Student Name: Ben Sottile

Class and Section CS 210

Total Points (40 pts + n extra pts) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Due: Refer to Blackboard (Late homework submission is not acceptable! You must turn in your own work in blackboard!)**

#### **Project: Project:** Eight Queens Game

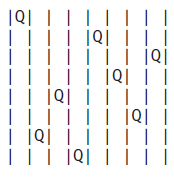
CS210 Introduction to Programming Principles

California Lutheran University

Problem Description:

The classic Eight Queens puzzle is to place eight queens on a chessboard such that no two queens can attack each other (i.e., no two queens are on the same row, same column, or same diagonal). There are many possible solutions. Write a program that displays one such solution.

A sample output is shown below:



Submit the following items:

1. Submit this world document with your answers before the due date.
2. Zip your project and submit it before the due date.
3. Analysis: (10 pts)

(Describe the program use cases. Write down the requirements for your code. Describe the stakeholders for your code. Describe what the SW should be able to do?)

Overview: We need to make a program to find random solutions to the eight queens problem and print them like so

| || || ||Q|| || || || |

| ||Q|| || || || || || |

| || || || ||Q|| || || |

| || || || || || || ||Q|

| || || || || ||Q|| || |

|Q|| || || || || || || |

| || ||Q|| || || || || |

| || || || || || ||Q|| |

Use cases:

People looking for solutions to the eight queens problem

* 1. Requirements:

The Software shall be written in JAVA.

The Software shall not have any coding errors.

The Software shall be documented well.

The Software shall be readable.

The Software shall be testable.

The Software will not crash.

The Software will use a Boolean array to remember what spaces are open

The Software will generate to random numbers to serve as an address for the queens.

The Software will use a method named invalidate to turn spaces that would be unavalibe into falses. True will be considered possible candidates for the next Queens

The program will retrieve working solutions addresses and put them in array

If a solution is not possible the process will be looped through again.

The Array will be read and used to edit a string array of “| |” Strings deemed to be queens will be set to“|Q|”

