ComS 327 Project Part 2

Checkers: process moves

Fall 2019

1 Summary

For the second part of the project, you must read an input file that describes a game configuration, and output the new game configuration reached after a sequence of moves. If the input file is invalid, an appropriate error message should be written to standard error (including the line number of the input file that caused the error), nothing should be written to standard output, and the program should terminate.

Specifically, your `Makefile` must now build the executable for part 1, and a new executable named `change`. Your `change` program will read an input file, perform the specified moves (with an appropriate message if a move is illegal), display information to standard output, and write the resulting configuration to output file(s) in the format(s) described below. Program `change` should accept the following optional command-line switches and arguments (you may implement additional switches, for example with debugging information, if you wish) which may appear in any order.

- The input filename, guaranteed not to begin with a `'-'` character. If omitted, read from standard input.
- Switch `-e efilename` (for "exchange"), says to create file named `efilename` and write the resulting configuration to that file in the "exchange" format.
- Switch `-h hfilename` (for "human readable") says to create file named `hfilename` and write the resulting configuration to that file in the "human readable" format.
- Switch `-m N`, indicating that at most N moves should be played. If omitted, all moves given in the input file should be played.

Note that program `change` should create more than one output file if more than one `-e`, `-h` switch is given.

2 Movement rules

1. Pieces must remain on black squares at all times.
2. Red pawns move by incrementing the rank by one, and either incrementing or decrementing the file by one. The destination square must exist and be empty.
3. Red pawns may capture (or "jump") by incrementing the rank by two, and either incrementing or decrementing the file by two. The destination square must exist and be empty, and the "jumped over square" must contain a black pawn or king, which is immediately removed from the board.
4. Whenever a red pawn reaches rank 8, it is promoted to a red king.
5. Black pawns move by decrementing the rank by one, and either incrementing or decrementing the file by one. The destination square must exist and be empty.
6. Black pawns may capture (or "jump") by decrementing the rank by two, and either incrementing or decrementing the file by two. The destination square must exist and be empty, and the "jumped over square" must contain a red pawn or king, which is immediately removed from the board.

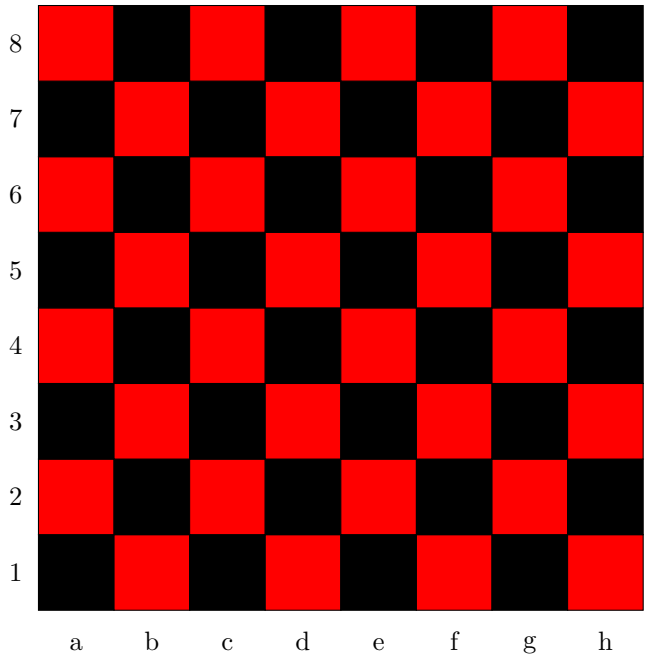

Figure 1: Standard checkerboard coordinates (ranks 1–8, files a–h)

7. Whenever a black pawn reaches rank 1, it is promoted to a black king.
8. Kings can move and capture like black pawns or like red pawns.
9. If "multiple jumps" are allowed (as specified in the input file), a capture move may be followed by additional capture moves by the same piece, as part of the same turn.
10. If the "capture" rule is enforced (as specified in the input file), then a player must capture a piece if possible. More precisely, if a player has one or more capture moves available on her turn, then she cannot choose a move that does not capture any piece.

3 Standard output

The following should be written to standard output (unless the input file has syntax errors) if all moves in the input file are legal, or if the first M moves are legal:

```
Input file has N moves
Processing M moves
All moves are legal
```

where N is the total number of moves given in the input file, and M is the total number of moves processed (which might be less than N if the `-m` switch is given). Any output files created will contain the game configuration after M moves. If move j is illegal, with $j \leq M$, then instead the following should be written to standard output:

```
Input file has N moves
Processing M moves
Move j is illegal: (show move j here)
```

Any output files created will contain the game configuration before the first illegal move.

