cGame Documentation

chessGame (white: Callable[[chess.Board, float], chess.Move], black: Callable[[chess.Board, float], chess.Move], board: chess.Board) -> (chess.pgn.Game, chess.Outcome)

inputs :

White and Black functions should take in (board: chess.Board, clockTime: float)

> **inputs :**
>
> board: chess.Board - board object from the chess library
> clockTime: float - float representing the amount of time left on the bot's game clock
>
> **Outputs :**
>
> chess.Move - move object from the chess library

board: chess.Board - object representing the current chess board that can be created using the chess library

Outputs :

chess.pgn.Game - Game object in the chess library that represents an entire chess game played

chess.Outcome - Another object from the chess library that represents the outcome of a game

Ex :

```
def bot1(board: chess.Board, clockTime: float):
    return random.choice(list(board.legal_moves))

board = chess.Board("8/8/3k4/8/8/8/5K2/8 w - - 0 1")

cGame.chessGame (bot1, bot1, board)
```

chessMatch (bot1: Callable[[chess.Board, float], chess.Move], bot2: Callable[[chess.Board, float], chess.Move], boardPositions: List[str]) -> (int, int, int, List[chess.pgn.Game])

inputs :

White and Black functions should take in (board: chess.Board, clockTime: float)

> **inputs :**
>
> board: chess.Board - board object from the chess library
> clockTime: float - float representing the amount of time left on the bot's game clock
>
> **Outputs :**
>
> chess.Move - move object from the chess library

boardPositions: List[str] - This will be a list of all starting positions represented by using FEN's such as "8/8/3k4/8/8/8/5K2/8 w - - 0 1". Whoever is first to play will be played by bot1 or the first bot listed.

Outputs :

bot1Score: int - the number of times bot1 won
Draws: int - the number of times the bots tied
bot2Score: int - the number of times bot2 won
Games: List[chess.pgn.Game] - list of games played represented as PGN's using the chess.pgn.Game object from the chess library. It is a list as long as the amount of positions with each element in the list being another list that contains the games played as both black and white for a specific position. For example [[position1 w, position1 b], [position2 w, position2 b]]

Ex :

```
def bot1(board: chess.Board, clockTime: float):
    return random.choice(list(board.legal_moves))

positions = ["rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0
1","r1bqkb1r/ppp2ppp/2n2n2/3pp1N1/2B1P3/8/PPPP1PPP/RNBQK2R w KQkq - 0
1","r1bqkbnr/pp1ppppp/8/2p5/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0
1","rnbq1bnr/ppppkppp/8/4p3/4P3/8/PPPPKPPP/RNBQ1BNR w - - 0
1","rnbq1rk1/pppppppbp/5np1/8/8/5NP1/PPPPPPPBP/RNBQ1RK1 w - - 0
1","rnbqkbnr/pppp1ppp/4p3/8/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0
```

1","rnbqkbnr/ppp1pppp/8/3p4/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0
1","rnbqkbnr/pp1ppppp/2p5/8/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0 1"]

bot1Score, draw, bot2Score, games = cGame.chessMatch(bot1,bot1,positions)

game = games[1][3] # [1] represents bot2 is white while [3] means it is the fourth position

downloadGame (game: chess.pgn.Game, filename: str)

inputs :

game: chess.pgn.Game - Game object in the chess library that represents an entire chess game played

filename: str - string represents the file name as well as path that you want the game PGN to be saved to

outputs :

None

Ex :

bot1Score, draw, bot2Score, games = cGame.chessMatch(bot1,bot1,positions)

game = games[1][3] # [1] represents bot2 is white while [3] means it is the fourth position

cGame.downloadGame(game, r"filelocation")