Assignment 3: Search Part 3
"Elegant!" "That's because it's simple."

Summary

For the next two assignments you will be working on developing an AI player for a chess variant called Two-player Wildebeest (2PW).

For this assignment, you will be implementing a program that computes the entire set of valid moves for a board in 2PW. Unlike previous assignments, this is a time consuming and somewhat tedious programming exercise. The second part of this task (Assignment 4) is where the real fun begins.

This will form the basis for Assignment 4 in which you will write a program that plays 2PW. For this assignment, your program will be evaluated based on the number of correct moves that it figures out (minus the number of any incorrect moves). For Assignment 4, your program will be evaluated by letting it compete with other students' programs.

Note: the rules below are subject to change between A3 and A4, and during the evaluation of A4, to ensure a fair balance between white and black, to prevent some pieces from dominating game-play (overly powerful classes may get NERFed), to result in more interesting game play, and for clarification of the rules. An objective of any rule change will be to minimize the impact of having to rewrite code.

Two-player Wildebeest (2PW) Rules (Version 2.0)

Two-player Wildebeest follows the same basic rules as chess (which can be found at http://www.uschess.org/content/view/7324) with the following variations:

Board

The board consists of an 11x11 grid. Pieces are placed in the centres of the 121 squares. The board consists of 11 rows (numbered 0 to 10) and 11 columns (numbered 0 to 10).

Special Squares

There are five special squares on the board.

Jet Pack (#)– The Jet Pack is a square at the middle of the board.

- o If a King lands on the Jeb Pack it becomes a King with a Jet Pack (W/w) (see below).

Transporter Pad (*) – There are 4 transporter pads located in fourth and eighth rows (row 3, 7) and the second and tenth columns (column 1, 9).

Pieces

The following pieces are used:

- **Serpent (S/s)** – the serpent is a piece that moves like a King.
  - o The Serpent cannot move into a square occupied by an opposing

piece (and cannot capture it).
    - The Serpent has the ability to poison any opposing, biological, adjacent piece (8-square). Any time it is adjacent to one ore more opposing biological, pieces, the opposing, biological pieces are ***all*** removed from the board (friendly pieces are unaffected).
    - If the Serpent ever poisons a piece that is also adjacent (8-square) to an opposing Old Woman, then the Serpent is captured (removed from the board) and the Old Woman piece is replaced by the Grand Empress piece. All opposing pieces next to the Serpent's position are still removed.
    - The Serpent can poison an opposing Old Woman (if it is directly adjacent). The serpent can poison a gorilla.
    - The Golf Cart and Time Machine, cannot be poisoned (and cannot be used to convert the old woman into the Grand Empress).
- **Old Woman (O/o)–** is a piece that moves and captures identically to a King.
    - The Old Woman can be converted into a Grand Empress by the poisoning of an adjacent friendly piece see the description of Serpent.
- **Grand Empress** (E/e)– is a piece that moves and captures like a knight, queen and serpent (simultaneously poisoning all adjacent enemy pieces).
    - It is not initially found on the board and is only created when the serpent poisons a piece next to the Old Woman
- **Prince Joey (J/j)–** is a piece that moves and captures identically to a King.
    - After each turn is completed, Prince Joey may explode (see below).
- **Catapult (C/c) –** is a piece that moves like a king (it cannot capture).
    - In addition, it can fling a friendly adjacent piece (8-square) from the piece's starting position to a location that is in the same direction as the direction from the adjacent piece to the catapult.
    - The "flung" piece can move any distance. It can land on an empty square or on an opposing piece (in which case the opposing piece is captured).
    - There is one exception: a "flung" piece cannot capture a King or Gorilla.
    - A Gorilla can be flung onto an enemy piece and thereby capture it.
    - A catapult can fling a friendly piece onto a transporter pad.
    - You cannot fling a time machine.
    - A catapult can fling a paralyzed piece unless the catapult is paralyzed.
    - The flinging operations counts as a move (the catapult cannot move its location and fling a piece in the same turn).
- **Gorilla (G/g)–** the Gorilla is a piece that moves like a king.
    - It does not capture (unless flung), nor can it ***be*** captured (but it can be poisoned).

