PROGRAMMING ASSIGNMENT 3

TAs : Bahar GEZİCİ, Nebi YILMAZ
Due Date : 11.12.2020 (23:00)
Click here to accept your Programming Assignment 3.

1 Introduction

In this assignment, you will get familiar with file operations, functions, lists, and design a relatively complex algorithm. For this purpose, you are going to write a chess simulator which will be able to take specific commands from the user (as a file), execute the commands, and return the results (if any) to the user.

2 BackGround

2.1 Chess

Chess is a two-player board game played on a chessboard, a square-checkered board with 64 squares arranged in an eight-by-eight grid. Each player begins the game with sixteen pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns.The object of the game is to checkmate the opponent's king, whereby the king is under immediate attack (in "check") and there is no way to remove or defend it from attack on the next move.

Initial position: first row: rook, knight, bishop, queen, king, bishop, knight, and rook; second row: pawns

2.2 Setup

Chess is played on a square board of eight rows (called ranks and denoted with numbers 1 to 8) and eight columns (called files and denoted with letters a to h) of squares. The colors

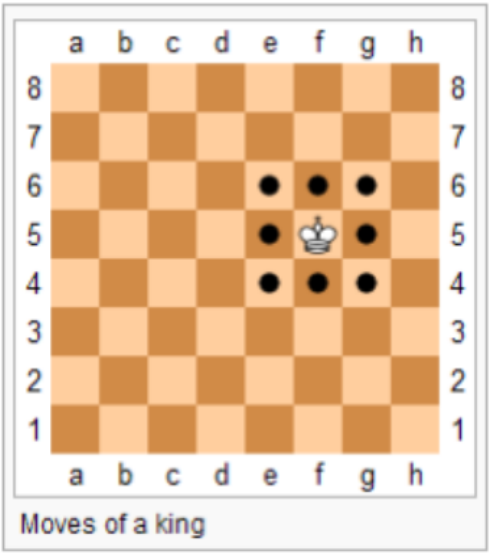
of the sixty-four squares alternate and are referred to as "light squares" and "dark squares". The chessboard is placed with a light square at the right hand end of the rank nearest to each player, and the pieces are set out as shown in the diagram, with each queen on its own color. The pieces are divided, by convention, into white and black sets. The players are referred to as "White" and "Black" and each begins the game with sixteen pieces of the specified color. These consist of one king, one queen, two rooks, two bishops, two knights, and eight pawns.

2.3 Movement

White always moves first. After the initial move, the players alternately move one piece at a time (with the exception of castling, when two pieces are moved). Pieces are moved to either an unoccupied square or one occupied by an opponent's piece, capturing it and removing it from play. With the sole exception of en passant, all pieces capture opponent's pieces by moving to the square that the opponent's piece occupies. A player may not make any move that would put or leave his king under attack. If the player to move has no legal moves, the game is over; it is either a checkmate—if the king is under attack—or a stalemate—if the king is not.

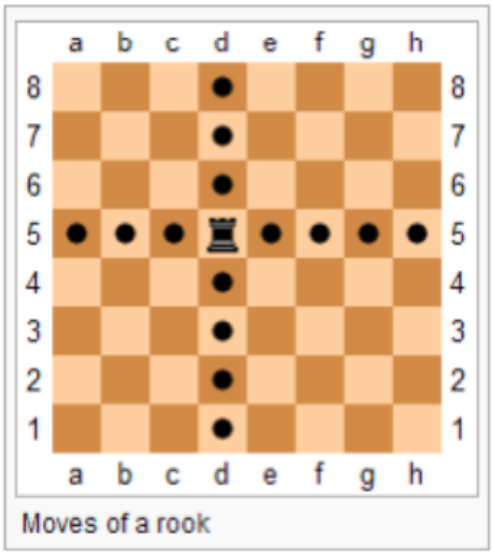
Each chess piece has its own style of moving. In the diagrams, the dots mark the squares where the piece can move if no other pieces (including one's own piece) are on the squares between the piece's initial position and its destination.

- The king moves one square in any direction.



