Alexandria University Faculty

of Engineering

Computer and Systems Engineering

Dept. CSE 121: Programming 1

Programming Project Report

Chess using C

Team members:

Name	ID	Number
Ramy Ahmed El Sayed	19015649	25
Ziad Samy Ramadan	19015720	28

Description

Our project is a Chess console application made in C programming language.

It provides the essential blocks found in any chess game, which are:

- A labelled chess board
- Two players competing against each other (Black and White)
- The special movement of each piece
- The core mechanics of the game (Promotion, Stalemate, and Checkmate)

In addition to the mentioned mechanics, some additional features were added to make the experience more flexible and enjoyable, such as:

- The ability to save and load your concurrent game
- The ability to undo and redo your actions

Design overview

- Our design is mainly focused on our chess board. Our game functions in an infinite loop that breaks when one of the terminating game conditions occur (Checkmate or Stalemate).
- It checks for the user input depending on the specified player turn (Player 1 or Player 2) and, checks whether the input is valid or not.
- It then checks the piece picked and whether the target position is in the boundaries of the piece's movements.
- Continuously, checks for a Checkmate or a Stalemate.
- Depending on the user input, various functions can occur, such as:
  - Saving
  - Loading
  - Undo
  - Redo
- Once one of the terminating conditions occur, It breaks the loop and terminates the program.

Design assumptions

It is assumed that the number of moves will not exceed 1000, since the lack of dynamic arrays in C, and the complexity of using other data structures, which would behave like a dynamic array (e.g., Hash tables or Linked lists).

Used Data structures

Our program mainly utilizes one data structure, which is Arrays.

We have several arrays that behave as main parameters for most of the functions, such as:

1. Chess board (an 8x8 array that contains the pieces and the tiles)
2. Moves (an array that records the moves done in each turn)
3. Pre-undo array (records the moves that will be undone)
4. Units removed (records the units removed)
5. Promotion array
6. Black and white king position arrays

Important functions

`print_board()`:

Our main function that prints our updated Chess board with the removed units and moved units.

`is_legalmove()`:

Checks if the move is valid to proceed to modify the main chess board array, and checks whether the moves will activate any special game states (Promotion/Stalemate/Checkmate).

`modify()`:

Modifies our Chess board, according to the user input, after checking whether it is valid or not. It is also used when redoing a move.

`is_check():`

Keeps checking on the king's current position (Black or White depending on the current turn) and checks whether any unit of the opposing team can capture the king, if any piece can capture it; then the king is in check.

`is_checkmate():`

If the **`is_check()`** function shows that a king is in check, then check if it can be denied by the threatened king or his pieces. If it cannot be denied, then a checkmate has been concluded and the game ends.

`is_stalemate():`

If a stalemate is reached, the game terminates with a draw.

Pseudo code

`Infinite loop pseudo code:`

```
Create grid board [8][8]
Fill Grid board.
Print Grid.
Start infinite loop: (1)
Ask for move.
If the move is invalid (ex AB35)
     return to (1)
If the input is save:
     save the board and turns in the text file.
     return to (1)

If the input is load:
     load the board and turns from the text file.
     return to (1)
```

If input is undo and it is not the first turn:

save the last move.

undo the action and return to the last turn.

return to (1)

If the input is redo and the number of redos is less than the number of undos:

check the saved moves before undoing and redo them.

go to the next turn.

If the move literals are not between ‘A’ & ‘H’ or the numbers are not between ‘1’ & ‘8’:

display error.

return to (1)

If the piece used is not the player’s:

display error.

return to (1)

Checks whether the move done by the piece is valid.

if the move is valid:

if it is a king:

check if it is the king that is moved and update its position.

if there is a checkmate or a stalemate:

end game

else:

update the board and remove any pieces attacked by it.

return to (1)

else:

update the board and proceed to the next turn.

return to (1)

else:

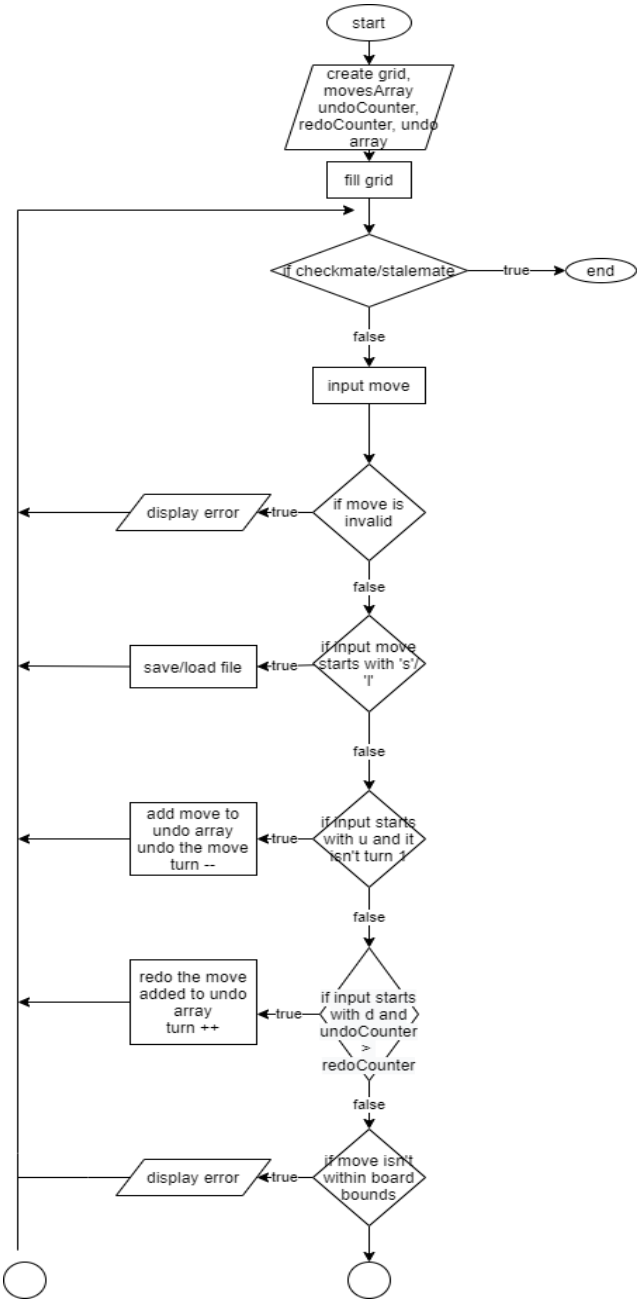
print error message.

