```java
package edu.ics211.h09;

/**
 * Test a HexadecimalSudoku solver.
 * Note from khj: Examples 3 & 4 are commented out to save time.
 * @author Biagioni, Edoardo and Cam Moore
 *     date August 5, 2016
 *     bugs none
 */
public class HexadecimalSudokuTest {

  /**
   * Checks the sudoku returning true if all cells are filled. Does not check
   * validity.
   *
   * @return true if all cells are filled, false otherwise.
   */
  private static boolean isFilled(int[][] sudoku) {
    for (int i = 0; i < 16; i++) {
      for (int j = 0; j < 16; j++) {
        if (sudoku[i][j] == -1) {
          return false;
        }
      }
    }
    return true;
  }


  /**
   * Test whether two sudoku are equal. If not, return a new sudoku that is
   * blank where the two sudoku differ.
   *
   * @param sudoku the sudoku to be checked.
   * @param solution the solution checked.
   * @return null if the two match, and otherwise a sudoku with 0 in every cell
   *         that differs.
   */
  private static int[][] sameSudoku(int[][] sudoku, int[][] solution) {
    int[][] result = new int[16][16];
    for (int i = 0; i < 16; i++) {
      for (int j = 0; j < 16; j++) {
        result[i][j] = sudoku[i][j];
      }
    }
    boolean same = true;
    for (int i = 0; i < 16; i++) {
      for (int j = 0; j < 16; j++) {
        if (result[i][j] != solution[i][j]) {
          same = false;
          result[i][j] = -1;
        }
      }
    }
    if (same) {
      return null;
    }
    return result;
  }


  /**
   * Try to solve a sudoku. If a solution is provided, also check against the
   * solution. Print the results.
   *
   * @param name the name of this sudoku.
   * @param sudoku the sudoku to be solved.
   * @param solution the given solution, or null.
   */
  private static void testSudoku(String name, int[][] sudoku, int[][] solution) {
    System.out.println("solving " + name + "\n" + HexadecimalSudoku.toString(sudoku, true));
    if (HexadecimalSudoku.solveSudoku(sudoku)) {
      if (isFilled(sudoku) && HexadecimalSudoku.checkSudoku(sudoku, true)) {
        System.out.println("success!\n" + HexadecimalSudoku.toString(sudoku, true));
        if (solution != null) {
          int[][] diff = sameSudoku(sudoku, solution);
          if (diff != null) {
           System.out.println("given solution:\n" + HexadecimalSudoku.toString(solution,
     true));
           System.out.println("difference between solutions:\n"
             + HexadecimalSudoku.toString(diff, true));
          }
        }
      } else { /* the supposed solution is not a complete or valid sudoku */
        if (!isFilled(sudoku)) {
          System.out.println("sudoku was not completely filled:\n"
            + HexadecimalSudoku.toString(sudoku, false));
        }
        if (!HexadecimalSudoku.checkSudoku(sudoku, false)) {
          System.out.println("sudoku is not a valid solution:\n"
            + HexadecimalSudoku.toString(sudoku, false));
        }
      }
    } else {
      System.out.println("unable to complete sudoku " + name
          + "\n" + HexadecimalSudoku.toString(sudoku, true));
    }
  }


  /**
   * Tests four Sudoku proglems.
   * @param arg command line arguments, ignored.
