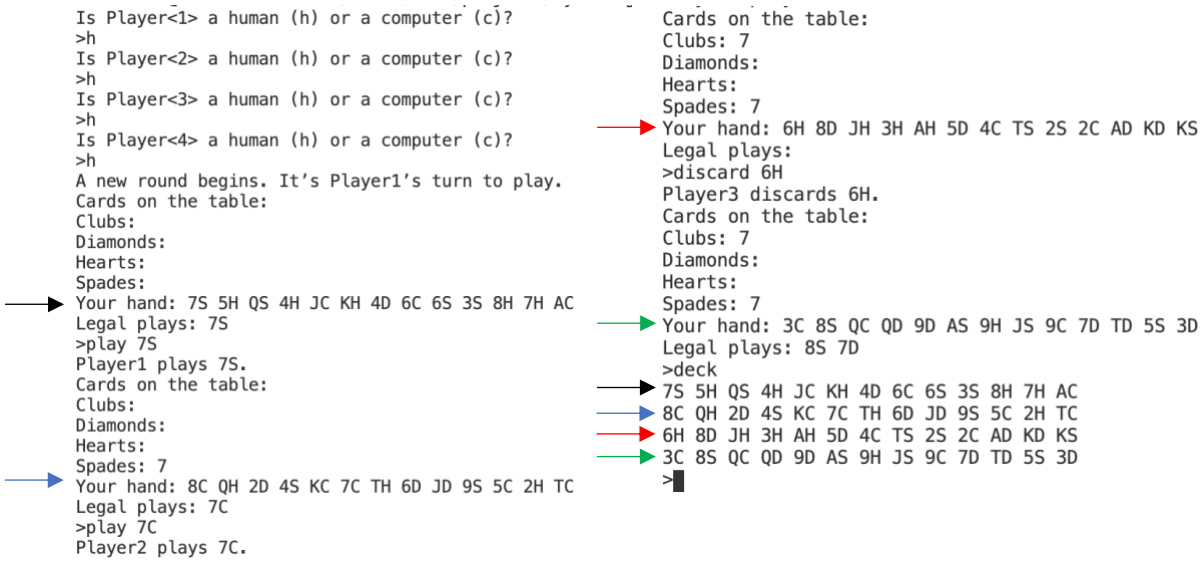
The starting deck is in the following order:

AC 2C 3C ... TC JC QC KC AD 2D ... QD KD AH 2H ... QH KH AS 2S ... QS KS

We can see how the starting deck is made in this order in main.cc : 67.

At the beginning of every round, the deck is shuffled once according to the seed. After the shuffle, the cards are dealt such that the first 13 cards belong to Player1, the next 13 belong to Player2, the next 13 belong to Player3, and the last 13 belong to Player4 (See below, from the hand of each of the players and the deck of that round).

All of these shuffling and dealing are done in the function GameController::shuffleDeal(). When the round is over, the deck is restored to its previous state before shuffling it.



This is done by running:

./straights 2

with inputs:

h

h

h

h

play 7S

play 7C

discard 6H

deck

Start

When a new round begins, the program will output (can be seen in the demo below):

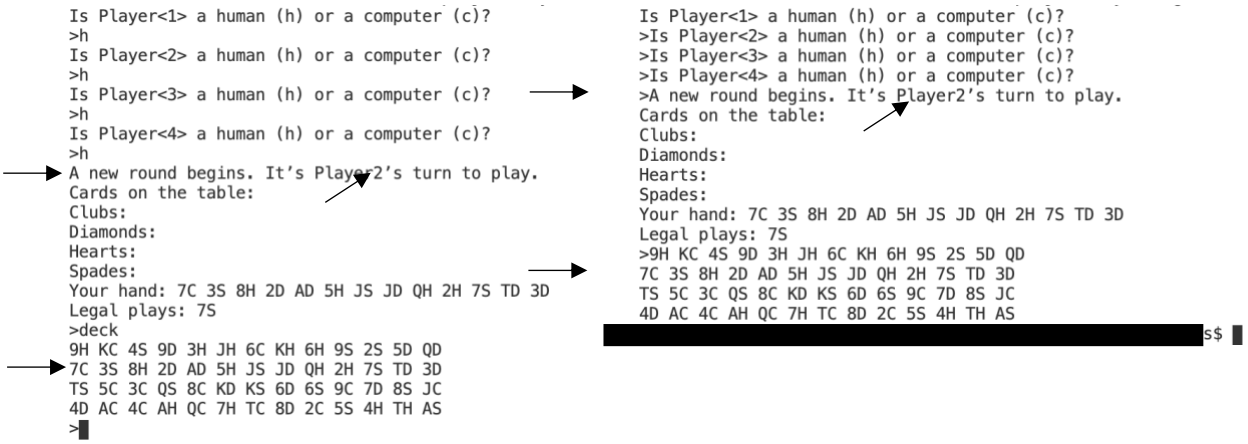
A new round begins. It's Player<i>'s turn to play.

