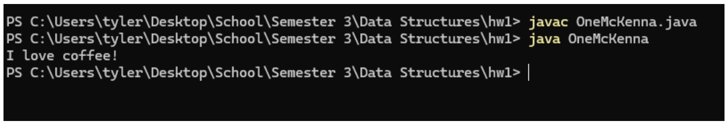
Tashfeen, Ahmad
CSCI 2114: Data Structures

Homework 1

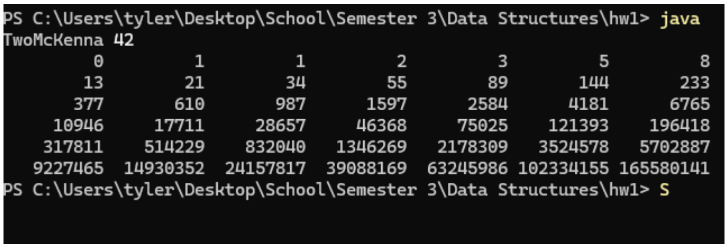
Question 1. Write a Java program that prints out one line of text to the console. It can be anything but "Hello World!"



```
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> javac OneMcKenna.java
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> java OneMcKenna
I love coffee!
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> |
```

Figure 1. Working Code for Q1

Question 2. Write a Java program that populates an array of size n with the first n Fibonacci numbers. The program should print out the array as shown in figure **??**. Here n should be the first command line argument.



```
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> java
TwoMcKenna 42
        0         1         1         2         3         5         8
       13        21        34        55        89       144       233
      377       610       987      1597      2584      4181      6765
    10946     17711     28657     46368     75025    121393    196418
   317811    514229    832040   1346269   2178309   3524578   5702887
  9227465  14930352  24157817  39088169  63245986 102334155 165580141
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> S
```

Figure 2. Working Code for Q2

Question 3. Using the Sieve of Eratosthenes, populate a boolean array of size n marking all the indices that are Prime numbers. Here n should be the first command line argument.

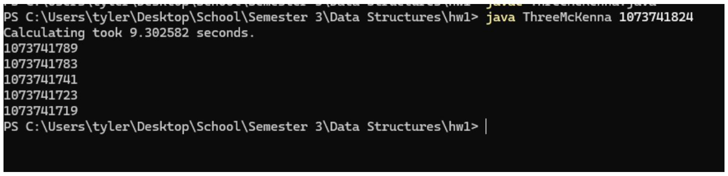
1) For debugging, have your program print all the prime numbers less than a 100. You should get the following,

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97$$

2) The program should print out at most the five largest prime numbers it computed and the time (seconds) it took to compute all the primes less than n. Here is a way to compute seconds taken by a function call `eratosthenes(toSieve)`.

```java
double startTime = System.nanoTime();
eratosthenes(toSieve);
double duration = System.nanoTime() - startTime;
duration = duration / Math.pow(10, 9);
```

3) With your program, calculate how long does it take (in seconds) to compute all the 30 bit prime numbers. These are all primes less than $n = 2^{30} = 1073741824$.



```
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> javac ThreeMcKenna.java
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> java ThreeMcKenna 1073741824
Calculating took 9.302582 seconds.
1073741789
1073741783
1073741741
1073741723
1073741719
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> |
```

Figure 3. Working Code for Q3

4) Can your implementation of the Sieve of Eratosthenes compute all the 32 bit prime numbers? If yes, give the time it takes or if it can not, then why not?

   My implementation of the Sieve of Eratosthenes cannot computer all 32 bit prime numbers due to the integer size limit in java, and even if there was not integer size limit, there is also an array size limit that would prevent that large of an array.

Question 4. Read all bytes in the file `half_gaps.bin`. You may use the function in code listing **??**.

The function in code listing **??** reads in signed bytes. While this maybe suitable for some binary arrangements, we want the bytes to be unsigned. One way to achieve this is to just loop and use `Byte.toUnsignedLong(byte x)` as seen in listing **??**.

Note that while we may also use `Byte.toUnsignedInt(byte x)` to turn the signed bytes into unsigned ints instead of longs, we prefer longs here to avoid overflows in future computations.

Compute the array of longs' cumulative sum, i. e.,

$$\left\{ x_i \in \texttt{cumsum}(x) : x_i = \sum_{k=1}^{i} x_k \right\}$$

Now multiply each of the sums with 2 and then add a 3.

$$\left\{ x_i \in \texttt{cumsum}(x) : y_i = 2x_i + 3 = 2\left(\sum_{k=1}^{i} x_k \right) + 3 \right\}$$

1) Print out the first fifteen and the last five elements of this final array.

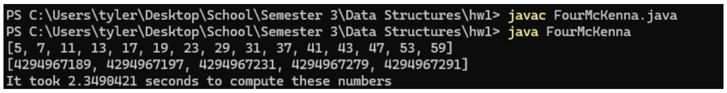
   My solution prints these values as shown in 4

2) Time this program (the reading of bytes, the cumulative sum computation and the doubling with adding a three) and print the result in seconds.