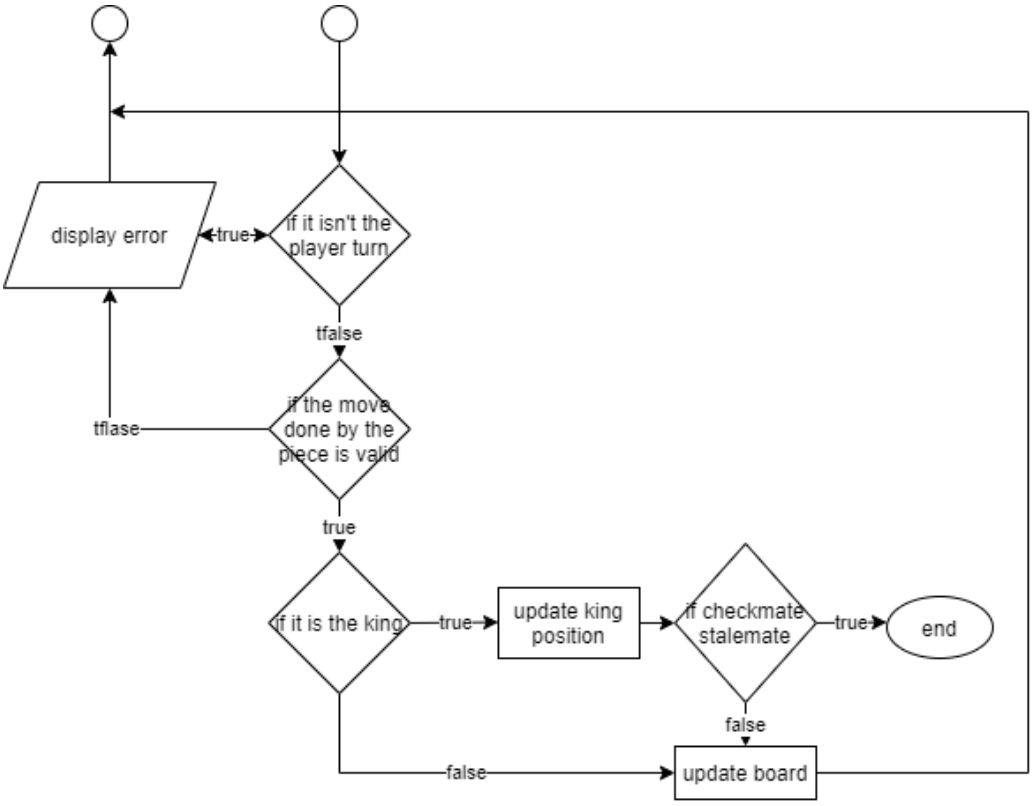
return to (1)

change player turn.

return to (1)

Infinite loop flowchart:





is_legalmove() pseudo code:

```
is_legalmove(piece, initial position, target position, chess
board, king check)

    if king check is true:
        create a temp grid in which the move is made.
        if the team is white/black:
            get the black/white king's position.
            if the king's position is threatened in the temp
             grid:
                print warning message.
                return false till the king is safe.
```

