Lab 03 Specification – Explore Data Processing using Arrays
Due (via your git repo) no later than 8 a.m., Wednesday, 6th Mar 2019.
50 points

Lab Goals

- Practice working with arrays
- Turn a sorting algorithm into a working function
- Implement a basic encryption scheme
- Answer a few questions to test your knowledge about the class content to-date

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

If you have not done yet, you should read the following to do well on this assignment. It is also highly important to review the code from class discussions, lecture slides, and class notes to do well in this lab.

- GT chapter 03 - Section 3.1

Assignment Details

In this lab, we will continue our discussion of arrays from the last class session. We will walk through algorithms that have practical uses, including sorting an array and encrypting a message. You will also answer a few short questions at the end.

So how to compile the program

At this point we should all be familiar with the compilation of Java programs using **javac** command. Since, there is a particular directory structure associated with the lab and to make the compilation process simple, there is a compile bash script shipped with the lab repository. In order to compile your programs, make sure to place all your Java files under the src directory, Then, open the terminal and navigate to the lab repository and type in:
"./compile.sh"
This should compile all your files and automatically place their class files in the classes sub directory located in the lab repository.

So how to execute the program

At this point we should all be familiar with executing Java programs using **java** command. Since, there is a particular directory structure associated with the lab and to make the execution process simple, there is a run bash script

shipped with the lab repository. In order to execute your programs, make sure to open the terminal and navigate to the lab repository and type in:
"./run.sh SelectionSort"
Here SelectionSort is the name of the class file that is to be executed. If you have a different class file that needs to be executed, then specify that class name while executing the run.sh bash file.

Sorting an Array (20 points)

In a recent class, we used code to find the minimum value in an array. By repeating this process, and always moving that smallest array item to the left, we can sort an array. This algorithm is called Selection Sort, because we Select the smallest array item from what remains and move it into the correct position. Here is the algorithm:

1. For each index in the array (call it i):
   (a) Scan through the array to find the smallest item from the items not yet positioned (call the location of the smallest item j).
   (b) Swap the items located in i and j.
2. Repeat until all items have been considered as i. (Print an output line each time.)

A copy of the [class activity code] called `ArrayProcessor1.java` that is used to find the min element in an array is provided under the src directory. It is recommended to take the code provided as a starting point to implement this part.

A trace of the algorithm on an array of 5 items can be seen at the top of the next page:

```
8  3  4  2  1
1  3  4  2  8
1  2  4  3  8
1  2  3  4  8
1  2  3  4  8
1  2  3  4  8
```

The first line represents the input array: $\{8, 3, 4, 2, 1\}$. Because we have five items to sort, there are then five iterations of the sorting algorithm that follow. In the first iteration, item i is the first position of the array, the 8. We scan across the array to find the smallest item, the 1 at the end, which we will call j. We swap i and j, and the new array is shown on the second line.

Because the 1 is the smallest item, we no longer need to worry about sorting it – it is definitely in the correct position. We move our i pointer to the second position of the array, the 3. We scan across the rest of the items looking for the smallest item, which we find as the 2 second from the right, which we set as j. We swap i and j, and the new array is shown on the third line. This algorithm repeats until we have shifted i across the entire array.

Note that nothing has changed in the last three lines. This is because when i was in the fourth position of the array, located at the 4 in the array, the smallest value j was also the 4. The swap still happened, it just swapped the number with itself! The same thing occurs on the fifth iteration, where the 8 is swapped with itself.

For this section, you will write a Java file called `SelectionSort.java`. The file should sort an array of integers (which can be hardcoded into the file). The algorithm should be able to sort the array regardless of the length of the array, so your `for` loops should be bounded by the `.length` property of the array, rather than have a hardcoded upper bound.

Additionally, your code should neatly print each line of the sorting process as it occurs, rather than simply displaying the input and final sorted result. You can use the 8-3-4-2-1 input in the above example for testing, but you should prove that your selection sort implementation works on an array of at least 10 numbers.

Caesar Cipher (20 Points)

One of the simplest ways to encrypt a message is to use what is called a Caesar Cipher. In this encryption scheme, each letter of the alphabet is replaced with a different letter a constant distance away. For example, if the distance is selected as 5, then every A is replaced with an F, every B is replaced with a G, and so on through the alphabet. When the end of the alphabet approaches, letters will have to wrap around – every W is replaced with a B, every X is replaced with a C, etc.

Java allows us to implement a Caesar Cipher rather easily, because the `char` datatype is simply a special version of an integer. If we run the code `char c = 'A' + 5;`, the result is the 'F' character. Likewise, we can decrypt the message by subtraction: `char c = 'T' - 5;` will result in the 'O' character. This gets complicated as you approach the end of the alphabet. `char c = 'X' + 4` will not return a 'B' character. To wrap around, we can use the modulus (`%`) operation, which returns the remainder left over following a division operation.

There exists one further complication – the letters 'A' through 'Z' are not nicely encoded as 0-25 or 1-26 in Java. You should spend some time researching the ASCII character encoding scheme to understand how Java interprets characters. Note that upper-case and lower-case characters are even encoded with different values under the ASCII scheme.

For this section, you will write a Java file called `CaesarCipher.java`. This file should contain a message stored as an array of characters, as well as an integer for the distance between characters. Although not a requirement for this part, it is recommended to apply a creative approach to your implementation by having the message and distance user driven. You should implement two functions: `encrypt()` and `decrypt()`. The `encrypt` function will take the plaintext message and encrypt it based on the set offset. The `decrypt` function will take the encrypted message that you generated and reverse the encryption back into plaintext. Note that if a character is not an upper-case or lower-case letter, the encryption and decryption should skip over it. So the input string "HELlo user" will 3-encrypt to "KHOor xvhu" with the space unchanged.

Additional Questions (10 points)

Please answer the following questions thoroughly:

1. Explain inheritance in your own words. What is it used for, and why did we choose to implement it in our Rock-Paper-Scissors game?
2. Write a short Java function to calculate the next 5 Fibonacci numbers, in the unusual case where we start with `fib[0]=-1` and `fib[1]=-2`.
3. What is an abstract class? Give a real-life example of when we would want to use one.

Submission Details

For this assignment, please submit the following to your GiHub repository by using the link shared to you by the course instructor:

1. Your source code for `SelectionSort.java` and `CaesarCipher.java`
2. Sample output runs for `SelectionSort.java` and `CaesarCipher.java` in PDF format.
3. The answers to the questions from the "Additional Questions" section in a PDF format.
4. A readme file explaining how to compile and run your program. In addition, include a detailed self reflection of the lab exercise in the readme file. The self reflection may include answers to questions such as:

- ”What challenges did you face in this lab?”
- ”What did you like in this lab?”
- ”What did you not like in this lab?”
- ”How long did you spend on this lab assignment?”

5. It is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllbus. Make sure to add the statement ”This work is mine, unless otherwise cited.” in all your deliverables such as source code and PDF files.

Grading Rubric

1. There will be full points awarded for the lab, if all the requirements in the lab specification are correctly implemented. Partial credits will be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your git repo, will lead you to receive ”0” points given for the lab submission. In this case, there is no solid base to grade the work.
3. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in github. In this way, an updated version of student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then in this case, there is 0 points to the student automatically for the lab work.
4. If you needed any clarification on your lab grade, talk to the instructor. The lab grade may be changed if deemed appropriate.