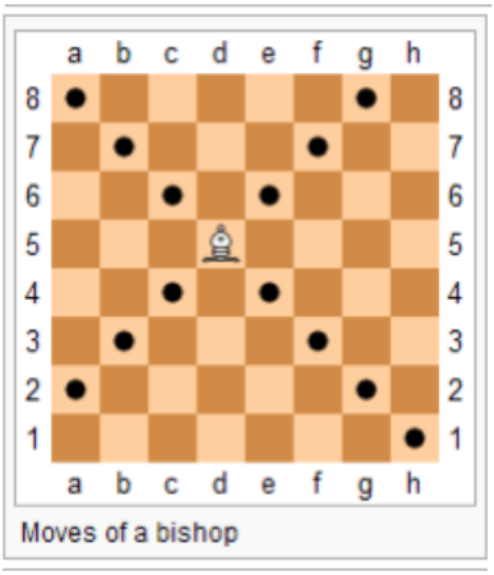
Moves of a king

- The rook can move any number of squares along any rank or file, but may not leap over other pieces.



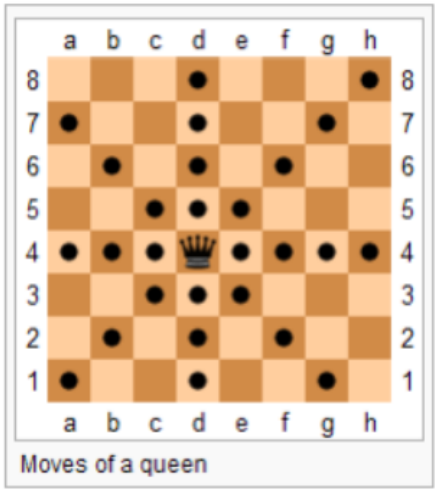
Moves of a rook

- The bishop can move any number of squares diagonally, but may not leap over other pieces.



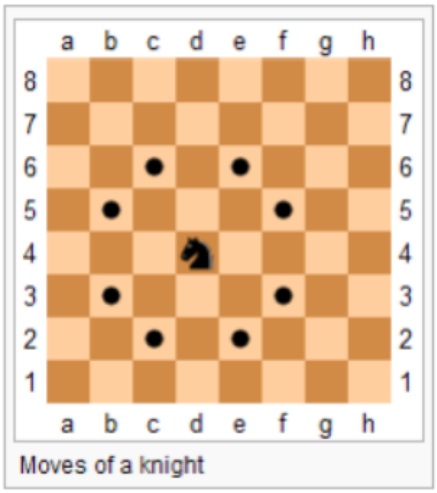
Moves of a bishop

- The queen combines the power of the rook and bishop and can move any number of squares along rank, file, or diagonal, but it may not leap over other pieces.



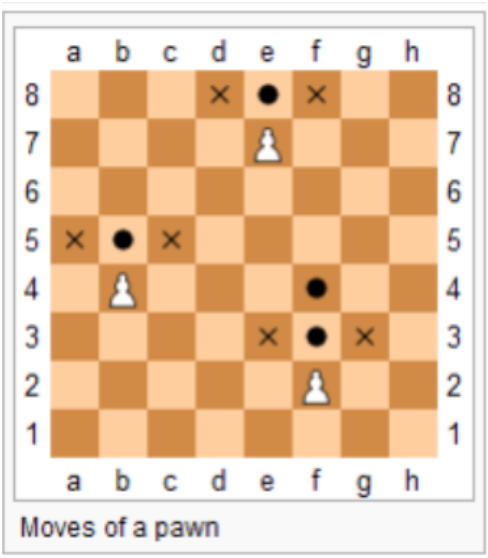
Moves of a queen

- The knight moves to any of the closest squares that are not on the same rank, file, or diagonal, thus the move forms an "L"-shape two squares long and one square wide. The knight is the only piece that can leap over other pieces.



Moves of a knight

- The pawn may move forward to the unoccupied square immediately in front of it on the same file, or on its first move it may advance two squares along the same file provided both squares are unoccupied, or it may move to a square occupied by an opponent's piece, which is diagonally in front of it on an adjacent file, capturing that piece.



3 Problem

In this assignment, you are going to develop a chess application that implements a subset of rules of the chess game. Your program will be able to initialize the chess board and move pieces on the board, and also print the board's layout and show possible moves of the given pieces on the board.

When the program is requested to move pieces on the board, it should firstly check whether the piece is allowed to move to the destination square or not. When the program is requested to move pieces on the board, it should check whether the destination square already contains another piece. If there is an opponent piece on the destination square and the program determines that the move is valid, moving piece will capture the piece at the destination square, removing it from play. If there is a friendly piece on the destination square, the move should be considered invalid. The pieces on the board will be represented with letters:

KI: Black King	ki: White King
QU: Black Queen	qu: White Queen
R1, R2: Black Rook	r1, r2: White Rook
B1, B2: Black Bishop	b1, b2: White Bishop
N1, N2: Black Knight	n1, n2: White Knight
P1, P2, P3, P4, P5, P6, P7, P8: Black Pawn	p1, p2, p3, p4, p5, p6, p7, p8: White Pawn

For this assignment, you will be handling pawns different than the actual chess game. White pawns will only be able to move upwards (from 1 to 8) and black pawns will only be able to

move downwards. The two-square starting move of pawns will not be available. Also, there will be one exception case for Knight. If the square is unoccupied, it can move on the same closest diagonal square, too.