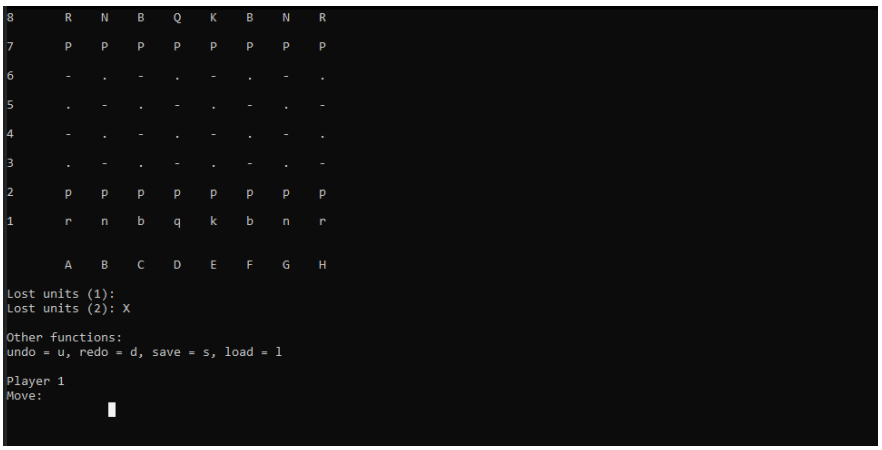
```
check piece letter (r/R, n/N, p/P, b/B, q/Q, k/K)
     if r/R:
          if initial position x/y = final position x/y:
               move one cell at a time
                vertically/horizontally.
               if a cell is not empty:
                    return false.
               if the final cell is not empty:
                    if it has a piece of the same team:
                         return false.
          return true.
     if n/N:
          if the displacement between the initial and
          target position is 3:
               check if the cell isn't empty:
                    check if it is the same team:
                         return false.
               return true.
          return false.

     if b/B:
          if abs(xdifference) = abs(ydifference):
               loop over each cell:
                    if it is not empty:
                         return false.
               if the last cell is not empty:
                    if the piece is from the same team:
                         return false.
               return true.
          return false.
     if q/Q:
```

```
      check for both the rook and bishop movement.


if k/K:
      copies grid to a temp grid.
      modify the move on the temp grid and check it
      if it leads to a checkmate:
            return false.
      check the sum of xdiff and ydiff.
      if sum = 1:
            if the tile is not empty:
                  if the piece is from the same team:
                        return false.
            Return true.
      Else if sum = 2:
            If it is on the same line:
                  Return false.
            Else:
                  if the tile is not empty:
                        if the piece is from the same
                        team:
                              return false.
                  Return true.
if p/P:
      if initialX != targetX:
            if it is only displaced by 1 diagonally:
                  if the tile is not empty:
                        if the piece on the tile is from
                        the same team:
                              return false.
                        return true.
            Return false.
```

```
                Else:
                    If yDiff = 1:
                        If tile is empty:
                            If it is the last tile on board:
                                Promote the pawn.
                            Return true.
                        Return false.
                    Else if yDiff = 2 and pawn in starting position:
                        Record the position of 1 or 2
                        movements.
                        If those positions are empty:
                            Return true if he picks any of
                            those.
                        Return false.
                    Return false.
            Break.
```

User manual

When the user starts the .exe, he is prompted by this screen:

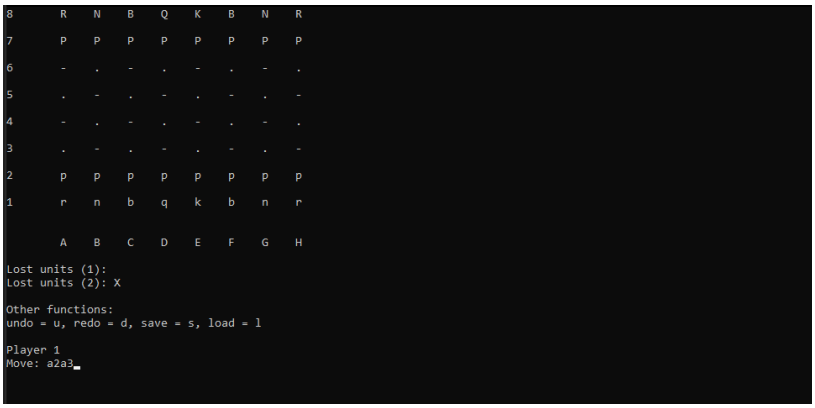


The screen contains:

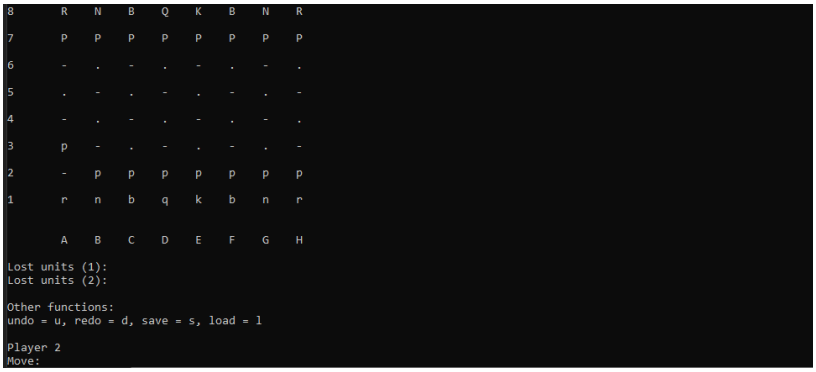
- The chess board.
- The lost units.
- Instructions for other functions.
- Player’s turn.
- Move to be inserted.

