**AP Computer Science – Final Exam Free Response**

There are six different pieces on a chess board. Each piece has certain rules that control how that piece can be moved. Your task on this quiz is to write code to explain where certain pieces can move on the board. Your code does need to factor in its location on the board, as well as the location of other pieces, to make sure it stays within the boundaries of the board and that it doesn’t move through its own pieces of its own color.

The goal is to build a boolean array that holds the valid move locations for a given chess piece on the board. A value of "true" indicates a possible move location on the board. You will write a method for three piece types. The method will receive the row and column values as integers as well as a 2D char array, which holds the current location of all chess pieces on the board. You can assume that the 2D boolean array is reset to false before each method call.

The following code are used to represent the different chess pieces:

White pieces (All use capital letters)

* R – white rook
* H – white horse(knight)
* B – white bishop
* Q – white queen
* K – white king
* P – white pawn

Black pieces (all use lower case letters)

* r – black rook
* h – black horse(knight)
* b – black bishop
* q – black queen
* k – black king
* p – black pawn

**Chess rules that must be followed:**

1. Each chess piece has its own rules for movement that must be followed.
2. A chess piece cannot move outside the board.
3. Except for the knights, a chess piece cannot move through a piece of its own color, nor occupy the same location on the board as a piece of its own color.
4. If your king is in *check* (attackable by your opponent), you can only make a move that takes your king out of check. (Either move the king or attack your opponent’s piece that threatens your king). Otherwise, on his turn, he will take your king and the game is over.
5. You cannot move through an opponent’s piece, but you can move on top of it and remove it from the board, though rule # 1 does have to be followed.

**Chess Piece Movement Rules**

**Pawn** - A pawn can only move forward one space, except on its first turn. On its first turn, a black pawn can move forward two spaces. It can never move backwards (down). When attacking, it can only attack pieces that are one spot away diagonally above it. It cannot attack a piece that is directly above it. A pawn cannot move diagonally, except to attack an opponent’s piece.

* A black pawn can only move “up” the board.
* A white pawn can only move “down” the board.

**Rook (Castle)** - A rook can move to any location on the board, as long as it stays in the same row or column. (up, down, left or right. It cannot move through piece of its own color.

**Knight (Horse)** - A knight's move is different than the other pieces. A knight moves in an "L" pattern. It moves three spaces, either two vertical moves and one horizontal, or two horizontal and 1 vertical. It can do this pattern in any direction, as long as it remains on the board. For example, the black knight in (7, 1) could move to (5, 0) or (5, 2) or (6, 3). A knight in the middle of the board will have eight possible move locations. A knight ***CAN*** move through pieces of its own color or his opponents color, as long as he lands on either an unoccupied space or on a space with an opponent’s piece.

**Bishop** - A bishop can only move diagonally, but he can move as many spaces as he wants in one diagonally direction. A bishop cannot move through pieces of its own color. A bishop will always stay on a space of the same color throughout the game.

**Queen** – A queen is a combination of knight and bishop. A queen ***can*** move any number of spaces in a horizontal, vertical or diagonal direction. A queen does have the same restriction of not being able to move through pieces of its own color.

**King** – A moves exactly like the queen, except it can only move 1 space. A king ***cannot*** move into “check”, meaning a space that is attackable by your opponent.

On your final exam, you will be asked to write methods that will control how certain pieces are able to move around the board. You will also be provided with some helper methods and may be asked to write a helper method as part of the exam.

Possible methods:

**Piece Movement Methods**

findBlackPawnMoves() – This method will receive the location of a black pawn (row, col) and the board, which is a 2D array of characters. This method should return a 2D Boolean array filled with the valid move locations for that black pawn.

findWhitePawnMoves() findRookMoves() findBishopMoves()

findQueenMoves() findKingMoves() findKnightMoves()

**Helper Methods**

isAttackable() – This method receives a location on the board (row, col), the 2D array of characters representing the board and the color of the person whose turn it is (String).. The method will return true if the opponent can attack that spot on the board, false if he/she cannot.

isCheck() – This method receives only a String for the color of the player whose turn it is. The method will return true if his King is in check, false if he is not. A king is in check if he is able to be attacked by his opponent.

isCheckmate() – This method receives the color of the player whose turn it is. It will return true if his king is currently attackable and all spaces that he could move to are also attackable by the opponent.

isStalemate() - This method receives the color of the player whose turn it is. It will return true if his king is currently not attackable and all spaces that he could move to are also attackable by the opponent and he has no other pieces on the board. (King is last remaining piece)

countBlackPieces(), countWhitePieces() – Receives the board and counts the number of black (or white) pieces remaining on the board and returns that integer value.

findAttackingPiece() – This method receives the color of the player whose turn it is. It will locate and return the char representing the piece on the board that is currently attacking your king.

Other smaller methods may be provided to you to help.