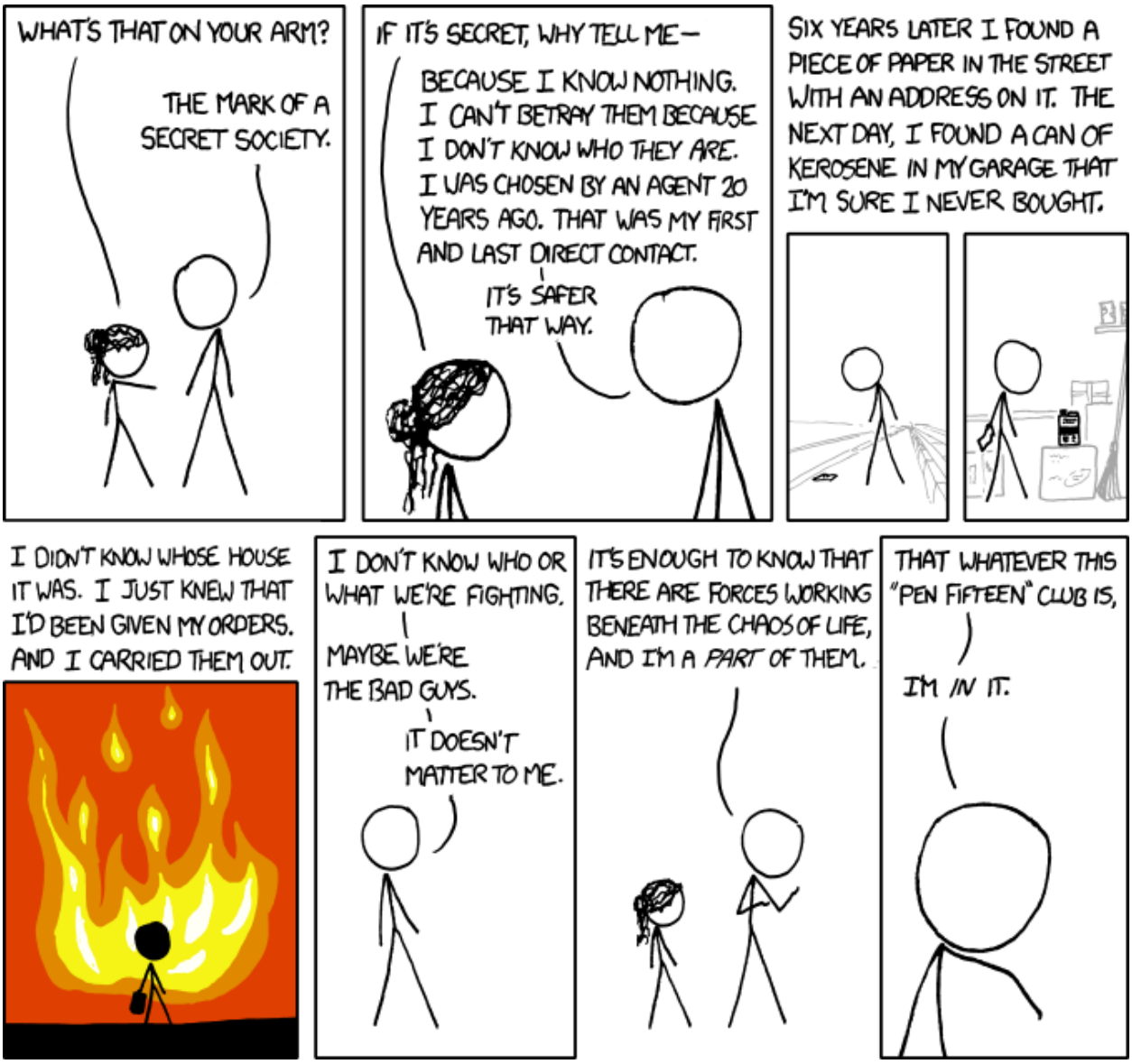
The Sorting Hat



You have been contacted by the mysterious *Pen-Fifteen* agency to help them write a visualization of sorting algorithms for students in *CS 201: Data Structures*. Unfortunately, for all their fancy graphics, they seem to have forgotten how basic sorting algorithms work!

Your task, should you choose to accept it, is to set the record straight.
Should you not, your grade, and family, will suffer.

To: [RECIPIENT-UNDISCLOSED]
From: Pen-Fifteen
Subject: Sorting Algorithms

Take a look at ArraySorter.java. This is the only file you must modify, so know it well. In it are four sorting algorithms: *bubble*, *shaker*, *insertion* and *selection* sort. Unfortunately, we only remembered how to write the simplest: bubble sort. You must succeed where our greatest minds have failed and implement the other three.

First, familiarize yourself with the *main()* method. Make sure you understand what each line does. There's a lot of redundant code, but we couldn't think of a better way to handle matters. Maybe you can help us with that later...

Second, read through the *bubbleSort()* method. Notice that *bubbleSort()* operates on a *SortableArray* object that we have provided for you. This class has three methods that you must know about. These are the only three you should need.

- **compare (int i, int j)** - Compare the items at indices i and j, and returns:
  NEGATIVE if item at $i <$ item at j
  ZERO if item at $i =$ item at j
  POSITIVE if item at $i >$ item at j
- **swap (int i, int j)** - Swap the items at indices i and j.
- **markAsSorted (int i, int j)** - Visually show that an index is sorted.
  This function can be called with 0, 1 or 2 arguments.
  With 0, it marks the entire array as sorted.
  With 1, it marks a single index as sorted.
  With 2, it marks all indices from i through j (inclusive) as sorted.

Finally, you are ready to begin your task. Start by implementing *shakerSort()* as it is the most similar to *bubbleSort()*. Or so we hear. Then, you may tackle *selectionSort()*. Then, and only then, must you best *insertionSort()*.

Good luck, and remember: your family is counting on you.

Grading

Algorithms

Shaker Sort	(5)
Selection Sort	(5)
Insertion Sort	(5)

Code Quality

Neatness, consistency, readability	(5)

Extra Credit

Merge Sort	(5)
Quick Sort	(5)

Super Duper IMPORTANT Things

Talk through the following with your partner and ask me about about any you don't understand. These aren't graded because that would be annoying for all involved, but not being able to answer any of them indicates some conceptual misunderstanding which is in your (and your family and grade's) best interest to resolve.

1. You should be able to understand every single line in main(). If you don't know quite what a particular line does, try commenting it out and seeing what changes! If things suddenly explode, chances are it was the *do not explod*e line.

2. What is the efficiency of each sort and why? No notes or memory! Reason only!