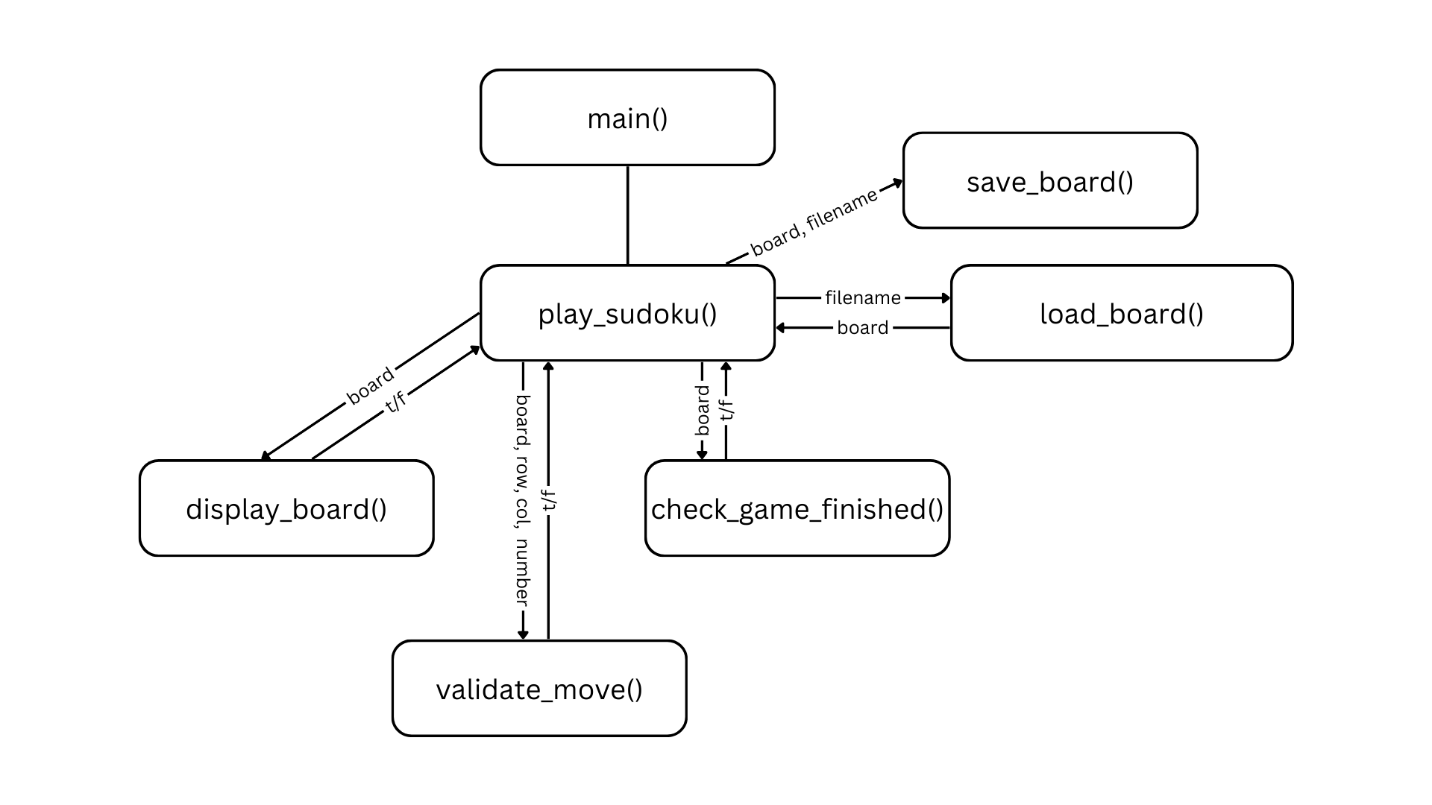
Structure chart:

DFD:

+-------------------+ +------------------------+

| User | <------> | Sudoku Program |

+-------------------+ +------------------------+

| | | |

| - Filename input | | - Display board |

| - Row/Col/Number | | - Validate user input |

| - Save/Exit choice | | - Save board to file |

+-------------------+ +------------------------+

+-------------------+ +---------------------+ +----------------------+

| Load/Save Board |<-------->| Validate User Move |<-------->| Display Board |

+-------------------+ +---------------------+ +----------------------+

| | | | | |

| - Load board file | | - Check Row | | - Show current board |

| - Save board file | | - Check Column | | |

+-------------------+ | - Check 3x3 subgrid | +----------------------+

+---------------------+ |

+----------------------+

| Check Game Finish |

+----------------------+

----------------------------------------------------------------------

Pseudocode: convert\_coordinate(coordinate)

FUNCTION convert\_coordinate(coordinate):

columns 🡨 {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6,

'H': 7, 'I': 8}

letter 🡨 coordinate[0]

number 🡨 coordinate[1]

column 🡨 columns[letter]

row 🡨 int(number) - 1

RETURN (row, column)

END FUNCTION

----------------------------------------------------------------------

Pseudocode (AI): is\_valid\_move(board, row, column, number)

Prompt: Write a pseudocode function that determines the validity of a given number in a square.

Results:

Function is\_valid\_move(board, row, column, number):

// Input: 2D array (board), row and column (integers), and number to place (integer)

// Output: Boolean (True if the move is valid, False otherwise)

// Step 1: Check if the number is already in the row

For i = 0 to 8:

If board[row][i] == number:

Return False // Invalid move: number already exists in row

// Step 2: Check if the number is already in the column

For i = 0 to 8:

If board[i][column] == number:

Return False // Invalid move: number already exists in column

// Step 3: Check if the number is already in the 3x3 subgrid

// Determine the starting row and column of the 3x3 grid

grid\_row\_start = (row // 3) \* 3

grid\_col\_start = (column // 3) \* 3

// Iterate over the 3x3 grid

For i = 0 to 2:

For j = 0 to 2:

If board[grid\_row\_start + i][grid\_col\_start + j] == number:

Return False // Invalid move: number already exists in 3x3 grid

// Step 4: If no conflicts are found, the move is valid

Return True

Changes: I am going to change function names, variable names, and calls to other functions to make this fit in to my program.