    - It does, however, have the ability to push a piece (as long as that piece is not also a Gorilla). It this case, the Gorilla moves into an adjacent square (8-square) occupied by a friendly or opposing piece and that friendly or opposing piece also moves a single square in the same direction as the Gorilla. If the destination square of the pushed piece is occupied but not by another Gorilla, then the occupant of the adjacent square is captured (even if it is a friendly piece).
    - The Gorilla cannot push any piece off the board.
- **Golf Cart (X/x)**– the Golf Cart can only move and capture left and right a single space in the bottom and top rows.
    - If the golf cart is charged after a turn, it additionally moves up or down the column (see below).
- **Time Machine (H/h)**– if a pawn makes it to the opposite end of the board it is automatically converted into a Time Machine.
    - As long as the Time Machine is on the board, the Golf Cart is charged (regardless of the position of opposing Pawns).
    - The Time Machine cannot move or capture.
- **Beekeeper (Z/z)**– the Beekeeper moves and captures like a King.
    - In addition, by releasing a swarm it paralyzes any opposing adjacent pieces (8-sqaure) and prevents them from moving.
    - Assume that the swarm is released before game starts (i.e. perma-swarm).
    - A piece can move through the swarmed space as long as it doesn't end in the swarmed space.
    - The Golf Cart, and Time Machine, cannot be paralyzed.
- **King (K/k)** - moves and captures like in regular chess (excl. castling move).
- **King with a Jet Pack (W/w)** – a King with a Jet Pack moves like a Bishop (B/b).
- **Pawn (P/p)** – moves and captures like a regular chess pawn (except that the *en passant* capture move is not allowed).
- **Bishop (B/b)** – moves and captures like in regular chess.
- **Knight (N/n)** – moves and captures like in regular chess.
- **Rook (R/r)** – moves and captures like in regular chess.

- The parenthesized letters after the piece name indicate how the piece is represented on the board. Upper case letters represent white, lower case letters black pieces.

After Effects

Once a player has completed their move, the following actions are performed (in this order).

1. If any pawn (of either team) occupies the row, 5 spaces forward of its staring

position (the middle row of the board), the opposing player's golf cart becomes fully charged and moves along its own column to the opposite end of the board.
    - As it moves from one end of the column to the other, it captures all pieces in the column (even friendly pieces and Gorillas).
    - Once it reaches the opposite end of the column it stops.
    - If both golf carts are in the same column and are both charged, everything in that column (including both golf carts) is removed.
2. If there is a Time Machine on the board the same player's golf cart is fully charged and moves along its column as described above.
3. All piece resting on a transporter pad (*) are moved to their diagonally opposite locations.
    - Two pieces can exchange locations in this step.
4. If at any point in time, the total number of pieces (including Prince Joey) in Prince Joey's row is evenly divisible by 5, prince Joey explodes and he, and any adjacent (8-square) pieces (friendly or enemy are removed from the board).
    a. If there are two Prince Joey's in the same row, the determination of whether each Prince Joey explodes should be performed once, prior to the explosion(s) and removal of pieces.

Winning
- "Checkmate" occurs when the opposing King is captured.
- If both Kings are captured at the same time during the After Effects, the game is a draw.

Promotion
- If a pawn makes it to the end of the board it is converted to a Time Machine. Pawns cannot be promoted to any other piece type.
- If a king makes it to the middle of the board it is converted to a King with a Jet Pack.

Operation
Your program will operate by reading a board file from standard input ("a3 < board.txt"). Make sure that your program can be executed by calling "a3". If necessary you can do this by either writing a script to call your interpreter or virtual machine, or by using the "#!/path/interpreter" UNIX convention, or by providing an executable.

As output it will produce a number of output files named "board.000", "board.001", "board.002", etc. These boards ***must*** be saved in the same directory from which the program is launched (don't save them in a subdirectory or a hard-coded path). Each file will contain a valid board that could occur on the following turn. Your program should generate files for all possible moves and valid boards.

The input file will consist of 15 lines. The first line indicates whose turn it is (white or black). The next 11 lines represent the chessboard. The following 3 lines are reserved for future use – each will contain a single integer.

The initial board is represented as:

```
W
RNZBOKXBCNR
GPPSPJPPPPG
...........
.*.......*.
...........
.....#.....
...........
.*.......*.
...........
gppspjppppg
rnzbokxbcnr
0
60000
0
```

Each line in the board file will be terminated by a single new-line (unix-style) character. Thus, the first line will consist of 2 characters, the next eleven lines will consist of 12 characters each, and the last 3 lines will be comprised of however many characters are required to represent the integers, plus one new-line per line.

An example board file is found in **`board.txt`** in the repo.

Your overall performance will be based on how many correct and how few incorrect successor boards you generate. Each board that you generate must have the same format as indicated above. You should transcribe the 3 final integers without modification, and convert the first character from black to white, and vice versa. In addition to the initial board above, your program will be subjected to a number of other test boards (that will not be provided to you).

For this assignment you are permitted and encouraged to share input game boards (test cases) with your classmates. You are also permitted to share the ***number*** of next board states (but not the actual board) that your program generates.

Testing
An example solution is provided at:

http://cis3700.socs.uoguelph.ca/

You can upload boards to this web-site and it will generate the intended outputs.

Grading Notes
If the rules, above, can be reasonably interpreted in more than one way, then either interpretation will receive full marks (you won't lose marks if my assignment description is ambiguous). If the output of the example solutions differs from the rule description, above, then either interpretation will receive full marks (you won't lose marks if my assignment description is ambiguous or my code is wrong).

Submission Instructions
Submit via git, as usual.