   */
  public static void main(String[] arg) {

    int[][] example1 = { { 11, 2, 5, -1, 4, -1, 9, -1, 6, 14, -1, 1, -1, 3, -1, -1 },
        { 14, -1, 0, 9, -1, -1, 2, 12, 13, -1, 3, -1, 15, -1, -1, -1 },
        { 1, -1, -1, -1, -1, -1, 7, -1, -1, 9, -1, 2, 11, 5, 14, 0 },
        { 13, 8, -1, -1, 5, -1, -1, 0, -1, -1, 15, -1, -1, 9, -1, 2 },
        { 0, 7, 14, 2, -1, -1, -1, 9, -1, -1, -1, 5, -1, -1, 3, 15 },
        { 3, -1, -1, -1, 10, -1, -1, -1, 2, 4, 13, 15, -1, -1, 6, 11 },
        { 12, -1, 10, 13, -1, -1, -1, -1, 8, -1, -1, -1, 7, -1, 5, 9 },
        { 6, 11, -1, -1, -1, 15, -1, -1, -1, 12, 9, 3, -1, -1, 10, -1 },
        { 2, -1, -1, -1, 3, 7, 11, 4, 5, -1, -1, -1, 0, 13, -1, 8 },
        { 7, 6, 12, 8, -1, -1, -1, -1, 0, 13, -1, 11, 4, -1, -1, -1 },
        { 4, 9, 3, -1, -1, -1, -1, -1, 15, -1, 12, 7, 6, -1, 1, -1 },
        { 10, -1, 11, -1, 15, -1, 12, 1, 3, -1, -1, 14, 9, 7, -1, -1 },
        { 9, -1, 2, -1, 7, 4, 0, -1, -1, -1, 5, -1, -1, 8, 13, -1 },
        { 8, 3, 7, -1, -1, 9, 6, -1, 12, -1, -1, -1, -1, -1, -1, 14 },
        { 15, -1, 4, -1, 12, -1, 8, 10, -1, -1, -1, -1, 1, 6, 9, 7 },
        { 5, 12, -1, 6, -1, 3, 15, -1, 9, 0, -1, -1, 2, -1, -1, -1 } };

    int[][] solution1 = { { 11, 2, 5, 7, 4, 10, 9, 15, 6, 14, 0, 1, 12, 3, 8, 13 },
        { 14, 4, 0, 9, 11, 8, 2, 12, 13, 5, 3, 10, 15, 1, 7, 6 },
        { 1, 10, 15, 12, 6, 13, 7, 3, 4, 9, 8, 2, 11, 5, 14, 0 },
        { 13, 8, 6, 3, 5, 14, 1, 0, 11, 7, 15, 12, 10, 9, 4, 2 },
        { 0, 7, 14, 2, 8, 11, 4, 9, 1, 6, 10, 5, 13, 12, 3, 15 },
        { 3, 5, 9, 1, 10, 12, 14, 7, 2, 4, 13, 15, 8, 0, 6, 11 },
        { 12, 15, 10, 13, 2, 1, 3, 6, 8, 11, 14, 0, 7, 4, 5, 9 },
        { 6, 11, 8, 4, 0, 15, 5, 13, 7, 12, 9, 3, 14, 2, 10, 1 },
        { 2, 14, 1, 15, 3, 7, 11, 4, 5, 10, 6, 9, 0, 13, 12, 8 },
        { 7, 6, 12, 8, 9, 2, 10, 5, 0, 13, 1, 11, 4, 14, 15, 3 },
        { 4, 9, 3, 5, 14, 0, 13, 8, 15, 2, 12, 7, 6, 11, 1, 10 },
        { 10, 13, 11, 0, 15, 6, 12, 1, 3, 8, 4, 14, 9, 7, 2, 5 },
        { 9, 1, 2, 14, 7, 4, 0, 11, 10, 15, 5, 6, 3, 8, 13, 12 },
        { 8, 3, 7, 10, 13, 9, 6, 2, 12, 1, 11, 4, 5, 15, 0, 14 },
        { 15, 0, 4, 11, 12, 5, 8, 10, 14, 3, 2, 13, 1, 6, 9, 7 },
        { 5, 12, 13, 6, 1, 3, 15, 14, 9, 0, 7, 8, 2, 10, 11, 4 } };

    int[][] example2 = { { 4, -1, -1, 9, -1, 14, -1, 0, -1, -1, -1, 6, -1, -1, -1, -1 },