User input:

The user is required to insert his move in this format, namely (A2A3/a2a3):



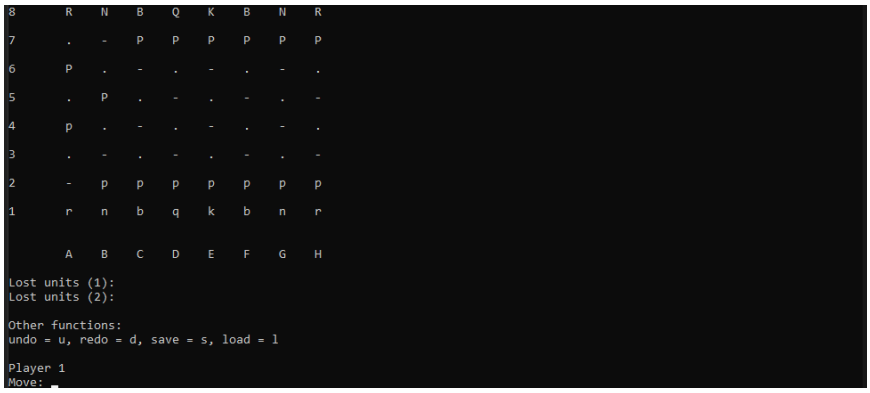
After inserting the input, the user is prompted with an updated chess board:



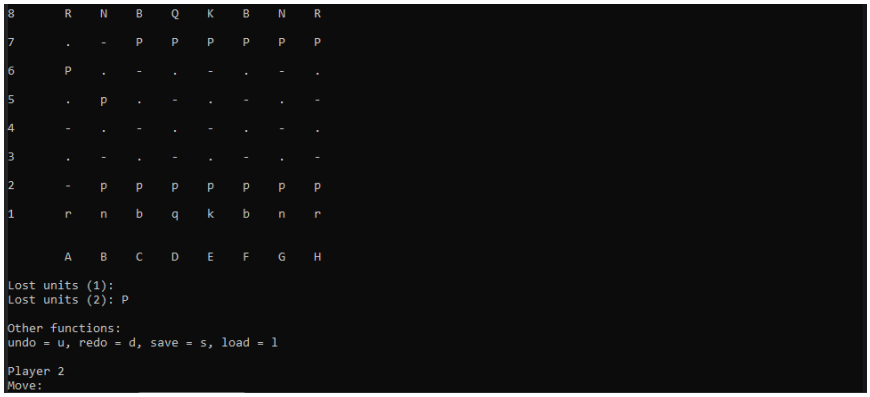
Undo/Redo:

If the user inserts ‘u’, the game undoes his last action.

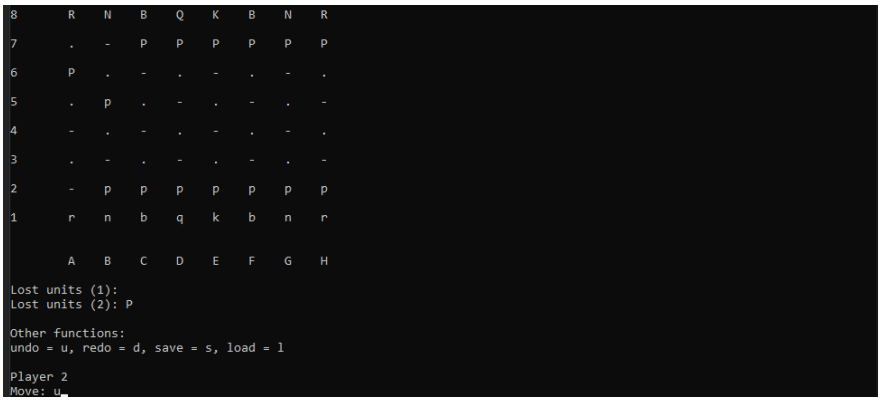
Example:



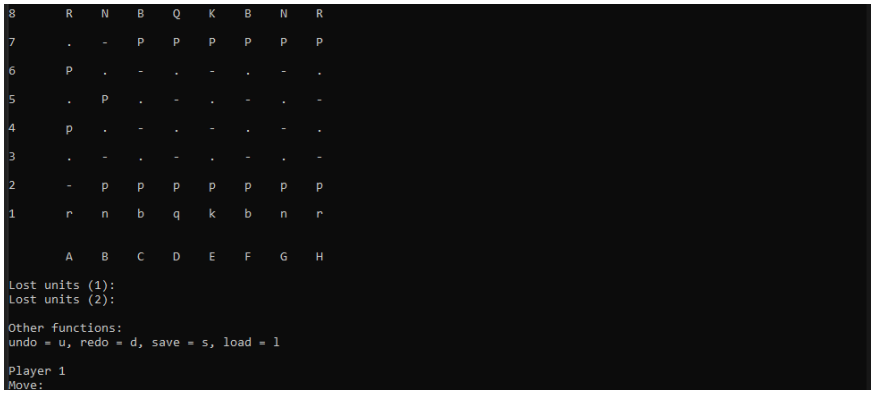
Player 1 will eat Player 2’s pawn located in B5:



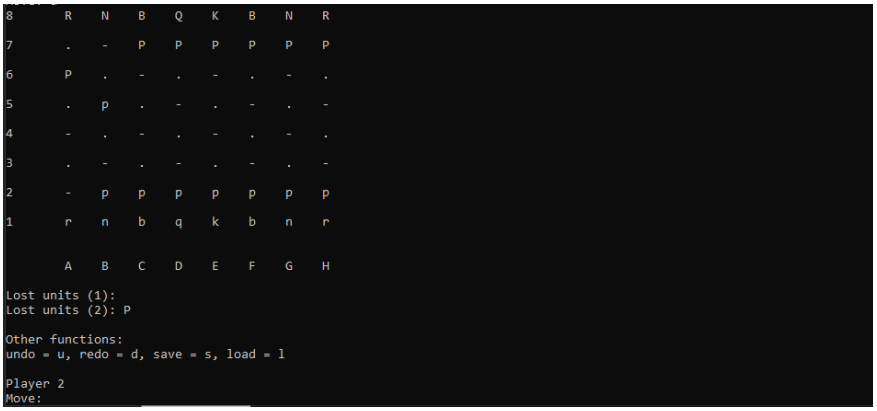
The pawn is now added to Player 2’s lost units. If we want to undo the action, we insert ‘u’:



Which will result in:

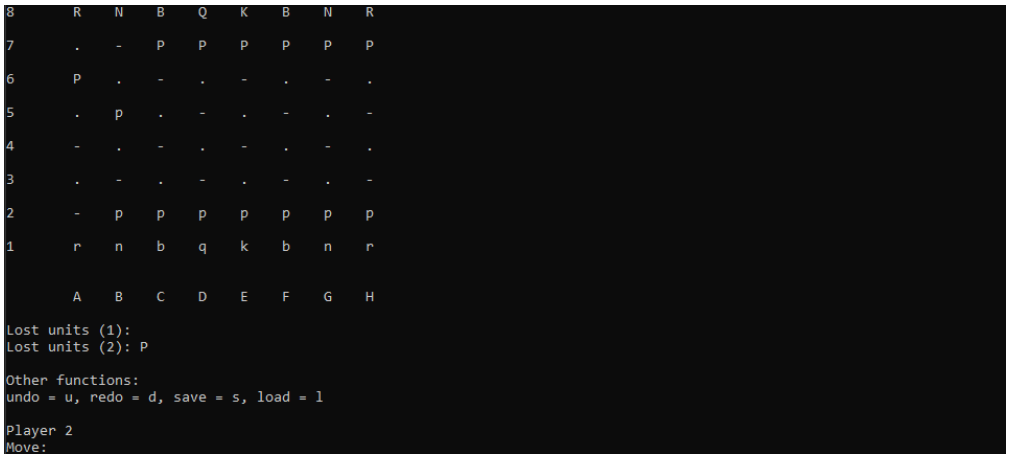


And to redo what we just undid, the user can simply insert ‘d’:

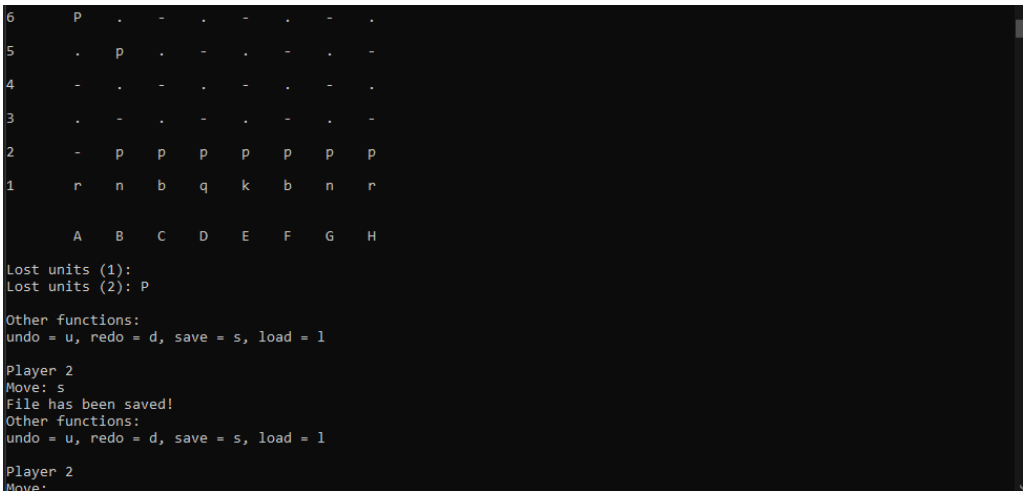


Save/Load:

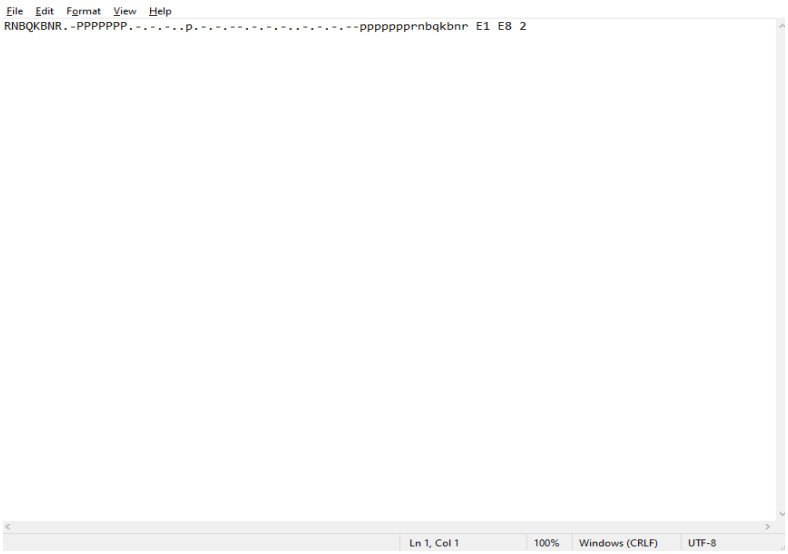
Supposing the player wants to save his chess match to continue later, for example:



