CS 11 – Mid-term exam #2 ANSWERS

1. QUESTION: (30 points) Write a method that performs *binary search* on an array of `int`. This method should search for a value specified by the caller and return the index at which the value is found or `-1` if it does not exist in the array. The method should begin like this:

   ```
   public static int binarySearch (int[] x, int v) {
   ```

   ANSWER: Although this method can be either recursive or non-recursive, here is a recursive version:

   ```
   public static int binarySearch (int[] x, int v) {

       return doBS(x, v, 0, x.length - 1);

   }

   public static int doBS (int[] x, int v, int lower, int upper) {

       if (lower > upper) {
           return -1;
       }

       int mid = (lower + upper) / 2;
       if (x[mid] == v) {
           return mid;
       } else if (x[mid] < v) {
           return doBS(x, v, lower + 1, upper);
       } else {
           return doBS(x, v, lower, upper - 1);
       }

   }
   ```

   COMMENTARY: Many forgot the difference between binary search and linear search, and thus wrote a method that performed the (much simpler) latter type of search. For those that attempted a true binary search, the most common problems were in computing the middle index in a way that progressively halved the range correctly.

2. QUESTION: (35 points) Consider the game *simple sudoku.* This game is played on a 9x9 grid of integers whose values are between 1 and 9. A solved simple sudoku grid contains one of each value (1 to 9) in each row and each column. The game begins with only a few values, scattered around the grid, and the player must fill in the remaining values to construct a solution. **Write a method** named `testGrid` that accepts a pointer to a simple sudoku grid, represented as a two-dimensional array of `int`, and then tests that grid to determine if it is a solved grid. That is, your method must return `true` if the 2-D array is of the correct size, each row contains the values 1 to 9, and each column contains the values 1 to 9; it must return `false` otherwise. *Hint:* It may be useful to write one or more supporting methods that `testGrid` calls to perform repeated tasks.

   ANSWER: This method is a simplified version of what you've done for project-3a:

```
public static boolean testGrid (int[][] grid) {

    if (grid.length != 9) {
        return false;
    }

    for (int i = 0; i < 9; i++) {

        if (grid[i].length != 9) {
            return false;
        }

        for (int v = 1; v <= 9; v++) {

            boolean foundRow = false;
            boolean foundColumn = false;

            for (int j = 0;
                 (j < 9) && (!foundRow) && (!foundColumn);
                 j++) {

                if (grid[i][j] == v) {
                    foundRow = true;
                }

                if (grid[i][j] == v) {
                    foundColumn = true;
                }

            }

            if (!foundRow || !foundColumn) {
                return false;
            }

        }
    }
}
```

   COMMENTARY: The most common mistakes were not to check the length of *every* column in the grid, not to detect repeated values in the same row/column, or to construct a testing loop that would return `true` prematurely (before all values had been tested for all rows/columns).

3. QUESTION: (35 points) The strangest thing happened this morning. When you woke up, sitting next to your bed was a stack of one-thousand dollar bills, a list of n numbers, and a note that read:

> These are the prices at which Google stock will sell over the next n days, starting today. You may choose to buy this stock on one particular day and then sell it on some later day, but you only get to make one purchase and sale—if you perform multiple purchases and sales, I will change the prices! Good luck!

A strong feeling came over you that these numbers were real, and that you were meant to use them to get rich. However, the list of numbers is long! On which day should you buy and on which day should you sell to make the greatest profit?
Write a method that accepts an array of `double` containing these n values. This method should determine, based on the array, the day on which you should buy your Google stock and the later day on which you should sell it to make the maximum single buy/sell profit. It should print these values to the user.
ANSWER: A pair of nested loops does the trick:

```
public static void maximizeProfit (double[] values) {

    int buyDay = 0;
    int sellDay = 0;
    double maxProfit = 0.0;

    for (int i = 0; i < values.length; i++) {
        for (int j = i; j < values.length; j++) {

            double profit = values[j] - values[i];
            if (profit > maxProfit) {

                maxProfit = profit;
                buyDay = i;
                sellDay = j;

            }

        }
    }

    System.out.println("Buy on day " + buyDay +
                       "and sell on day " + sellDay +
                       "for a profit of " + maxProfit);

}
```

COMMENTARY: By far the most common error was to search for *the* minimum and *the* maximum values as the buy and sell days. This approach violates the critical requirement that the purchase occur before the sale. A slightly better (but still flawed) variation was to search for *the* minimum value and then find the highest value that follows that minimum. While this doesn't violate the buy-then-sell requirement, it doesn't guarantee maximal profit. (Be sure that you see why!)
You must consider every pair of possible days for which the purchase day precedes the sale day, and so the above loops take that approach.