PROJECT 11 : SUDOKU

Program Description

The program will prompt the user for the filename of the game he or she is currently working on and display the board on the screen. The user will then be allowed to interact with the game by selecting which square he or she wishes to change. While the program will not solve the game for the user, it will ensure that the user has not selected an invalid number. If the user types 'S' in the prompt, then the program will show the user the possible valid numbers for a given square. When the user is finished, then the program will save the board to a given filename and exit.

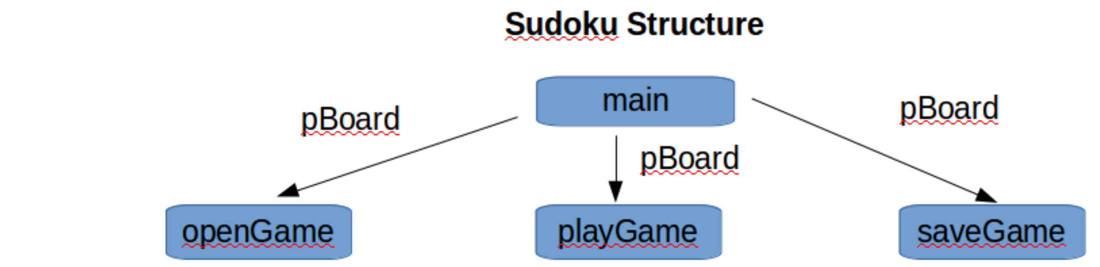
Algorithms

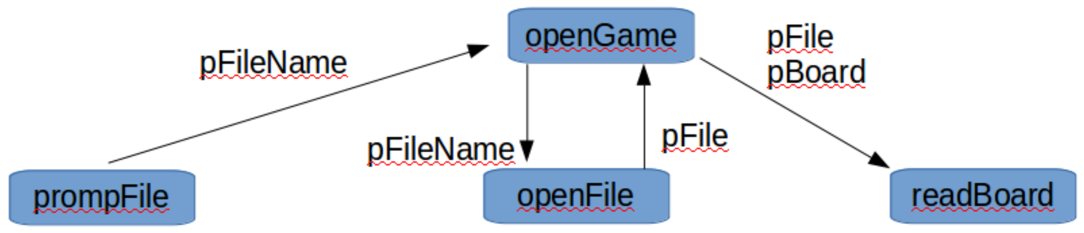
computeValues(int row, int col)

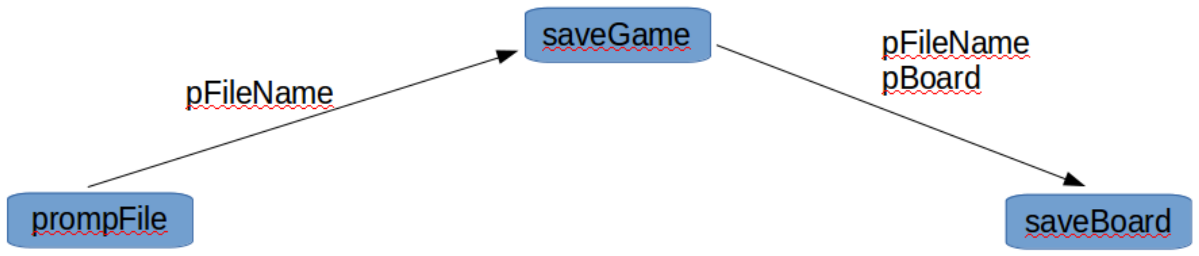
```
// set size of not possibles for each row , col and box
SET rowCount , colCount, boxCount to 1

// create an array for row, col and box NOT possible values
SET rowNotPossible TO createArray
SET colNotPossible TO createArray
SET *boxNotPossible TO createArray

// create array and return pointer
Int *createArray(size)

        SET *p to new array [size +1]
        RETURN p

// takes an array , and the current size and expands the array by 1
Int expandArray(*array, size)
    SET tempArray IS new array +1
    FOR 0 to size
        tempArray[i] IS array[i]
    DELETE array
    array IS tempArray

    RETURN *array
```

