INTRODUCTION TO COMPUTER SCIENCE I
LAB 5
Searching arrays

1 Seeing the future

You walk into Frost Library to return a book.[1] You decide to look up Knuth's *The Art of Computer Programming*. Seeing that it is down in C-level, you descend the stairs. Strangely, though, when you get to C-level, you notice, in the semi-darkness, that the stairs **continue down another level**. You have never seen these stairs before, thinking that C-level was the bottom of this building.

Feeling adventurous, you descend to the little-known and, as it turns out, magical D-level. In it, you find stacks of periodicals—newspapers, magazines, etc. You approach a stack of Wall Street Journals, and pick up the one on the top. It's date is *April 1, 2019*. "But . . . that's *next Monday*!" you exclaim. Digging through the stack, each Journal is marked as being yet another day into the future. Disbelieving, you write down the titles of a couple of articles: *Complete Mueller Report Leaked*, and *British Parliment on Brexit: Aw, Forget It*. There's even a sports article: *Auburn Tops Kentucky In Double OT, Reaches Final Four*.

Sunday comes, and you watch Auburn hit shots at the buzzers to end regulation and the first overtime, and then win the game. The game is exactly as described in the article that you saw on D-level. You wake up on Monday morning, run to Hastings, and buy a copy of the WSJ—and there are those headlines that you copied while in D-level, verbatim. Wow. You run back to Frost, sprinting down the stairs into D-level again. Monday's paper is there, and so is Tuesday's, Wednesday's, *et cetera*. Even more oddly, you seem to be the only person in the room. It's not clear anyone else sees it; perhaps nobody else has ever seen it. Creepy.

Sensing opportunity, you grab the next few days of the journal, and you look at the stock prices.[2] If these newspapers are what they seem to be, you could make a killing! So, you grab your laptop out of your backpack, and you pick a stock—*Lyft* is brand new—and start writing down its prices for tomorrow, the next day, and so on. After many hours of labor, you have a lengthy list of Lyft share prices, day by day, going years into the future.

Now you realize that you need to take advantage of this magical information in order to make as much money as possible. Good thing you're taking COSC 111! Otherwise, you would have trouble figuring out how to calculate, from this list of numbers, on which day you should buy as much Lyft stock as you can afford, and on which later day you should sell it.[3] You settle in to write a program that will read that list of stock prices, and after a few seconds of calculating, print out the **day to buy**, the **day to sell**, and the **factor increase in your money**.[4]

[1] You know, the old codex things made of stacked and bound paper.

[2] You could try to use this knowledge to save the world, but really, who would believe you? Might as well make some money.

[3] In the WSJ from Apr 1, 2019, the Securities and Exchange Commissions ban short selling, so you know that you must buy first and sell later.

[4] That is, if the stock price increases from \$25 per share on your buy-day to \$125 per share on your sell-day, then your increase is a factor of $\frac{125}{25} = 5$.

2 What you must do

Create a directory for this project, change into it, and grab source code:
`bit.ly/COSC-111-lab-5-source`

Open the Java source code file, `PickEm.java`, with *Emacs/Aquamacs*. It contains a fully-formed `main()`, which calls on a methods to generate a random array of "stock prices" (floating point numbers) of some given length (on the command line).[5] It then calls on the method `findBuySell()`, passing it the array of stock prices. This method is expected to return, as a two-element `int` array:

- `[0]` The day number on which one ideally should buy the stock, and
- `[1]` The day number on which one should sell it.

Therefore, this method should **always return an array of length `2`**, where the value at index `0` contains the *buy-day*, and the value at index `1` contains the *sell-day*.

2.1 A solution

Write the `findBuySell()` method. Specifically, employ a *brute force algorithm* that considers every possible buy/sell day pairings (remembering that the buy-day must **precede** the sell-day), and chooses the best. Once you write such a method, **test it**. Add code to print the array of prices, and then run the program with small but increasing numbers of days (which you get to specify on the command line). For example:

```
$ java PickEm 1
Day = 0, price = 811.3915355625351
Buy on 0 at 811.3915355625351
Sell on 0 at 811.3915355625351
Factor profit = 1.0

$ java PickEm 5
Day = 0, price = 811.3915355625351
Day = 1, price = 27.67945220153589
Day = 2, price = 359.17924706247993
Day = 3, price = 149.29240060832484
Day = 4, price = 814.3190563057799
Buy on 1 at 27.67945220153589
Sell on 4 at 814.3190563057799
Factor profit = 29.419623277826087
```

You likely want to test your solution on as lengthy a list of prices as you can examine by hand. Notice that, if you choose a sufficiently high number of days, then the output will be too long

[5] We will discuss, in lab, how this works.

to view in your terminal window. So, you can employ *shell redirection*, sending the output that normally appears within the terminal window into a file instead. Having done so, you can then open the file with *Emacs* and examine the whole output, like so:

```
$ java PickEm 500 >& output.txt
$ emacs output.txt &
```

Once you have determined that your program is working correctly, then **remove the debugging code** that prints the array of prices, and run the program on a large input. Specifically, run your program with $250,000$ days:

```
$ java PickEm 250000
```

If a sane output results (although it is hard to know if it is *correct*), then you are done. I will run your program with this input size to test it.

3 Submitting your work

Submit your `PickEm.java` file with the CS submission system, using one of the two methods:

- **Web-based:** Visit the CS submission systems web page at:
  `www.cs.amherst.edu/submit`

- **Command-line based:** Use the `cssubmit` command at your shell prompt. (WARNING: This method works only on `remus/romulus`.)

This assignment is due on Thursday, Apr-04, 11:59 pm.