The Output will look something like this

| || || ||Q|| || || || |

| ||Q|| || || || || || |

| || || || ||Q|| || || |

| || || || || || || ||Q|

| || || || || ||Q|| || |

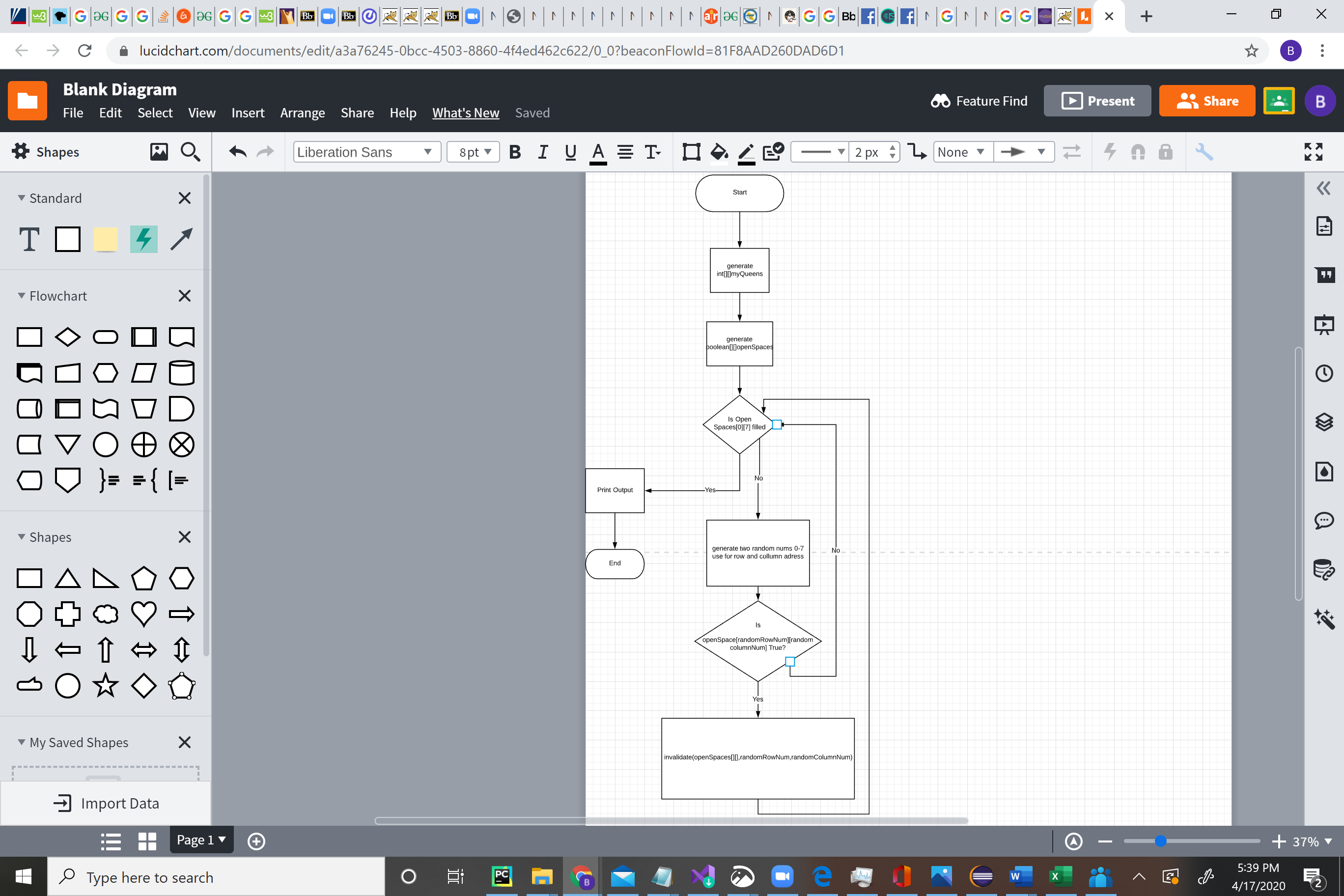
|Q|| || || || || || || |

| || ||Q|| || || || || |

| || || || || || ||Q|| |

1. Design: (10 pts)

(Describe the major steps for solving the problem. Only a flow chart will be acceptable.)



1. Coding: (10 pts)

(Copy and Paste Source Code here. Format your code using Courier 10pts. Screen shot of your code is also acceptable.)

**package** cs210;

//Welcome people

//generate board based on input

//randomze space

//generate invalid ints

**public** **abstract** **class** HW8 {

**public** **static** **void** main(String[] args) {

System.out.println("Welcome to Eight Queen's. Let us see who is better. I will go first.");

**int**[][] myQueens = { { -1, -1, -1, -1, -1, -1, -1, -1 }, { -1, -1, -1, -1, -1, -1, -1, -1 } };

**boolean**[][] openSpaces = { { **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** },

{ **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** },

{ **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** },

{ **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** },

{ **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** },

{ **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** },

{ **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** },

{ **true**, **true**, **true**, **true**, **true**, **true**, **true**, **true** } };

**int** queenCount = 0;

**while** (**true**) {

**if** (myQueens[0][7] != -1) {

**break**;

}

**int** addressRowGuess = ((**int**) (Math.random() \* 8));

**int** addressColumnGuess = ((**int**) (Math.random() \* 8));

**if** ((openSpaces[addressRowGuess][addressColumnGuess]) == **false**) {

**for** (**int** i = 0; i <= 1; i++) {

**for** (**int** y2 = 0; y2 <= 7; y2++) {

myQueens[i][y2] = -1;

}

}

**for** (**int** i = 0; i <= 7; i++) {

**for** (**int** y2 = 0; y2 <= 7; y2++) {

openSpaces[i][y2] = **true**;

}

}

queenCount = 0;

**continue**;

} **else** {

invalidate(openSpaces, addressRowGuess, addressColumnGuess);

myQueens[0][queenCount] = addressRowGuess;

myQueens[1][queenCount] = addressColumnGuess;

queenCount++;

}

}

String[][] board = { { "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" },

{ "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" },

{ "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" },

{ "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" },

{ "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" },

{ "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" },

{ "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" },

{ "| |", "| |", "| |", "| |", "| |", "| |", "| |", "| |" }, };

**for** (**int** y2 = 0; y2 < myQueens[0].length; y2++) {

board[myQueens[0][y2]][myQueens[1][y2]] = "|Q|";

}

**for** (**int** i = 0; i < board.length; i++) {

System.out.println();

**for** (**int** y2 = 0; y2 < board[0].length; y2++) {

System.out.print(board[i][y2]);

}

}

System.out.println("");

System.out.println("Hahaha I win !");

}

**public** **static** **void** invalidate(**boolean**[][] openSpaces, **int** addressRow, **int** addressColumn) {

**for** (**int** i = 0; i < 8; i++) {

openSpaces[addressRow][i] = **false**;

}

**for** (**int** i = 0; i < 8; i++) {

openSpaces[i][addressColumn] = **false**;

}

**int** y = addressRow;

**for** (**int** x = addressColumn; x >= 0; x--) {

**try** {

openSpaces[y][x] = **false**;

} **catch** (Exception e) {

**break**;

}

y--;

}

y = addressRow;

**for** (**int** x = addressColumn; x >= 0; x--) {

**try** {

openSpaces[y][x] = **false**;

} **catch** (Exception e) {

**break**;

}

y++;

}

**int** x = addressColumn;

**int** y3 = addressRow;

**try** {

**while** (**true**) {

openSpaces[y3][x] = **false**;

x++;

y3--;

}

} **catch** (Exception e) {

}

x = addressColumn;

**for** (**int** y1 = addressRow; y1 <= 7; y1++) {

**try** {

openSpaces[y1][x] = **false**;

} **catch** (Exception e) {

**break**;

}

x++;

}

}

}

1. Testing: (10 pts)

(Describe how you test this program, and attach your results screen shots here. For this homework at least 3 test cases are required (Be clear in your results.)