If the player inserts ‘s’ in the input, the grid board is saved in a text file:

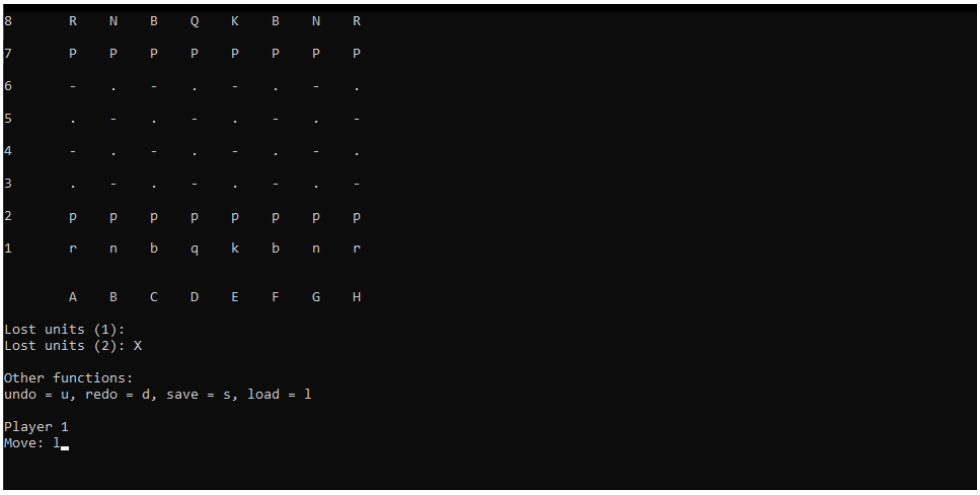


Which will look like this:

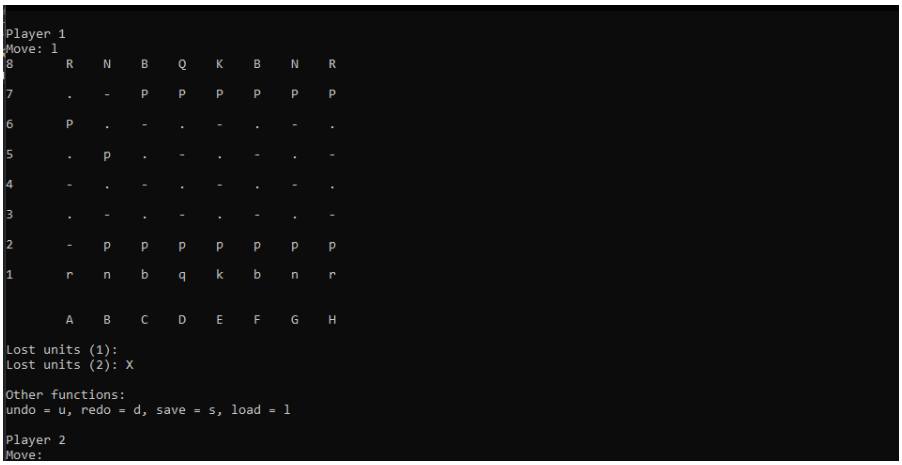


The text file contains the grid board, black king's position, white king's position, and the turn at which it was saved.

If the player were to proceed and close the game and re-open, then type "l" as an input, the game will load and will proceed as usual:

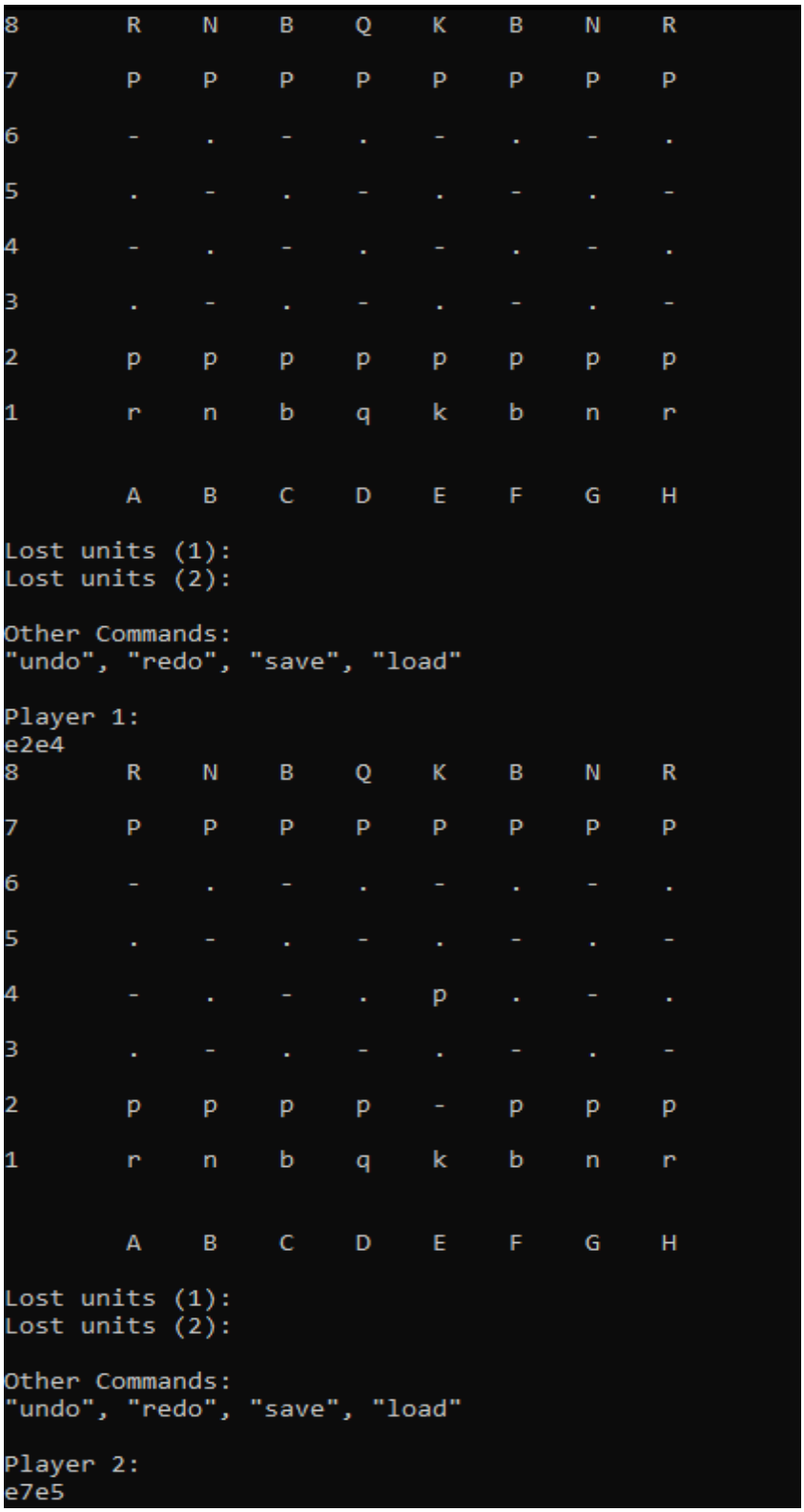


The loaded game:



Test cases

1.


8 R N B Q K B N R
7 P P P P P P P P
6 - . - . - . - .
5 . - . - . - . -
4 - . - . - . - .
3 . - . - . - . -
2 p p p p p p p p
1 r n b q k b n r
A B C D E F G H
Lost units (1):
Lost units (2):
Other Commands:
"undo", "redo", "save", "load"
Player 1:
e2e4
8 R N B Q K B N R
7 P P P P P P P P
6 - . - . - . - .
5 . - . - . - . -
4 - . - . p . - .
3 . - . - . - . -
2 p p p p - p p p
1 r n b q k b n r
A B C D E F G H
Lost units (1):
Lost units (2):
Other Commands:
"undo", "redo", "save", "load"
Player 2:
e7e5

```
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    .    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    P    -    .    -

4       -    .    -    .    p    .    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    -    p    p    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 1:
d1f3
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    .    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    P    -    .    -

4       -    .    -    .    p    .    -    .

3       .    -    .    -    .    q    .    -

2       p    p    p    p    -    p    p    p

1       r    n    b    -    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 2:
b8c6
```

```
8       R    .    B    Q    K    B    N    R

7       P    P    P    P    .    P    P    P

6       -    .    N    .    -    .    -    .

5       .    -    .    -    P    -    .    -

4       -    .    -    .    p    .    -    .

3       .    -    .    -    .    q    .    -

2       p    p    p    p    -    p    p    p

1       r    n    b    -    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 1:
f1c4
8       R    .    B    Q    K    B    N    R

7       P    P    P    P    .    P    P    P

6       -    .    N    .    -    .    -    .

5       .    -    .    -    P    -    .    -

4       -    .    b    .    p    .    -    .

3       .    -    .    -    .    q    .    -

2       p    p    p    p    -    p    p    p

1       r    n    b    -    k    -    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 2:
c6d4
```

