Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems over the course of ***20-30 minutes***.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your Course Assistants will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

Problem Solving Team Members

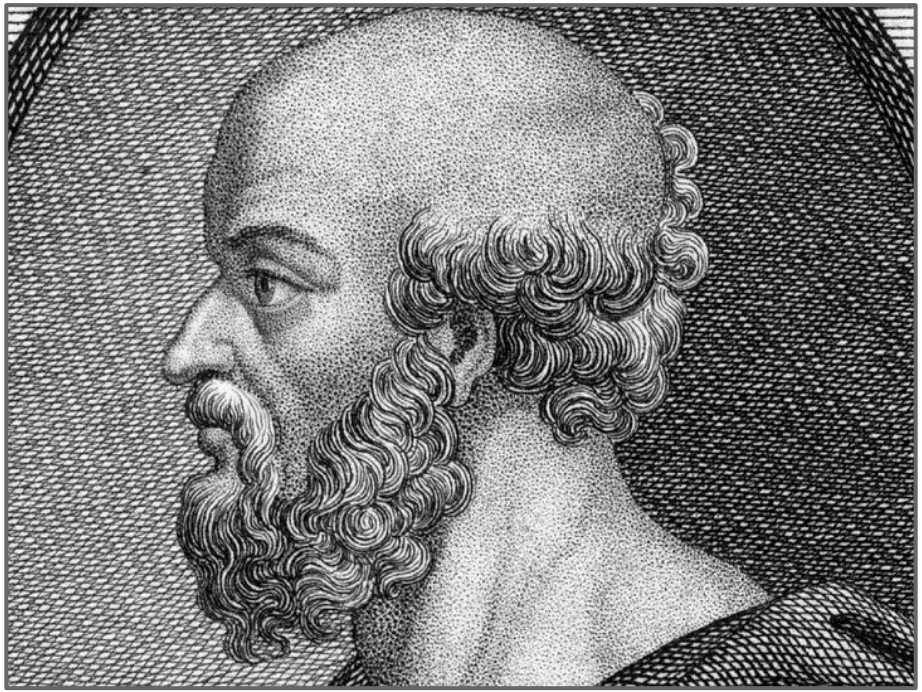
Record the name of each of your problem solving team members here.

Do not forget to ***add every team member's name***! Your instructor (or course assistant) may or may not use this to determine whether or not you participated in the problem solving session.

Eric Manning
Andrew Xie
Dessa Shapiro

The Sieve of Eratosthenes

- One way to determine whether or not a number n is prime is through brute force: divide *n* by all values *k* such that *1 < k < n*.
  - If n is not divisible by any value k, then it is prime.
  - You did something similar to this for the last homework assignment.
- An alternative technique is to use a *Sieve of Eratosthenes*[1].
  - An array of boolean or integer values where the value at each index `n` indicates whether or not `n` is prime.
  - With such an array we can determine if a number is prime in constant time simply by checking its index in the array.



[1] Pronounced "Air-ah-toss-thuh-knees"

Problem Solving 1

Remember that, by default, Java will fill an `int` array with `0`s. A *Sieve of Eratosthenes* can be created by following a simple algorithm:

- `0` and `1` are not prime, so set the value at both indexes to `1`.
- For each value `n` that is greater than `1`:
  - If `sieve[n]` is `1`, skip it.
  - If `sieve[n]` is `0`, set all of the values at indexes that are ***multiples of*** `n` to `1`.

Use the above described algorithm to create a Sieve of Eratosthenes using the table to the right.

n	0	1	2	3	4	5	6	7	8
P?	1	1	0	0	1	0	1	0	1
n	9	10	11	12	13	14	15	16	17
P?	1	1	0	1	0	1	1	1	0
n	18	19	20	21	22	23	24	25	26
P?	1	0	1	1	1	0	1	1	1
n	27	28	29	30	31	32	33	34	35
P?	1	1	0	1	0	1	1	1	1

```
public static int[] makeSieve(int size) {
      Int[] sieve = new int[size];
      sieve[0] = 1;
      sieve[1] = 1;

      For (int number=2; number<size; number++) {
            for (int num=number; num<size;
number++) {
            If (num % number == 0) {
                  sieve[num] = 1;
                  }
            }
      }
      return sieve;
}
```

Problem Solving 2

Working together with your problem solving team, complete the method to the left so that it creates and returns a Sieve of Eratosthenes of the specified size.

For example, if `size=10`, you would return the array:

1	1	0	0	1	0	1	0	1	1
0	1	2	3	4	5	6	7	8	9

What is the time complexity of building a sieve of size *n*?

O(?)

Problem Solving 3

Consider that a *Sieve of Eratosthenes* may be saved to a text file in the following format:

```
7
1100
101
```

The first line indicates the ***size*** of the sieve. The remaining lines contain the values at each index in the array. There may be any number of values on each line.

Complete the method to the right to read such a file and return the sieve that it contains.

Hint: You can use the `Integer.parseInt(String)` method to convert the first line from a `String` to an `int`., e.g.

```
int x = Integer.parseInt("43");
```

However, this only works with strings and ***will not*** work for a `char` like `'1'` or `'0'`.

```
public static int[] readSieve(String filename) {
      Try (FileReader fReader = new
FileReader(filename);) {
      BufferedReader reader = new
BufferedReader(fReader);) {
      String line;
      Int size =
Integer.parseInt(reader.readLine())
      Return makeSieve(size);
      } catch (IOException ioe) {
      System.out.println(“Problem printing file”)
}
```

```
public static void repair(int[] sieve) {
      int len = sieve.length()
      for (int num = 0; num<len; num++) {
            Int value = sieve[num]
            Boolean prime = Primes.isPrime()
            if (prime == true && value != 0) {
                  sieve[num] = 0

            } else if (prime == false && value
      != 1) {
            Sieve[num] = 1;
            }
      }
}
```

Problem Solving 4

A Sieve of Eratosthenes is only valuable if it does not contain any errors. Work together with your problem solving team to complete the method to the left.

Use the `Primes.isPrime(int n)` function that you wrote in your previous assignment to check the value at each index `n` in the sieve. If the value at any index `n` is incorrect, print a message indicating at which index you found the error, and set it to the correct value.