   My solution is shown to take 2.35 seconds in 4

3) Do you recognise the printed numbers? What will these be if we further added a 2 and a 3 to them?

   These are the prime numbers. Adding a 2 or a three to these numbers would stop them from being prime, I'm not sure if it would do anything other than that.



```
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> javac FourMcKenna.java
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> java FourMcKenna
[5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]
[4294967189, 4294967197, 4294967231, 4294967279, 4294967291]
It took 2.3490421 seconds to compute these numbers
```

Figure 4. Working Code for Q4

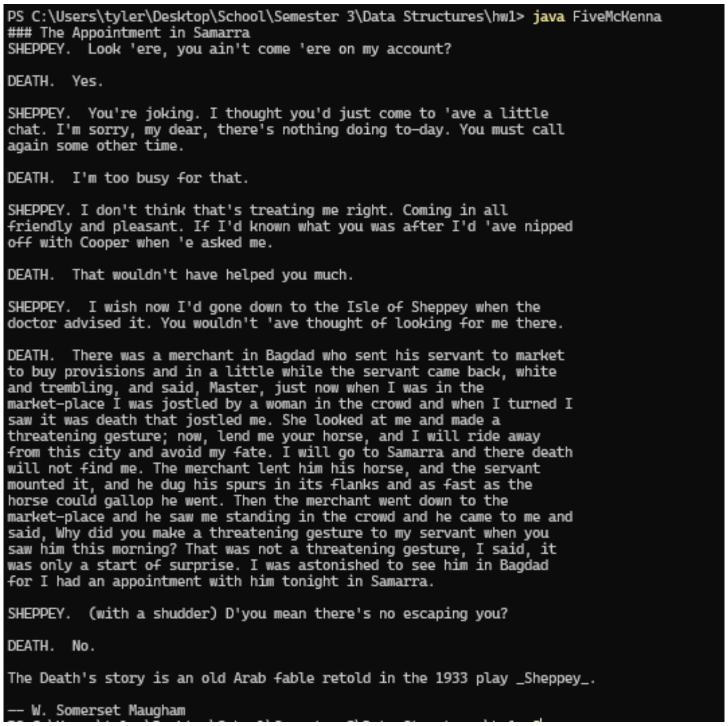
Question 5. Break the Affine cipher. Your professor encrypted a plain text file called `plain.txt` using the program given in listing **??**.

He then redirected the output to a cipher file called

1) Use the cipher text file and the code in listing **??** to recover the plain text. *Hint*: $7^{-1} = 55 \mod 2^7$.

2) What should the 2^7 tell you about the text encoding of the original plain text file?

   Two to the seventh is 128, which is the amount of characters in the ASCII table. We are using mod(2^7) our modular arithmetic because we are using ASCII characters, not just the plain alphabet which would be mod(26)



```
PS C:\Users\tyler\Desktop\School\Semester 3\Data Structures\hw1> java FiveMcKenna
### The Appointment in Samarra
SHEPPEY.  Look 'ere, you ain't come 'ere on my account?

DEATH.  Yes.

SHEPPEY.  You're joking. I thought you'd just come to 'ave a little
chat. I'm sorry, my dear, there's nothing doing to-day. You must call
again some other time.

DEATH.  I'm too busy for that.

SHEPPEY. I don't think that's treating me right. Coming in all
friendly and pleasant. If I'd known what you was after I'd 'ave nipped
off with Cooper when 'e asked me.

DEATH.  That wouldn't have helped you much.

SHEPPEY.  I wish now I'd gone down to the Isle of Sheppey when the
doctor advised it. You wouldn't 'ave thought of looking for me there.

DEATH.  There was a merchant in Bagdad who sent his servant to market
to buy provisions and in a little while the servant came back, white
and trembling, and said, Master, just now when I was in the
market-place I was jostled by a woman in the crowd and when I turned I
saw it was death that jostled me. She looked at me and made a
threatening gesture; now, lend me your horse, and I will ride away
from this city and avoid my fate. I will go to Samarra and there death
will not find me. The merchant lent him his horse, and the servant
mounted it, and he dug his spurs in its flanks and as fast as the
horse could gallop he went. Then the merchant went down to the
market-place and he saw me standing in the crowd and he came to me and
said, Why did you make a threatening gesture to my servant when you
saw him this morning? That was not a threatening gesture, I said, it
was only a start of surprise. I was astonished to see him in Bagdad
for I had an appointment with him tonight in Samarra.

SHEPPEY.  (with a shudder) D'you mean there's no escaping you?

DEATH.  No.

The Death's story is an old Arab fable retold in the 1933 play _Sheppey_.

-- W. Somerset Maugham
```

Figure 5. Working Code for Question 5

5.1. Example Executions

Figure **??** shows how the output of the code for the first three questions should look like on the standard out. All your programs must compile/run from the command line using `javac` and `java` commands, e. g.,

```
javac Program.java
java Program
```

5.2. Submission Instructions

Please *replace the professor's last name* with yours wherever appropriate.

Oklahoma City University, Petree College of Arts & Sciences, Computer Science