Recursion
Assignment 4

Note : Use concepts of Recursion for completing assignment. Also write JUnit test cases for below problems.

Question 1

Design a class to implement following mathematical problems.

- L.C.M. of two numbers x and y. Receive x and y as method parameters and return computed value.
- H.C.F of two numbers x and y using Euclid’s algorithm. Receive x and y as method parameters and return computed value.

Assume x and y as positive integers.

Question 2

Design a class “Search” to search for an element in an array using following strategy

- Linear Search
- Binary Search

Each of the above methods will receive an array and an element to be search in array as input. It will return the index where this element occurs in array. Assume array to be sorted for Binary search.

Linear Search : Linear search or sequential search is a method for finding a target value within an array. It sequentially checks each element of the array for the target value until a match is found or until all the elements have been searched.

Binary Search : Binary search is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array. If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found.

Question 3

Implement N Queens Problem using Recursion

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. (Thus, a solution requires that no two queens share the same row, column, or diagonal)

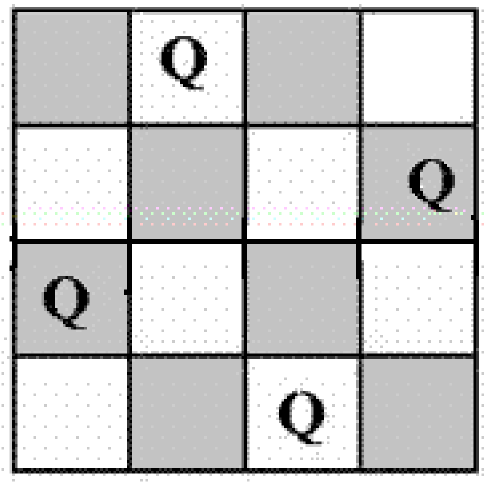
```
Boolean  nQueen(int[][], int startRow, int dimensionOfMatrix);
```

```
int [][] board = new int [][] {
        { 0,  0,  0,  0},
        { 0,  0,  0,  0},
        { 0,  0,  0,  0},
        { 0,  0,  0,  0}

};
```

Boolean result = nQueen(board , 0, 4);
where result should be true and board should have following content.

Following is a solution for 4 Queen problem.



The **expected** output is a binary matrix which has 1s for the blocks where queens are placed. For example following is the output matrix for above 4 queen solution.

```
        { 0,  1,  0,  0}
        { 0,  0,  0,  1}
        { 1,  0,  0,  0}
        { 0,  0,  1,  0}
```

Challenging Problem

Knights-tour-problem:
A **knight's tour** is a sequence of moves of a knight on a chessboard such that the knight visits every square only once. If the knight ends on a square that is one knight's move from the beginning square (so that it could tour the board again immediately, following the same path), the tour is *closed*, otherwise it is *open*.

Implement knights-tour-problem for n*n matrix