Basics

1. Run `chess` program with command line argument `-demo`.

   ```
   ./chess -demo

   game player1 player2
   8 rnbqkbnr
   7 pppppppp
   6  _ _ _ _
   5 _ _ _ _ 
   4  _ _ _ _
   3 _ _ _ _ 
   2 PPPPPPPP
   1 RNBQKBNR

     abcdefgh

   --Round 1, white player1's turn--
   ```

   The demo is configured for two human players to play the chess game. To start the game, enter `game white-player black-player` in the console. The parameters `white-player` and `black-player` can be either player's name or "computer1" to indicate computer player. The program will automatically reset the board to default and display the board on the screen. White's camp is the first to move a piece by default, and a text message is displayed to remind the player whose turn it is and the round number.

2. The white and black camps take turns moving pieces by the command `move loc1 loc2`. The program checks if that is a valid move. If a piece move is valid, then the board after the move is printed on the screen and shows whose turn it is next.

   ```
   --Round 1, white player1's turn--
   move e2 e4
   8 rnbqkbnr
   7 pppppppp
   6  _ _ _ _
   5 _ _ _ _ 
   4  _ _P_ _
   3 _ _ _ _ 
   2 PPPP PPP
   1 RNBQKBNR
   ```

```
  abcdefgh

--Round 1, black player2's turn--
move c7 c6
8 rnbqkbnr
7 pp_ppppp
6  _p_ _ _
5 _ _ _ _
4  _ _P_ _
3 _ _ _ _
2 PPPP PPP
1 RNBQKBNR

  abcdefgh

--Round 2, white player1's turn--
```

3. When a player wants to end the game by conceding, the `resign` command is used. The program specifies who conceded, and which player gets an point. In the below special case, the program detected the black king is in check and black player2 decided to resign.

```
--Round 11, white player1's turn--
move g5 d8
8 rnbB b r
7 ppk _ppp
6  _p_ _ _
5 _ _ q _
4  _ _n_ _
3 _ _ _ _
2 PPP_ PPP
1 _ KR_BNR

  abcdefgh

--Black is in check.--
--Round 11, black player2's turn--
--Black resigns! White wins!--
```

4. After all the inputs have been read and all of them have been executed, the final score will be displayed on the screen according to the player's colour.

```
Final Score:
White: 1
Black: 0
```

5. The "demobasic.in" file demostrates the commands above

Setup

1. To set up your own initial board configuration, enter `setup` in the console. An empty board will be displayed on the screen.

```
setup
8  _ _ _ _
7 _ _ _ _ 
6  _ _ _ _
5 _ _ _ _ 
4  _ _ _ _
3 _ _ _ _ 
2  _ _ _ _
1 _ _ _ _ 

  abcdefgh
```

2. The command `+ piece loc` places valid pieces to the expected location.

```
+ K e1
8  _ _ _ _
7 _ _ _ _ 
6  _ _ _ _
5 _ _ _ _ 
4  _ _ _ _
3 _ _ _ _ 
2  _ _ _ _
1 _ _ K _ 

  abcdefgh

+ k e8
8  _ _k_ _
7 _ _ _ _ 
6  _ _ _ _
5 _ _ _ _ 
4  _ _ _ _
3 _ _ _ _ 
2  _ _ _ _
1 _ _ K _ 

  abcdefgh
```

If the entered character is not a valid piece, an error message will be displayed.

```
+ & a4
--Invalid piece, please input again--
+ Y h3
--Invalid piece, please input again--
+ R c4
8  _ _k_ _
7 _ _ _ _
6  _ _ _ _
5 _ _ _ _
4  _R_ _ _
3 _ _ _ _
2  _ _ _ _
1 _ _ K _

  abcdefgh
```

3. The command `- loc` removes the pieces from the board. Similarly to the `+` command, if the input is not recognized by the program, an error message will be displayed.

```
8  _ _k_ _
7 _ _ _ _
6  _ _ _ _
5 _ _ _ _
4  _ _ _ _
3 _ _ _ _
2  _ _ _ _
1 _ _ K _

  abcdefgh

- K e1
--Incorect command for set up, please input again--
- e1
8  _ _k_ _
7 _ _ _ _
6  _ _ _ _
5 _ _ _ _
4  _ _ _ _
3 _ _ _ _
2  _ _ _ _
1 _ _ _ _

  abcdefgh
```

4. The command `done` is always expected to exit the setup mode, the program will check if the conditions to exit are met. The board is checked thoroughly from top to bottom and the corresponding error messages will be displayed if the rules are violated.

