**from utilities import place\_random, print\_board**

**DEV\_MODE = False**

**def main(game\_board: [[int, ], ]) -> [[int, ], ]:**

**"""**

**2048 main function, runs a game of 2048 in the console.**

**Uses the following keys:**

**w - shift up**

**a - shift left**

**s - shift down**

**d - shift right**

**q - ends the game and returns control of the console**

**:param game\_board: a 4x4 2D list of integers representing a game of 2048**

**:return: returns the ending game board**

**"""**

**move = {"a": left(game\_board),**

**"d": right(game\_board),**

**"w": up(game\_board),**

**"s": down(game\_board)}**

**# Initialize board's first cell**

**if DEV\_MODE:**

**# This line of code handles the input of the develop mode.**

**column, row, value = (int(i) for i in input("column,row,value:").split(','))**

**else:**

**#generate a random piece and location using the place\_random function**

**random\_piece = place\_random(game\_board)**

**game\_board[random\_piece["row"]][random\_piece["column"]] = random\_piece["value"]**

**random\_piece = place\_random(game\_board)**

**game\_board[random\_piece["row"]][random\_piece["column"]] = random\_piece["value"]**

**# place the piece at the specified location**

**print\_board(game\_board)**

**pass**

**# Initialize game state trackers**

**# Game Loop**

**while not game\_over(game\_board):**

**# Reset user input variable**

**user\_move = input("Please make a move: ").lower()**

**while user\_move not in ["w", "a", "s", "d", "q"]:**

**user\_move = input("Invalid move!!\nPlease make another move: ").lower()**

**if user\_move == "q":**

**break**

**elif user\_move == "a":**

**game\_board = left(game\_board)**

**elif user\_move == "d":**

**game\_board = right(game\_board)**

**elif user\_move == "w":**

**game\_board = up(game\_board)**

**elif user\_move == "s":**

**game\_board = down(game\_board)**

**# Take computer's turn**

**# place a random piece on the board**

**# check to see if the game is over using the game\_over function**

**try:**

**random\_piece = place\_random(game\_board)**

**game\_board[random\_piece["row"]][random\_piece["column"]] = random\_piece["value"]**

**except Exception:**

**break**

**# TODO: Show updated board using the print\_board function**

**print\_board(game\_board)**

**# TODO: Take user's turn**

**# Take input until the user's move is a valid key**

**# if the user quits the game, print Goodbye and stop the Game Loop**

**# Execute the user's move**

**# Check if the user wins**

**print("Game Over")**

**return game\_board**

**def game\_over(game\_board: [[int, ], ]) -> bool:**

**"""**

**Query the provided board's game state.**

**:param game\_board: a 4x4 2D list of integers representing a game of 2048**

**:return: Boolean indicating if the game is over (True) or not (False)**

**"""**

**# TODO: Loop over the board and determine if the game is over**

**if any(2048 in row for row in game\_board):**

**return true**

**elif not h\_move\_possible(game\_board) and not v\_move\_possible(game\_board):**

**return truw**

**return False # TODO: Don't always return false**

**def stack(game\_board):**

**updated\_board = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]**

**for i in range(4):**

**non\_zero = 0**

**for j in range(4):**

**if game\_board[i][j] != 0:**

**updated\_board[i][non\_zero] = game\_board[i][j]**

**non\_zero += 1**

**return updated\_board**

**def combine(game\_board):**

**for i in range(4):**

**for j in range(3):**

**if game\_board[i][j] != 0 and game\_board[i][j] == game\_board[i][j+1]:**

**game\_board[i][j] \*= 2**

**game\_board[i][j+1] = 0**

**return game\_board**

**def reverse(game\_board):**

**reverse\_board = []**

**for i in range(4):**

**reverse\_board.append([])**

**for j in range(4):**

**reverse\_board[i].append(game\_board[i][3-j])**

**return reverse\_board**

**def transpose(game\_board):**

**tranpose\_board = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]**

**for i in range(4):**

**for j in range(4):**

**tranpose\_board[i][j] = game\_board[j][i]**

**return tranpose\_board**

**def left(game\_board):**

**return stack(combine(stack(game\_board)))**

**def right(game\_board):**

**return reverse(stack(combine(stack(reverse(game\_board)))))**

**def up(game\_board):**

**return transpose(stack(combine((stack(transpose(game\_board))))))**

**def down(game\_board):**

**return transpose((reverse((stack(combine((stack((reverse((transpose(game\_board))))))))))))**

**def h\_move\_possible(game\_board):**

**for i in range(4):**

**for j in range(3):**

**if game\_board[i][j] == game\_board[i][j+1]:**

**return True**

**return False**

**def v\_move\_possible(game\_board):**

**for i in range(3):**

**for j in range(4):**

**if game\_board[i][j] == game\_board[i+1][j]:**

**return True**

**return False**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main([[0, 0, 0, 0],**

**[0, 0, 0, 0],**

**[0, 0, 0, 0],**

**[0, 0, 0, 0]])**