        { 3, -1, -1, 2, -1, -1, -1, -1, -1, 8, 5, 11, 10, 0, -1, 14 },
        { 13, -1, -1, -1, 10, 2, 8, -1, 1, 12, -1, -1, -1, -1, 9, -1 },
        { 10, 7, -1, -1, 4, -1, 3, 15, -1, -1, -1, -1, -1, 8, -1, 12 },
        { 5, -1, 3, -1, -1, 12, 4, -1, 13, -1, -1, -1, -1, 11, -1, -1 },
        { 14, -1, -1, -1, -1, 0, -1, 13, 15, -1, 9, -1, 6, 3, 8, -1 },
        { 7, 8, -1, 15, -1, 3, 1, 10, 14, -1, -1, 4, -1, 5, -1, -1 },
        { 11, 10, 1, -1, -1, -1, 9, -1, -1, -1, -1, -1, -1, -1, 0, 4 },
        { 9, 3, 13, -1, 7, 8, 15, -1, 6, -1, -1, 0, -1, 14, -1, -1 },
        { 8, -1, 15, 1, -1, -1, -1, -1, 5, -1, -1, 14, 0, 12, 10, -1 },
        { 6, -1, -1, 14, 12, 10, -1, -1, 3, -1, 15, 13, 8, -1, 1, 7 },
        { 0, -1, -1, 7, -1, -1, 2, 1, -1, -1, -1, 8, 15, -1, -1, -1 },
        { 12, 0, 7, -1, 8, -1, 11, -1, 10, -1, 1, -1, 5, -1, -1, -1 },
        { 1, 6, -1, -1, -1, -1, 5, 2, -1, -1, -1, 7, 11, 10, 15, -1 },
        { 2, -1, 14, 5, 13, -1, 10, -1, -1, -1, 4, -1, 9, -1, 7, 8 },
        { 15, -1, 9, 10, -1, 1, -1, -1, -1, 2, -1, -1, -1, 6, 4, -1 } };

    int[][] solution2 = { { 4, 1, 8, 9, 5, 14, 12, 0, 7, 10, 13, 6, 3, 2, 11, 15 },
        { 3, 15, 12, 2, 1, 7, 13, 9, 4, 8, 5, 11, 10, 0, 6, 14 },
        { 13, 14, 6, 0, 10, 2, 8, 11, 1, 12, 3, 15, 4, 7, 9, 5 },
        { 10, 7, 5, 11, 4, 6, 3, 15, 9, 14, 0, 2, 1, 8, 13, 12 },
        { 5, 9, 3, 6, 15, 12, 4, 14, 13, 0, 8, 10, 7, 11, 2, 1 },
        { 14, 12, 2, 4, 11, 0, 7, 13, 15, 5, 9, 1, 6, 3, 8, 10 },
        { 7, 8, 0, 15, 2, 3, 1, 10, 14, 11, 6, 4, 13, 5, 12, 9 },
        { 11, 10, 1, 13, 6, 5, 9, 8, 2, 3, 7, 12, 14, 15, 0, 4 },
        { 9, 3, 13, 12, 7, 8, 15, 4, 6, 1, 10, 0, 2, 14, 5, 11 },
        { 8, 4, 15, 1, 9, 11, 6, 3, 5, 7, 2, 14, 0, 12, 10, 13 },
        { 6, 2, 11, 14, 12, 10, 0, 5, 3, 4, 15, 13, 8, 9, 1, 7 },
        { 0, 5, 10, 7, 14, 13, 2, 1, 11, 9, 12, 8, 15, 4, 3, 6 },
        { 12, 0, 7, 3, 8, 4, 11, 6, 10, 15, 1, 9, 5, 13, 14, 2 },
        { 1, 6, 4, 8, 0, 9, 5, 2, 12, 13, 14, 7, 11, 10, 15, 3 },
        { 2, 11, 14, 5, 13, 15, 10, 12, 0, 6, 4, 3, 9, 1, 7, 8 },
        { 15, 13, 9, 10, 3, 1, 14, 7, 8, 2, 11, 5, 12, 6, 4, 0 } };

    @SuppressWarnings("unused")
    int[][] example3 = { { 15, 4, -1, 2, -1, 5, 3, -1, -1, 12, -1, 14, -1, -1, 9, 11 },
        { 0, 12, -1, -1, -1, -1, 7, 10, 3, -1, -1, -1, 8, 4, 15, -1 },
        { 8, 5, 10, 6, -1, -1, -1, 11, 0, -1, -1, -1, -1, -1, 3, -1 },