   The error message will give the player a clear explanation of the particular error, so that the error can be found and corrected much easier.

   ```
   8  pp_k_ P
   7 _ _ _ _
   6  _p_ _ _
   5 _ _ _ Q
   4  _ _ _ _
   3 _ _kR _
   2 K_ _k_ _
   1 K _ P _

     abcdefgh

   done
   --Pawns exist on the first row of the board, cannot exit setup mode--

   8  _ _k_ _
   7 _ _ _ _
   6  _p_ _ _
   5 _ _ _ Q
   4  _ _ _ _
   3 _ _kR _
   2 K_ _k_ _
   1 K _ P _

     abcdefgh

   done
   --Pawns exist on the last row of the board, cannot exit setup mode--
   ```

   Remove the pawns from the first and last rows of the board, the program will check for the next error.

   ```
   8  _ _k_ _
   7 _ _ _ _
   6  _p_ _ _
   5 _ _ _ Q
   4  _ _ _ _
   3 _ _kR _
   ```

```
2 K_ _k_ _
1 K _ _ _

  abcdefgh

done
--More than one black king exists, cannot exit setup mode--

8  _ _k_ _
7 _ _ _ _
6  _p_ _ _
5 _ _ _ Q
4  _ _ _ _
3 _ _ R _
2 K_ _ _ _
1 K _ _ _

  abcdefgh

done
--More than one white king exists, cannot exit setup mode--
```

Remove the extra black kings and white kings, and make sure no kings are in check, then the setup mode can be exited safely.

```
8  _ _k_ _
7 _ _ _ _
6  _p_ _ _
5 _ _ _ Q
4  _ _ _ _
3 _ _ R _
2  _ _ _ _
1 K _ _ _

  abcdefgh

--Black is in check, cannot exit setup mode--
- e3
8  _ _k_ _
7 _ _ _ _
6  _p_ _ _
5 _ _ _ Q
4  _ _ _ _
3 _ _ _ _
2  _ _ _ _
```

```
1 K _ _ _

  abcdefgh
```

5. The command `= colour` should be used before exiting setup mode to make it's `colour`'s turn to go next. The `colour` should either be `white` or `black`.

```
= black
done
--Exit setup mode--
```

6. Following by the command `game white-player black-player`, a new game will be started using the board with the your own bord configuration.

```
game player1 player2
8  _ _k_ _
7 _ _ _ _
6  _p_ _ _
5 _ _ _ Q
4  _ _ _ _
3 _ _ R _
2  _ _ _ _
1 K _ _ _

  abcdefgh

--Round 1, black player2's turn--
```

7. Once the game starts, you are unable to enter the setup mode again.

```
setup
--In game, cannot enter setup mode--
```

8. The "demosetup.in" file demostrates the commands above

Special Moves Examples

1. Capture

   The black pawn at d5 is captured by the white pawn previously at c4, then the white pawn at d5 is captured by the black pawn previously at c6.

```
--Round 9, black player2's turn--
move g8 h6
```

```
8  r qk_ r
7 pp_n_ pp
6  _pbp_ n
5 _ _p_p_
4  _PP _b_
3 _Q_ _NP
2 PP NPPBP
1 R B R K

  abcdefgh

--Round 10, white player1's turn--
move c4 d5
8  r qk_ r
7 pp_n_ pp
6  _pbp_ n
5 _ _P_p_
4  _ P _b_
3 _Q_ _NP
2 PP NPPBP
1 R B R K

  abcdefgh

--Round 10, black player2's turn--
move c6 d5
8  r qk_ r
7 pp_n_ pp
6  _ bp_ n
5 _ _p_p_
4  _ P _b_
3 _Q_ _NP
2 PP NPPBP
1 R B R K

  abcdefgh
```

The white knight at e6 is captured by the black bishop previously at f7.

```
--Round 15, white player1's turn--
move g5 e6
8  r qk_ r
7 pp_n_bpp
6  _ bN_ n
5 _ _ _ _
```

```
4  _ Pp_ _
3 _Q_ _ PP
2 PP _ PB_
1 R B R K

  abcdefgh

--Round 15, black player2's turn--
move f7 e6
8  r qk_ r
7 pp_n_ pp
6  _ bb_ n
5 _ _ _ _
4  _ Pp_ _
3 _Q_ _ PP
2 PP _ PB_
1 R B R K

  abcdefgh
```