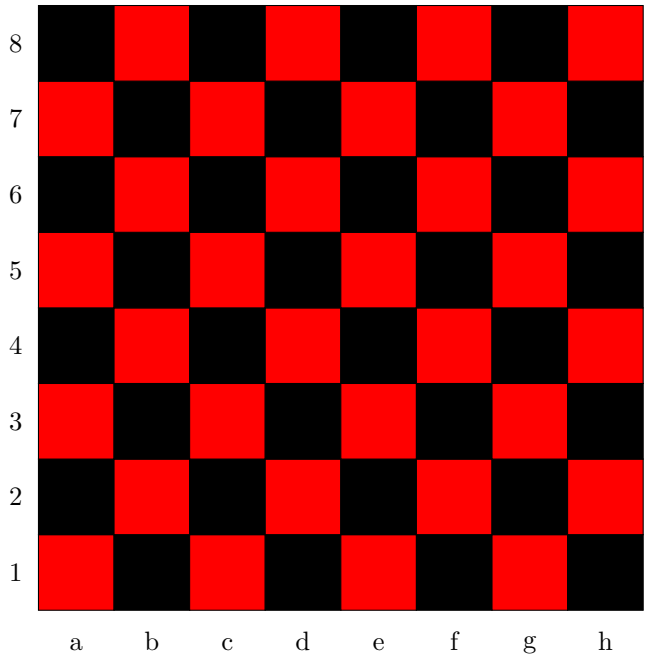

Figure 2: A "flipped" checkerboard (ranks 1–8, files a–h)

4 Exchange format `-e`

Output files in the exchange format should have the same format as the input files. This means that exchange files can then be used as input files to both the `info` and `change` utilities. The output file should contain the moves (if any) that are set to occur after the game configuration specified in the output file. This means, for example, if the input file has N moves, and all moves were processed and legal, then the output file should not contain any moves. If fewer than N moves were processed (either because the `-m` switch was given, or an illegal move was found) then any unprocessed moves should be copied into the output file.

5 Output format `-h`

Output files in the human readable format should display the current board (with grid coordinates), followed by the remaining list of moves (if any), in precisely the following format (with the appropriate pieces on the board):

```
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | " | b | " | b | " | b | " | b | 8
  +---+---+---+---+---+---+---+---+
7 | b | " | b | " | b | " | b | " | 7
  +---+---+---+---+---+---+---+---+
6 | " | b | " | b | " | b | " | b | 6
  +---+---+---+---+---+---+---+---+
5 | . | " | . | " | . | " | . | " | 5
  +---+---+---+---+---+---+---+---+
4 | " | . | " | . | " | . | " | . | 4
  +---+---+---+---+---+---+---+---+
3 | r | " | r | " | r | " | r | " | 3
  +---+---+---+---+---+---+---+---+
```

```
2 | " | r | " | r | " | r | " | r | 2
  +---+---+---+---+---+---+---+---+
1 | r | " | r | " | r | " | r | " | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h

(move)
(move)
(move)
```

Note that the characters shown in the grid are the same ones used in the input file format:

- `'"'`: an empty red square.
- `'.'`: an empty black square.
- `'b'`: a square containing a black pawn.
- `'B'`: a square containing a black king.
- `'r'`: a square containing a red pawn.
- `'R'`: a square containing a red king.

In each output line, there should be no leading spaces before a move, or before the rank numbers. There should be exactly one blank line after the board.

6 Examples

6.1 Input file: `start4.txt`

```
# Start of a game

RULES:
  no capture      # Don't enforce the rule that pieces must be captured
  multiple jumps  # Allow multiple jumps

TURN:
  red

BOARD:

 " | b | " | b | " | b | " | b
---+---+---+---+---+---+---+---
 b | " | b | " | b | " | b | "
---+---+---+---+---+---+---+---
 " | b | " | b | " | b | " | b
---+---+---+---+---+---+---+---
 . | " | . | " | . | " | . | "
---+---+---+---+---+---+---+---
 " | . | " | . | " | . | " | .
---+---+---+---+---+---+---+---
 r | " | r | " | r | " | r | "
---+---+---+---+---+---+---+---
 " | r | " | r | " | r | " | r
```

```
---+---+---+---+---+---+---+---
 r | " | r | " | r | " | r | "


MOVES:
# Red             black
e3->d4            b6->c5
d4->b6            c5->b4

# The fourth move is illegal, because the piece that was at c5
# was captured in the third move.
```