i will always be the person who has 7S in their hand and its legal play will only be 7S.

If we run:

./straights 3 -f demo1.txt

We can see that the 2nd line of the deck is the only one that consist of 7S and thus, the 2nd player is the one that begins the round. Also, its legal plays only consist of only 7S, since it's the first turn of the first player of the round.



Note:

The left one is done by running ./straights 3 and the input is typed one by one while the right one is done by running ./straights 3 -f demo1.txt. Thus, we can't see the h h h h deck there and the output will be different since the inputs are taken from the demo1.txt.

Human Player

1.

Whenever it's a human player's turn, the program will print:

Cards on the table:

Clubs:<list of clubs>

Diamonds:<list of diamonds>

Hearts:<list of hearts>

Spades:<list of spades>

Your hand:<cards in your hand>

Legal plays:<legal plays in your hand>

>

This can be seen when running:

./straights 3 -f demo2.txt

```
Cards on the table:
Clubs:
Diamonds:
Hearts:
Spades:
Your hand: 7C 3S 8H 2D AD 5H JS JD QH 2H 7S TD 3D
Legal plays: 7S
```

The legal plays are the cards that are adjacent with the cards on the table (this rule applies to both human and computer players)

For example if we **run:**
./straights 5
with inputs:

c

h

c

c

play 7S

discard KH

play 4S

we'll get:

```
Cards on the table:
Clubs: 5 6 7
Diamonds:
Hearts: 6 7
Spades: 4 5 6 7 8
Your hand: 3D JS AC JD KS 5H 2D 4H 3S 8H
Legal plays: 5H 3S 8H
>
```

In this case, players are only able to play 4C, 8C, 7D, 5H, 8H, 3S & 9S, since they are adjacent to the cards on the table. From all of these 7 cards, Player2 currently has only 3 of these cards, which are its legal plays, 5H, 3S & 8H. Hence, Player2 can only play either of these cards for this current turn.

Continuing the inputs on top with:

play 5H

play 4H

play AC

we'll get:

```
Cards on the table:
Clubs: A 2 3 4 5 6 7
Diamonds: 6 7
Hearts: A 2 3 4 5 6 7
Spades: 4 5 6 7 8 9
Your hand: 3D JS JD KS 2D 3S 8H
Legal plays: 3S 8H
>
```

Now, in this case, players are only able to play 8C, 5D, 8D, 8H, 3S & TS, since there's no card adjacent to AC and AH (all aces and kings). From all of these 6 cards, Player2 currently has only 2 of these cards, which are its legal plays, 3S & 8H. Hence, Player2 can only play either of these cards for this current turn.

2.

<cards in your hand> and <legal plays in your hand> are arranged in the same order that they appeared in the deck.

This can be seen when running:

./straights 3 -f demo3.txt

```
Cards on the table:
Clubs: 7 8
Diamonds:
Hearts: 6 7
Spades: 5 6 7
Your hand: 9H KC 4S 9D 3H JH 6C KH 9S 2S 5D QD
Legal plays: 4S 6C
>9H KC 4S 9D 3H JH 6C KH 6H 9S 2S 5D QD
7C 3S 8H 2D AD 5H JS JD QH 2H 7S TD 3D
TS 5C 3C QS 8C KD KS 6D 6S 9C 7D 8S JC
4D AC 4C AH QC 7H TC 8D 2C 5S 4H TH AS
```

The picture above shows Player1's hand and legal plays that are arranged in the same order that they appeared in the first line of the deck.

3.
Every list of cards, except for that produced by the deck command, starts with a space character, and a single space separates each card from the next. There is no space after the final card in the list.