2. Castling

   The program checks if the expected castling move statisfies the castling requirements. If it does, castling command is excecuted by move the king piece.

```
--Round 4, white player1's turn--
move e1 g1
8 rn qkbnr
7 pp_ pppp
6  _p_ _ _
5 _ _p_ _
4  _ _ _b_
3 _ _ _NP
2 PPPPPPBP
1 RNBQKR_

  abcdefgh

--King castling--
8 rn qkbnr
7 pp_ pppp
6  _p_ _ _
5 _ _p_ _
4  _ _ _b_
3 _ _ _NP
2 PPPPPPBP
```

```
1 RNBQ_RK

  abcdefgh
```

Otherwise, a detailed error message will be displayed, specifying which rook or king have been moved before. For example,

```
--Round 16, black player2's turn--
move g7 g6
8  r qk_ r
7 pp_n_ _p
6  _ bb_pn
5 _ _ _ _
4  _ Pp_ _
3 _Q_ _ PP
2 PP _ PB_
1 R B RK_

  abcdefgh

--Round 17, white player1's turn--
move f1 d1
--White king have moved, cannot castling--
```

Computer

1. computer1

   Computer1 makes chooses a random piece and makes a random move (including special moves). Computer1 can also randomly replace pawns that reached the other end of board.

```
--Round 63, white computer1's turn--
8  PQ_ _k_
7 _ _ _ _
6  _ _ Pb_
5 _ _b_r_
4  _ _ Qp_
3 _ _ _ K
2  b _R_ _
1 _ _q_ _

  abcdefgh

--computer reached the other end, replacing P with N--
8  NQ_ _k_
```

```
7 _ _ _ _
6  _ _ Pb_
5 _ _b_r_
4  _ _ Qp_
3 _ _ _ K
2  b _R_ _
1 _ _q_ _

  abcdefgh
```

2. human vs computer1

   By command `game white-player computer1` or `game computer1 black-player`, you can start a game with a computer player. When computer captures your king, you lose this game.

3. computer1 vs computer1

   By command `game computer1 computer1`, you can start a game with computer1 vs computer1. The game will automatically end when one king has been captured.

```
--Round 65, black computer1's turn--
--white king has been captured, black wins--
```

Error handling

The program is designed to handle invalid inputs:

1. You cannot access the board out of bounds.

```
8 rnbqkbnr
7 pppppppp
6  _ _ _ _
5 _ _ _ _
4  _ _ _ _
3 _ _ _ _
2 PPPPPPPP
1 RNBQKBNR

  abcdefgh

--Round 1, white player1's turn--
move i2 i3
--Out of bounds, please input again--
```

2. You cannot violate the moving rules of pieces.

```
8 rnbqkbnr
7 ppppppp
6  _ _ _ p
5 _ _ _ _
4  _ _ _ _
3 _ _ _ _P
2 PPPPPPP_
1 RNBQKBNR

  abcdefgh

--Round 2, white player1's turn--
move g2 h2
--Invalid move, please input again--
```

3. You cannot move empty square or the pieces from opposite camps.

```
8 rnbqkbnr
7 ppppppp
6  _ _ _ p
5 _ _ _ _
4  _ _ _ _
3 _ _ _ _P
2 PPPPPPP_
1 RNBQKBNR

  abcdefgh

move e3 e4
--Not your piece, please move again--
--Round 2, white player1's turn--
move g7 g6
--Not your piece, please move again--
```

4. You cannot capture your own piece.

```
8 rnbqkbnr
7 ppppppp
6  _ _ _ p
5 _ _ _ _
4  _ _ _ _
3 _ _ _ _P
2 PPPPPPP_
1 RNBQKBNR

  abcdefgh

move g2 h3
--You cannot capture your own pieces, please input again--
```

5. You need to enter the correct form of commad.

```
move e2 to e3
--Out of bounds, please input again--
--Round 2, white player1's turn--
--Your command is not in the correct form, please input a string--
--Round 2, white player1's turn--
e2 to e3
--Incorect command, please input again--
--Round 2, white player1's turn--
move e2 e3
8 rnbqkbnr
7 ppppppp
6  _ _ _ p
5 _ _ _ _
4  _ _ _ _
3 _ _ P _P
2 PPPP PP_
1 RNBQKBNR

  abcdefgh
```