Your program should be able to process and respond to 5 different commands, which are defined below. You will take these commands from an input file (input.txt), according to commands, show the output to the console. When the program starts, the chess board should be full of pieces.

COMMAND-1: *initialize*

Arguments: This command does not take any arguments.

Description: This command load the pieces to the board.

Output: OK

COMMAND-2: *showmoves*

Arguments: piece

Description: Lists the possible target positions of the given piece can move. The order of the positions will be from a to h and from 1 to 8 increasing, i.e. a3 will be printed before a5 and a7 will be printed before b2.

Output: position1 position2 position3...

Output on Failure: FAILED

COMMAND-3: *move*

Arguments: piece position

Description: Moves the given piece to the given position. The operation should fail if the move is not valid.

Output on successful operation: OK

Output on Failure: FAILED

COMMAND-4: *print*

Arguments: This command does not take any arguments.

Description: Prints the status of the board to the console.

Output: The output will be 8 lines of 8 characters, where the rows will be 8 to 1 from top to bottom and the columns will be a to h from left to right. Pieces will be represented with their corresponding letters and empty squares will be represented as a single whitespace character.

COMMAND-5: *exit*

Arguments: This command does not take any arguments.

Description: Instructs the program to exit.

Output: This command does not output any information to the console.

4 Read File

In this part, the python code for reading from file has been shared, since you have not yet handled file operations.

```
import sys
f=open(sys.argv[1],"r")
commands=[[line.split()] for line in f.readlines()]
f.close()
```

5 Execution and Test

At the beginning, you have to pose an input file which denotes the initial configuration of the board and the other commands. Here, please note that file name should be given as an argument and at the end, according to the commands in the file, you will write output to the console. A command and all of its arguments will be entered as a single line and there will be one or more space character between the command and arguments. Your program must not output anything other than what the command is expected to output. Outputs that are different than what's specified will cause you a grade loss. Below is an example session with inputs and outputs:

```
move p1 a3
move P4 d6
move n1 a2
move B1 e6
print
showmoves B1
showmoves r1
move p3 c3
move QU d7
print
initialize
exit
```

Figure 1: An example of input.txt

```
>move p1 a3
OK
>move P4 d6
OK
>move n1 a2
OK
>move B1 e6
OK
>print
```

```
R1 N1    QU KI B2 N2 R2
P1 P2 P3    P5 P6 P7 P8
         P4 B1



p1
n1 p2 p3 p4 p5 p6 p7 p8
r1    b1 qu ki b2 n2 r2
```

```
>showmoves B1
a2 b3 c4 d5 f5 g4 h3
>showmoves r1
b1
>move p3 c3
OK
>move QU d7
OK
>print
```

```
R1 N1       KI B2 N2 R2
P1 P2 P3 QU P5 P6 P7 P8
         P4 B1



p1    p3
n1 p2    p4 p5 p6 p7 p8
r1    b1 qu ki b2 n2 r2
```

```
>initialize
```

```
R1 N1 B1 QU KI B2 N2 R2
P1 P2 P3 P4 P5 P6 P7 P8




p1 p2 p3 p4 p5 p6 p7 p8
r1 n1 b1 qu ki b2 n2 r2
```

```
>exit
```

Figure 2: An example of output screen

6 Grading Policy

Task	Point
Compiled	5
Command1	15
Command2	25
Command3	25
Command4	25
Command5	5

7 Important Notes

- Do not miss the submission deadline.
- Compile your code on dev.cs.hacettepe.edu.tr before submitting your work to make sure it compiles without any problems on our server.
- You do not need to use Numpy library.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating. You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza. You cannot share algorithms or source code. All work must be individual! Assignments will be checked for similarity, and there will be serious consequences if plagiarism is detected.
- *Click here to accept your Programming Assignment 3 for 1 day late submission.* (It will be degraded over 90 points)
- *Click here to accept your Programming Assignment 3 for 2 days late submission.* (It will be degraded over 80 points)
- You may assume that the input file will be given as command-line arguments, so to execute your code on dev use the following command in your terminal:
  python3 assignment3.py input.txt

- You must submit your work with the file as stated below:

  assignment3.py