This can be seen from:

gamecontroller.cc:23, GameController::printStartDeck()

table.cc:54, Table::printTable()

player.cc:74, Player::printHand()

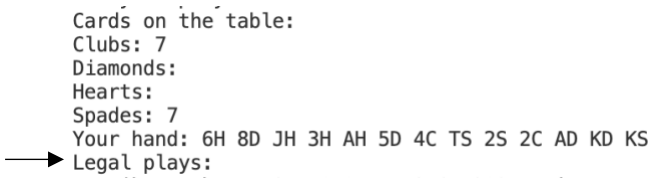
player.cc:142, Player::printLegalPlays()

player.cc:101, Player::printDiscard()

If there are no legal plays, then the list of cards consists of an empty string
i.e. print Legal plays:\n.

This can be seen when running:

./straights 2 -f demo4.txt



Commands

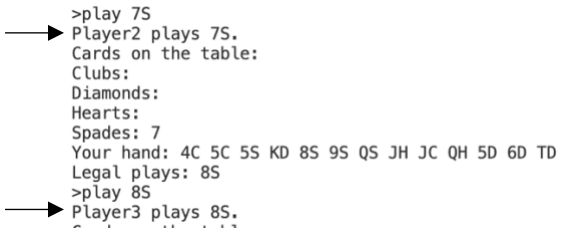
- play <card>

  If the play is legal, the program will print:
  Player<x> plays <card>.

  And then, the next player will play.

  These can be seen when running:

  ./straights 5 -f demo5.txt



In the picture above, after Player2 plays 7S, which is a legal play, it's Player3's turn to play.

If the card played is not a legal play, the program will print:

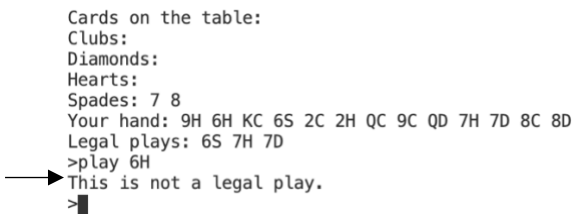
This is not a legal play.

>

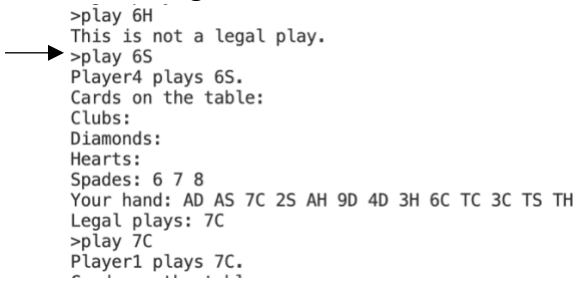
And it will not proceed to the next player until a legal play is made

This can be seen when running

./straights 5 -f demo6.txt



Continuing…



Note:

The pictures above are taken when running ./straights 5 with inputs of demo6.txt inputted one by one to show the outputs more clearly.

- discard <card>

When a player has no legal play, it must discard a card and the program will print:

Player<x> discards <card>.

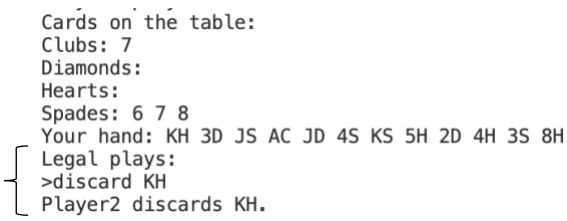
If the player has a legal play but tries to discard a card, the program will print:

You have a legal play. You may not discard.

>

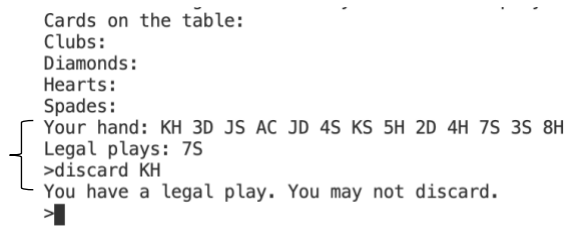
The first one can be seen when running:

./straights 5 -f demo7.txt



The second case can be seen when running:

./straights 5 -f demo8.txt



Note:

The pictures above are taken when running ./straights 5 with inputs of demo7.txt and demo8.txt respectively inputted one by one to show the outputs more clearly.

- deck

The decks of that current round are printed in order, 13 card per line.

