# CSC260 Homework 5 (CH8)

- 50 points in total
  - 10 points from MyProgrammingLab questions (2 points per each question you select)
  - 20 points from class activities
  - 20 points from Programming questions.
- Submit your homework at Canvas/Assignments/HW5
- If you have any questions about the homework, please send me an email (start with CSC260 with a section number) or open a Discussion on Canvas.

## 3. Programming (20 points)

1. You should make the program that returns correct answer.
2. You should print out the results and paste them as a comment.
3. For getting outputs, use the inputs from the sample run. Some programs don't need an input, then print out and copy the results.
4. Copy only Java files for submission; copy only the Java files in the `programming` directory.
5. Students earn 100% when they get correct answers and copied results, 60% when they get wrong answers, 0% when they can't compile the Java source or no answers copied.

### 3.1 FindNearestPoints.java (5 points)

Given a set of points, the closest-pair problem is to find the two points that are nearest to each other. In this figure, for example, points (1, 1) and (2, 0.5) are closest to each other. There are several ways to solve this problem. An intuitive approach is to compute the distances between all pairs of points and find the one with the minimum distance, use this algorithm to find the points that are closest to each other.

- You can use the skeleton code.
- **You can use a hint from Liang's text book case study of "Finding the closest pair."**
- Your program should print out the following: "The closest two points are (1.0, 1.0) and (2.0, 0.5)''

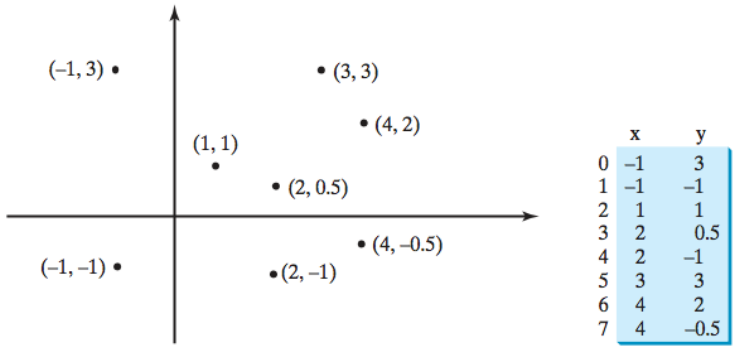


**FIGURE 8.3** Points can be represented in a two-dimensional array.

Figure 1: The nearest points are (1,1) and (2, 0.5)

**Hint - skeleton of main method** Use this code only when you absolutely need it.

```
        // Compute distance for every two points
        for (int i = 0; i < points.length; i++) {
            for (int j = i + 1; j < points.length; j++) {
                double distance = ???

                if (shortestDistance > distance) {
                    ??? // Update p1
                    ??? // Update p2
                    ??? // Update shortestDistance
                }
            }
        }

        // Display result
        String result = "The closest two points are " +
                "(" + points[p1][0] + ", " + points[p1][1] + ") and (" +
                points[p2][0] + ", " + points[p2][1] + ")";
        System.out.println(result);
    }
```

### 3.2 IdenticalArrays.java (5 points)

The two-dimensional arrays m1 and m2 are strictly identical if their corresponding elements are equal. Write a method that returns true if two arrays, m1 and m2, are strictly identical.

- You first check if the length of an array is the same: `m1.length * m1[0].length != m2.length * m2[0].length`
- You then check if each elements are the same or not.
- Your program should print out the following:

```
true <- array m1 and m2 are the same
true <- aray m1 and m3 are not the same, but because of !, you should have true outptul.
```

**Hint**

```
public static boolean equals(int[][] m1, int[][] m2) {
    // you can check each element the same or not

    for (int i = 0; i < m1.length; i++)
        for (int j = 0; j < m1[i].length; j++)
            if (m1[i][j] != m2[i][j])
                return ???;

    return true;
}
```

### 3.3 SummingElements.java (10 points)

Given a two dimensional array, write three methods that (1) sums all elements, (2) sums only certain column elements, and (3) sums only certain row elements.

For example, with an array of {{1,2},{3,4}}, (1) the sum of all elements is (1+2+3+4), (2) the sum of first column elements are (1+3), and (3) the sum of first row elements are (1+2).

- You can use the skeleton code.
- Your program should print out the following:

```
true
true
true
```

**Hints** Use hints only if you need them absolutely necessary. Try to solve the problem on your own.

```
public static int summingAllElements(int[][] matrix) {
    int sum = 0;
    for (int row = 0; row < matrix.length; row++) {
        for (int column = 0; column < matrix[row].length; column++) {
            ???
        }
    }
    return sum;
}
```