        { 9, 7, -1, -1, -1, -1, -1, -1, -1, 5, 6, -1, -1, 2, -1, 14 },
        { 13, 8, -1, 4, 0, -1, -1, 14, -1, 3, -1, 12, -1, 9, -1, 1 },
        { 11, -1, 7, 15, -1, -1, -1, 13, -1, 2, 9, -1, 4, -1, 10, 6 },
        { 10, 6, 14, -1, -1, 7, 2, -1, -1, 13, -1, -1, -1, -1, -1, -1 },
        { 2, -1, 12, -1, -1, 4, 6, -1, -1, 15, 7, -1, 14, 11, -1, -1 },
        { 7, 1, 4, 0, -1, -1, -1, 2, 11, -1, -1, -1, -1, -1, -1, 3 },
        { 14, 15, 2, 11, -1, -1, -1, 3, -1, 0, -1, -1, 1, -1, -1, -1 },
        { 12, 13, -1, 10, -1, -1, 1, 6, -1, -1, 3, 7, 15, -1, -1, 9 },
        { 3, -1, 6, -1, -1, -1, -1, 12, -1, 1, -1, 2, -1, 8, 14, -1 },
        { 4, -1, -1, -1, -1, 14, 15, -1, 10, 6, -1, -1, -1, 13, -1, -1 },
        { 5, 14, 3, -1, -1, -1, -1, 7, 2, -1, 0, 1, -1, -1, -1, -1 },
        { 1, -1, 0, -1, 6, -1, 13, -1, -1, -1, -1, -1, -1, 12, 5, -1 },
        { 6, 10, -1, 12, -1, -1, 8, 1, 13, 7, -1, -1, 3, 14, -1, -1 } };

    @SuppressWarnings("unused")
    int[][] example4 = { { 15, 4, -1, -1, 8, -1, -1, 0, 7, 12, -1, -1, -1, -1, 9, 11 },
        { 0, 12, -1, -1, 13, 6, -1, -1, -1, -1, -1, -1, 8, -1, 15, 5 },
        { 8, -1, -1, -1, -1, -1, -1, -1, -1, -1, 15, 13, -1, 1, 3, 7 },
        { 9, 7, 11, -1, -1, 1, -1, 15, 8, -1, 6, -1, 0, -1, -1, 14 },
        { 13, 8, 5, -1, -1, 15, -1, 14, -1, -1, 10, 12, 2, -1, 7, -1 },
        { 11, -1, 7, 15, 1, 12, -1, 13, -1, 2, -1, -1, -1, -1, 10, -1 },
        { 10, 6, 14, -1, -1, 7, -1, -1, 4, -1, -1, -1, -1, 15, -1, -1 },
        { 2, -1, 12, 9, -1, -1, -1, -1, 1, -1, -1, -1, 14, -1, -1, 13 },
        { 7, -1, 4, -1, -1, 9, -1, 2, -1, 10, 8, -1, 13, 5, -1, -1 },
        { 14, 15, 2, -1, 5, -1, -1, -1, -1, -1, 13, 9, -1, -1, -1, -1 },
        { 12, -1, 8, -1, -1, 11, -1, -1, 5, -1, -1, 7, 15, 0, 2, -1 },
        { 3, -1, 6, -1, 7, 13, -1, -1, 15, 1, -1, -1, -1, 8, -1, 10 },
        { 4, -1, -1, -1, -1, -1, -1, 5, 10, -1, -1, 3, -1, 13, 1, 0 },
        { 5, -1, 3, -1, -1, 10, -1, -1, -1, 8, -1, -1, -1, -1, -1, 15 },
        { 1, -1, 0, 7, 6, 3, -1, 4, 9, -1, 14, -1, -1, -1, -1, -1 },
        { 6, -1, 15, -1, 9, -1, -1, 1, 13, -1, 5, -1, -1, 14, -1, -1 } };

    testSudoku("example 1", example1, solution1);
    testSudoku("example 2", example2, solution2);
    //testSudoku("Hard", example3, null);
    //testSudoku("Harder/Impossible?", example4, null);
  }
}
```