This can be seen when running:

./straights 5 -f demo9.txt

```
>deck
AD AS 7C 2S AH 9D 4D 3H 6C TC 3C TS TH
KH 3D JS AC JD 4S KS 5H 2D 4H 7S 3S 8H
4C 5C 5S KD 8S 9S QS JH JC QH 5D 6D TD
9H 6H KC 6S 2C 2H QC 9C QD 7H 7D 8C 8D
>
```

Note:

The pictures above are taken when running ./straights 5 with inputs of demo9.txt inputted one by one to show the outputs more clearly.

- quit

Terminates the program immediately

This can be seen when running:

./straights

with inputs:

h

h

h

h

quit

```
Is Player<1> a human (h) or a computer (c)?
>h
Is Player<2> a human (h) or a computer (c)?
>h
Is Player<3> a human (h) or a computer (c)?
>h
Is Player<4> a human (h) or a computer (c)?
>h
A new round begins. It's Player4's turn to play.
Cards on the table:
Clubs:
Diamonds:
Hearts:
Spades:
Your hand: AH QS KC 8C KD 5H 9C 8H 6S 2S 2H 8S 7S
Legal plays: 7S
>quit
[redacted]:~/cs246/1209/projects/myStraights$
```

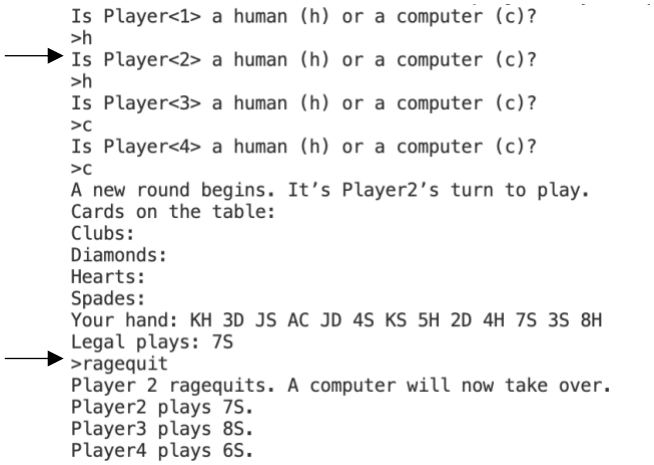
- ragequit

  Replace the current human player with a computer player, and resume the game. When this happens, the program will print:

  Player <x> ragequits. A computer will now take over.

  This can be seen when running:

  ./straights 5 -f. demo10.txt



From the picture, we can see that when Player2 (human) ragequits, a computer directly takes over and play.

Note:

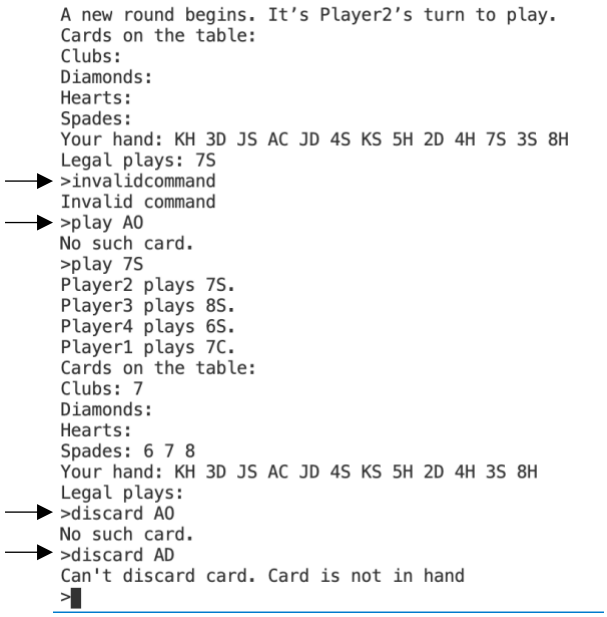
The pictures above are taken when running ./straights 5 with inputs of demo10.txt inputted one by one to show the outputs more clearly.

Handling of Commands

If a player enters an invalid command or card value, exception is thrown and the game still continues. This can be seen in gamecontroller.cc:255.

And can be demonstrated when running:

./straights 5 -f. demo11.txt



Note:

The pictures above are taken when running ./straights 5 with inputs of demo11.txt inputted one by one to show the outputs more clearly.