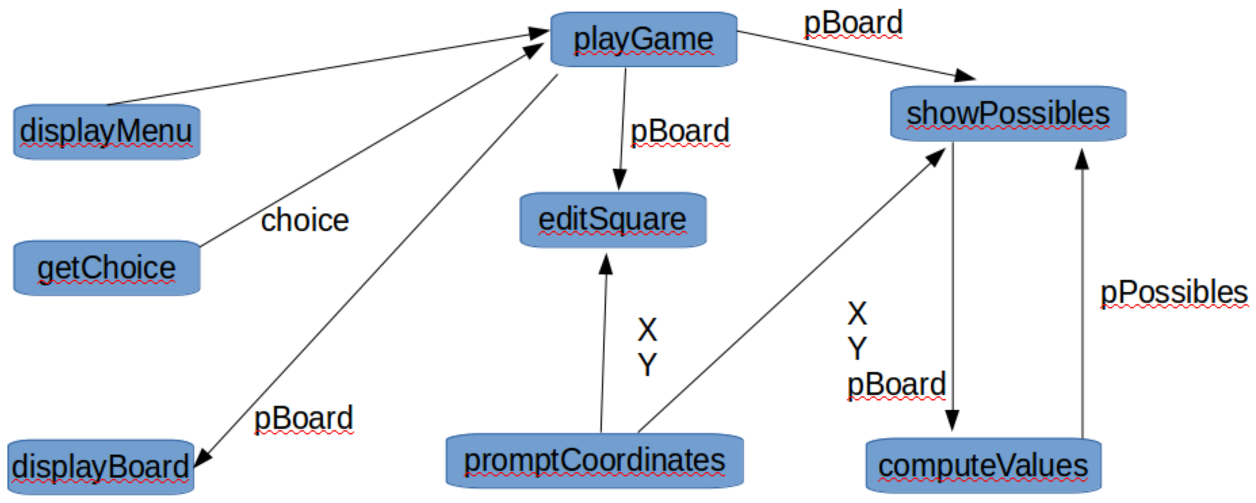
```
FOR 0 to 8i++ // loop through row / columns
    // check rows if not 0
    IF board[row][i] IS != 0
        rowNotPossible[rowCount] IS board[row][i]
            expandArray(rowNotPossible, rowCount)
            rowCount++

    //check col if not 0
    IF board[col][i] IS != 0
        colNotPossible[colCount] IS board[col][i]
            expandArray(colNotPossible, colCount)
            colCount++


    // this is where the real magic happens…. Find out the beginning of the box
SWITCH row % 3
    CASE 0
        rowBlockStart IS row
    CASE 1
        rowBlockStart IS (row – 1)
    CASE 2
        rowBlockStart IS (row – 2)

SWITCH col % 3
    CASE 0
        colBlockStart IS col
    CASE 1
        colBlockStart IS (col – 1)
    CASE 2
        colBlockStart IS (col – 2)

// loop through box and see if its not 0
FOR rowBlockStart TO (rowBlockStart + 2)
    FOR colBlockStart TO (colBlockStart + 2)
        IF board [rowBlockStart] [colBlockStart] != 0
            boxNotPossible[boxCount] IS board [rowBlockStart] [colBlockStart]
            boxCount++
    valueCount IS 0
    FOR 1 to 9 i++

        FOR 0 to boxCount j++
            boxPossible IS TRUE
            IF i == boxNotPossible[j]
                boxPossible IS FALSE

        FOR 0 to rowCount j++
            rowPossible IS TRUE
            IF i == rowNotPossible[j]
                rowPossible IS FALSE

        FOR 0 to colCount j++
            colPossible IS TRUE
            IF i == colNotPossible[j]
                colPossible IS FALSE

        IF boxPossible && rowPossible && colPossible
            possibleValues[valueCount]
            valueCount++

RETURN *possibleValues


interact()

DO
    exit IS false
    SWITCH getChoice()
        CASE ?
            DisplayMenu()
        CASE D
            displayBoard()
        CASE E
            editSquare()
        CASE S
            showPossibles()
        CASE Q
            exit IS true
WHILE !exit
```

Structure Chart









Grading

The grading criteria are:

	Exceptional 100%	Good 90%	Acceptable 70%	Developing 50%	Missing 0%
Structure Chart 40%	Design is elegant and well thought out	Design will solve the problem	The structure chart is incomplete or has a serious defect making the solution unworkable	Structure chart not showing the connection between the functions, name of the functions, or the data passed between the functions	No structure chart
computeValues(): Rules are enforced 30%	The most efficient solution has been found	All the rules are correctly represented in the pseudocode	A bug exists or not all the rules are enforced	Pseudocode insufficient quality to help in the design process	No rule logic in the computeValues()function
computeValues(): List possible values 10%	Design is elegant and well thought out	All possible values will be listed for a given square on the board	A bug exists in the algorithm or does not solve all the problems	Pseudocode not detailed enough to tell if the problem has been solved	No display logic in the computeValues()function
interact() 20%	The most elegant solution has been found	The function will allow the user to interact with the game	The design fails to do one of the following: · Select an option · Execute the option specified · Continue looping until the user is finished	Pseudocode insufficient quality to help in the design process	No pseudocode for the interact function

You are required to attach the **above rubric** to your design document. Please self-grade.