6.2 Example run 1

If we run

```
./change -m 10 start4.txt -h human4.txt -e exchange4.txt
```

we should get the following on standard output:

```
Input file has 4 moves
Processing 4 moves
Move 4 is illegal: c5->b4
```

we should get the following written to `human4.txt`:

```
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | " | b | " | b | " | b | " | b | 8
  +---+---+---+---+---+---+---+---+
7 | b | " | b | " | b | " | b | " | 7
  +---+---+---+---+---+---+---+---+
6 | " | r | " | b | " | b | " | b | 6
  +---+---+---+---+---+---+---+---+
5 | . | " | . | " | . | " | . | " | 5
  +---+---+---+---+---+---+---+---+
4 | " | . | " | . | " | . | " | . | 4
  +---+---+---+---+---+---+---+---+
3 | r | " | r | " | . | " | r | " | 3
  +---+---+---+---+---+---+---+---+
2 | " | r | " | r | " | r | " | r | 2
  +---+---+---+---+---+---+---+---+
1 | r | " | r | " | r | " | r | " | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h

c5->b4
```

and we should get something like the following written to `exchange4.txt`:

```
#
# Automatically generated by running
# ./change -m 10 start4.txt -h human4.txt -e exchange4.txt
#

RULES:
  no capture
```

```
  multiple jumps
TURN:
  black
BOARD:
-----------------
|"|b|"|b|"|b|"|b|
|b|"|b|"|b|"|b|"|
|"|r|"|b|"|b|"|b|
|.|"|.|"|.|"|.|"|
|"|.|"|.|"|.|"|.|
|r|"|r|"|.|"|r|"|
|"|r|"|r|"|r|"|r|
|r|"|r|"|r|"|r|"|
-----------------
MOVES:
c5->b4
```

6.3 Example run 2

If we run

```
./change -e exchange4a.txt -m 1 < start4.txt
```

we should get the following on standard output:

```
Input file has 4 moves
Processing 1 moves
All moves are legal
```

and we should get something like the following written to `exchange4a.txt`:

```
#
# Automatically generated by running
# ./change -e exchange4a.txt -m 1
#

RULES:
  no capture
  multiple jumps
TURN:
  black
BOARD:
-----------------
|"|b|"|b|"|b|"|b|
|b|"|b|"|b|"|b|"|
|"|b|"|b|"|b|"|b|
|.|"|.|"|.|"|.|"|
|"|.|"|r|"|.|"|.|
|r|"|r|"|.|"|r|"|
|"|r|"|r|"|r|"|r|
|r|"|r|"|r|"|r|"|
-----------------
MOVES:
b6->c5
d4->b6
c5->b4
```

6.4 Example run 3

If we run

```
./change exchange4a.txt
```

we should get the following on standard output:

```
Input file has 3 moves
Processing 3 moves
Move 3 is illegal: c5->b4
```

and note that no output files will be created.

7 Required functionality

Your `change` program should work correctly for "standard boards", "single jumps", and "no capture" (i.e., players are not required to capture a piece). If an input file specifies a feature that you have not implemented, your code should exit cleanly with an appropriate error message to standard error (something like, "Sorry, multiple jumps are not implemented").

If you are working in a group of 2, you must complete at least one extra-credit feature (see below).

8 Extra credit features

- Allow input files with flipped boards and adjust game play accordingly.
- Allow input files that specify the "capture" rule, and adjust game play accordingly.
- Allow input files that specify "multiple jumps", and adjust game play accordingly.

9 What to submit

Remember to submit the following in your git repository.

- C source code.
- A `Makefile`, so your executables can be built by simply typing "`make`".
- A `README` file, to indicate which features are implemented.
- A `DEVELOPERS` file, with necessary information about the source code for other developers. For group submissions, this file also should indicate which student(s) implemented which function(s).
- A `GRADEME` file, containing either

  > "Please grade my updated part 1 submission, for at most half credit. I understand that this could lower my grade for part 1."

  OR

  > " "

  An omitted file will be treated the same as an empty file.

Also, please turn in a short note in Canvas with the URL of your git repository and the members of your group.