Computer Player

The computer player will play the first card (left most) in its legal plays if it has legal plays and will discard the first card (left most) in it hand if it has no legal plays.

When it plays, the program will print:

Player<x> plays <card>.

When it discards, the program will print:

Player<x> discards <card>.

Since hand and legal plays are not printed when it's a computer player's turn, the demonstration will be done by having a human player ragequits and taken over by a computer player.

Demonstration of computer playing the first card (left most) in its legal plays can be seen **by running the code:**

./straights 5 -f demo12.txt

```
Cards on the table:
Clubs: 5 6 7
Diamonds:
Hearts: 6 7
Spades: 4 5 6 7 8
Your hand: 3D JS AC JD KS 5H 2D 4H 3S 8H
Legal plays: 5H 3S 8H
>ragequit
Player 2 ragequits. A computer will now take over.
Player2 plays 5H.
Player3 plays 4C.
Player4 plays 7D.
```

Demonstration of computer discarding the first card (left most) in its hand can be seen **by running the code:**

./straights 5 -f demo13.txt

```
Cards on the table:
Clubs: 7
Diamonds:
Hearts:
Spades: 6 7 8
Your hand: KH 3D JS AC JD 4S KS 5H 2D 4H 3S 8H
Legal plays:
>ragequit
Player 2 ragequits. A computer will now take over.
Player2 discards KH.
```

Note:

The pictures above are taken when running ./straights 5 with inputs of demo12.txt and demo13.txt respectively inputted one by one to show the outputs more clearly.

Scoring

At the end of each round, for each player the program prints:

Player<x>'s discards:<list of discards>

Player<x>'s score: <old score> + <score gained> = <new score>

If the round ends and no more rounds will be played (this happens when a player's score reaches >= 80), the program will print:

Player<x> wins!

This can be seen by running the code:

./straights 5 -f demo14.txt

```
Player1's discards: AD
Player1's score: 0 + 1 = 1
Player2's discards: KH 3D JS JD KS 2D
Player2's score: 0 + 53 = 53
Player3's discards: KD QS JH QH
Player3's score: 0 + 48 = 48
Player4's discards: KC QD
Player4's score: 0 + 25 = 25
```

demo14.txt

```
1   c
2   h
3   c
4   c
5   play 7S
6   discard KH
7   play 4S
8   play 5H
9   play 4H
10  play AC
11  play 3S
12  play 8H
13  discard 3D
14  discard JS
15  discard JD
16  discard KS
17  discard 2D
18
```

The left picture shows the discards and scoring of the 4 players in the first round. The values of the discard cards are their scores. Each card's values are counted by Card::value() and the total scores of each player is counted by Player::countDiscard(). In this case, since Player2's discard are KH(13) + 3D(3) + JS(11) + JD(11) + KS(13) + 2D(2), its score gained is 53.

Since no players reaches 80 yet, another round will begin, with a new shuffled deck.
Also, in this case, we are playing as Player2. We can see that the discards are ordered according to what it discarded first (see the right picture for the order of the discard).

(continuation of the one above) To demonstrate the second round, which is also the last round, **run:**

./straights 5 -f demo15.txt

```
Player1's discards:
Player1's score: 1 + 0 = 1
Player2's discards: TD KD QC KC
Player2's score: 53 + 48 = 101
Player3's discards: 2S AH QD JD
Player3's score: 48 + 26 = 74
Player4's discards: JC AS
Player4's score: 25 + 12 = 37
Player1 wins!
```

Here, the old scores are added with the new score (according to each player's discards). Since Player2 score is >= 80, there is no new round and the game stops. Since Player1 Has the least score, Player1 wins!

To see multiple players tie for the win, **run:**

./straights 13 -f demo16.txt

```
Player1's discards:
Player1's score: 39 + 0 = 39
Player2's discards: AD KH AS KC
Player2's score: 19 + 28 = 47
Player3's discards: QH TC JC 3S QC
Player3's score: 64 + 48 = 112
Player4's discards: 4S 2S
Player4's score: 33 + 6 = 39
Player1 wins!
Player4 wins!
```

Here, since Player3 has a total score of 122 >= 80, the game ends. Since both Player1 and Player4 have the least score, with score 39, both of them wins!

Extra Credit Features

Handle all memory management via STL containers and smart pointers.