Assignment 2

Problem 1: Test cases for Sudoku class

1. **Input Validation Test Cases:**
   a. **Public Sudoku (int size)**
      i. String passed
         1. Compile time error
      ii. Long/float data type passed (both positive or negative)
         1. Compile time error
      iii. char data type passed
         1. Accepts the input
      iv. Short data type passed
         1. Accepts the input
      v. Boolean data type passed
         1. Compile time error
      vi. Null passed
         1. Compile time error

   b. **Public boolean setPossibleValues (Stirng values)**
      i. Int data type passed
         1. Compile time error
      ii. Float/double data type passed
         1. Compile time error
      iii. Char data type passed
         1. Compile time error
      iv. Duplicate passed in values string
         1. Returns false
      v. Special characters passed in values (as a unique character)
         1. Returns true
      vi. Object passed
         1. Compile time error
      vii. Null String passed
         1. Returns false
      viii. Empty String passed
         1. Returns false
      ix. String contains a single space char among other unique chars **(considering single space as a unique character)**
         1. Returns true

   c. **Public boolean setCellValue (int x, int y, char letter)**
      i. Negative x & y passed
         1. Returns false
      ii. Letter passed which is not present in the values string
         1. Returns false
      iii. Less parameters passed (like 2/1 parameter passed)
         1. Compile time error
      iv. Float/double data type passed instead of int
         1. Compile time error
      v. Char data type passed to all 3 parameters
         1. Returns true
      vi. Short data type passed instead of int

            1. Returns true
        vii. String passed instead of char data type/ String passed to all 3 parameters
            1. Compile time error
        viii. Int data type passed to all 3 parameters
            1. Compile time error
        ix. Single space char passed to letter
            1. Returns true
        x. Null char passed to letter
            1. Returns false

    d. **Public boolean solve ()**
        i. Parameter passed while calling the method
            1. Compile time error

    e. **Public String toPrintString (char emptyCellLetter)**
        i. String passed instead of char data type
            1. Compile time error
        ii. \n passed
            1. Returns the string representing the current state of the sudoku
        iii. Null char passed
            1. Throws exception
        iv. Int data type passed
            1. Compile time error
        v. Float/double data type passed
            1. Compile time error
        vi. Single space char passed
            1. Returns the string representing the current state of the sudoku

2. **Boundary Tests Cases:**
    a. **Public Sudoku (int size)**
        i. 0 passed
            1. Throws exception
        ii. An int data type greater than the range of int is passed (both positive or negative)
            1. Compile time error
        iii. -1 passed
            1. Throws exception
        iv. 1 passed
            1. Throws exception
        v. 2 passed
            1. Accepts the input and creates a sudoku of 2 x 2

    b. **Public boolean setPossibleValues (Stirng values)**
        i. String half the size of (size^2) is passed
            1. Returns false
        ii. String of (size^2) with unique characters passed
            1. Returns true
        iii. String greater than (size^2) is passed
            1. Returns false
        iv. String lesser than (size^2) is passed
            1. Returns false

    c. **Public boolean setCellValue (int x, int y, char letter)**

i. Set value in outermost row/column (border row/column)
   1. Returns true
ii. Set value in the last cell (size passed to x & y)
   1. Returns true
iii. 1 passed to x & y
   1. Returns true
iv. 0 passed to x & y
   (safely assuming that the grid starts from 1 x 1)
   1. Returns false
v. Negative int data type passed to x & y
   1. Returns false
vi. Called on a cell outside the sudoku (x & y greater than size)
   1. Returns false

d. **Public boolean solve ()**
   i. Solve an empty sudoku
      1. Returns false
   ii. Solve a sudoku with one sub-grid being empty
      1. Returns false
   iii. Solve a sudoku with more than one sub-grid empty
      1. Returns false
   iv. Solve a big sudoku (size being the size of the int data type)
      1. Returns true
   v. Solve a very small sudoku (like a 2 x 2 or 3 x 3)
      1. Returns true

e. **Public String toPrintString (char emptyCellLetter)**
   i. Called on an empty sudoku
      1. Returns string with emptyCellLetter
   ii. Called after solve () method
      1. Solved sudoku string is printed
   iii. Called on an unsolved sudoku
      1. Unsolved sudoku string is printed

3. **Control Flow Test Cases:**
   a. **Public Sudoku (int size)**
      (considering normal flow)
      i. Constructor is called under normal conditions
         1. Instantiates the Sudoku class and creates a sudoku

   b. **Public boolean setPossibleValues (Stirng values)**
      i. Reading the values string twice
         1. Returns false

   c. **Public boolean setCellValue (int x, int y, char letter)**
      i. Trying to set duplicate values in any row
         1. Returns false
      ii. Trying to set duplicate values in any column
         1. Returns false
      iii. Trying to set duplicate values in any sub-grid
         1. Returns false
      iv. Every row contains all unique characters from the values string

            1. Returns true
        v. Every coumn contains all unique characters from the values string
            1. Returns true
        vi. Every sub-grid contains all unique characters from the values string
            1. Returns true
        vii. Called on the same cell again
            1. Returns false

d. **Public boolean solve ()**
    i. Solve when a row/column contains duplicate value
        1. Returns false
    ii. Sudoku maintains the threshold limit for an unsolved sudoku
    (Minimum entries needed to solve the sudoku of **rank n** is **n^2-1** for a unique solution)
        1. Returns true
    iii. Threshold limit of the unsolved sudoku less than required
    (as this could lead to multiple solutions)
    (sudoku of **rank n** with **n^2-2** entries)
        1. Returns true with any of the possible solution
    iv. Solve when the values are more than the minimum threshold values in an unsolved sudoku
    (more than **n^2** entries for a sudoku of **rank n**)
        1. Returns true
    v. Put assert statements before this method to check if the sudoku is not solved
    vi. Put assert statement after this method to check whether the sudoku is solved

e. **Public String toPrintString (char emptyCellLetter)**
    i. Print a partially solved sudoku
        1. Partially solved sudoku string is printed

4. **Data Flow Test Cases:**
    a. Call Sudoku (int size) twice in a row
    b. Call solve () twice in a row
    c. Call toPrintString (char emptyCellLetter) twice in a row
    d. Call toPrintString (char emptyCellLetter) before calling the solve () method
    e. Call toPrintString (char emptyCellLetter) after calling the solve () method
    f. Call setCellValue (int x, int y, char letter) before calling the setPossibleValues (String values)
    g. Call setCellValue (int x, int y, char letter) after calling the setPossibleValues (String values)
    h. Call solve () before calling the setCellValue (int x, int y, char letter)
    i. Call solve () before calling the setPossibleValues (String values)