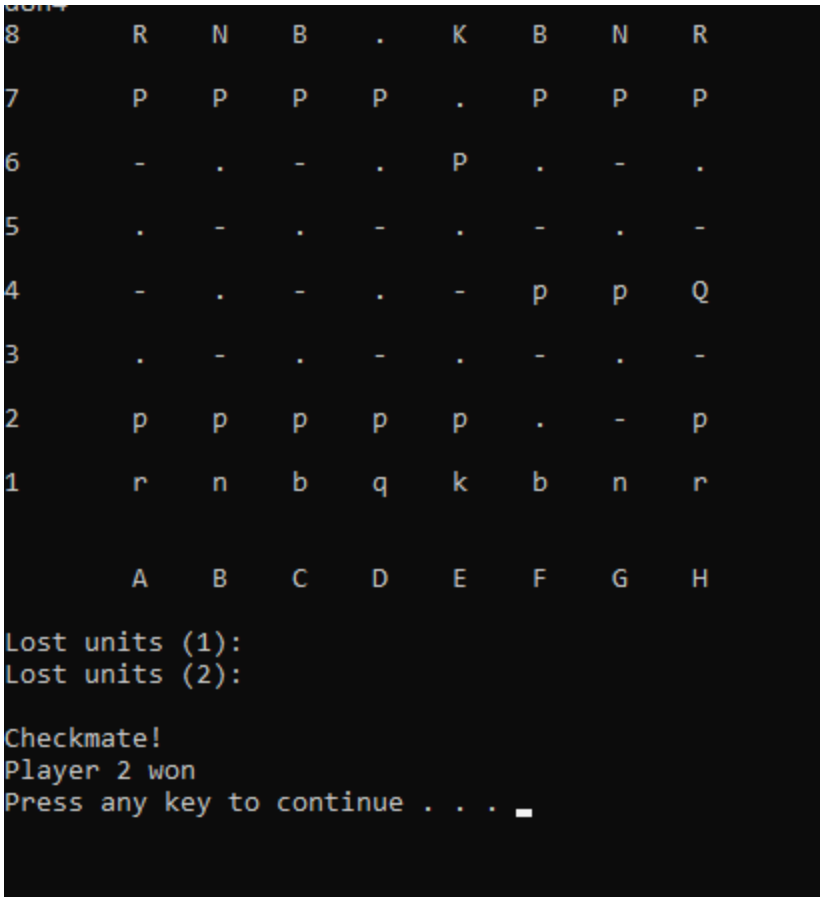
```
8       R    .    B    Q    K    B    N    R

7       P    P    P    P    .    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    P    -    .    -

4       -    .    b    N    p    .    -    .

3       .    -    .    -    .    q    .    -

2       p    p    p    p    -    p    p    p

1       r    n    b    -    k    -    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 1:
f3f7
8       R    .    B    Q    K    B    N    R

7       P    P    P    P    .    q    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    P    -    .    -

4       -    .    b    N    p    .    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    -    p    p    p

1       r    n    b    -    k    -    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2): P

Checkmate!
Player 1 wonPress any key to continue . . .
```

2.

```
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    P    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    .    -    .    -

4       -    .    -    .    -    .    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    p    p    p    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 1:
f2f4
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    P    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    .    -    .    -

4       -    .    -    .    -    p    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    p    .    p    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 2:
d7d5
```

```
8       R    N    B    Q    K    B    N    R

7       P    P    P    -    P    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    P    .    -    .    -

4       -    .    -    .    -    p    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    p    .    p    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 1:
undo
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    P    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    .    -    .    -

4       -    .    -    .    -    p    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    p    .    p    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 2:
e7e6
```

```
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    .    P    P    P

6       -    .    -    .    P    .    -    .

5       .    -    .    -    .    -    .    -

4       -    .    -    .    -    p    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    p    .    p    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 1:
g2g4
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    .    P    P    P

6       -    .    -    .    P    .    -    .

5       .    -    .    -    .    -    .    -

4       -    .    -    .    -    p    p    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    p    .    -    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 2:
d8dh4

Please insert a valid position
Other Commands:
"undo", "redo", "save", "load"

Player 2:
d8h4
```


8 R N B . K B N R
7 P P P P . P P P
6 - . - . P . - .
5 . - . - . - . -
4 - . - . - p p Q
3 . - . - . - . -
2 p p p p p . - p
1 r n b q k b n r
A B C D E F G H
Lost units (1):
Lost units (2):
Checkmate!
Player 2 won
Press any key to continue . . .

3. Testing save/load

```
8       R    N    B    .    -    B    N    R

7       P    P    P    K    Q    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    p    -    .    -

4       -    .    -    .    -    .    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    .    p    p    p    p

1       r    n    b    -    k    b    n    r

        A    B    C    D    E    F    G    H

Lost units (1): q
Lost units (2): P  P

Other Commands:
"undo", "redo", "save", "load"

Player 1:
```

```
Other Commands:
"undo", "redo", "save", "load"

Player 1:
save
File has been saved!
Other Commands:
"undo", "redo", "save", "load"

Player 1:
```

```
8       R    N    B    Q    K    B    N    R

7       P    P    P    P    P    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    .    -    .    -

4       -    .    -    .    -    .    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    p    p    p    p    p

1       r    n    b    q    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1):
Lost units (2):

Other Commands:
"undo", "redo", "save", "load"

Player 1:
load
```

```
Player 1:
load
8       R    N    B    .    -    B    N    R

7       P    P    P    K    Q    P    P    P

6       -    .    -    .    -    .    -    .

5       .    -    .    -    p    -    .    -

4       -    .    -    .    -    .    -    .

3       .    -    .    -    .    -    .    -

2       p    p    p    .    p    p    p    p

1       r    n    b    -    k    b    n    r


        A    B    C    D    E    F    G    H

Lost units (1): q
Lost units (2): P  P

Other Commands:
"undo", "redo", "save", "load"

Player 1:
```

4.Stalemate(s)