COS 126 Programming Exam 1 Spring 2020

YOU MAY READ THIS PAGE BEFORE THE EXAM PERIOD BEGINS.

Instructions. You will have 50 minutes to create and submit one or two programs. When given the instructions start the exam, download the project `.zip` file, which includes all the files you will need, from the **`Exams`** page.

Resources. You may use your book, your notes, your code from programming assignments and precepts, the code on the COS 126 course website, the booksite, and `Ed` posts. No form of communication is permitted (e.g., talking, texting, etc.) during the exam, except with course staff.

Submissions. Submit your work using the `Submit!` link on the **`Exams`** page.

Grading. Your program will be graded on correctness, clarity (including comments), design, and efficiency. You will lose a substantial number of points if your program does not compile or if it crashes on typical inputs.

Discussing this exam. Due to travel for extracurriculars and sports, some of your peers will take this exam next week. Do not show this exam to anyone until after next week and do not discuss exam contents with anyone who has not taken the exam.

Honor code pledge. The project `.zip` includes a file named `HONORCODE.TXT`. When you are instructed to start, you may "electronically sign" the honor code, by typing this pledge and then your name in this file.

```
I pledge my honor that I have not violated the Honor Code during this examination.
```

DO NOT READ FURTHER OR START DOWNLOADING UNTIL SO INSTRUCTED.

Programming Exam 1: Birthday Problem

Description. Your task in this exam is to write a program that addresses the following version of the well-known Birthday Problem.

> *Suppose that you are given a long list of people's birthdays. Examine the entries on the list one by one to keep track of birthdays that match. How many entries do you have to examine before finding M birthdays that match?*

Part 1 (7 points). Download the project zip file, which includes all the files you will need, from the **Exams** page. Complete the implementation of `Birthday.java` by adding code to the `main()` method that takes an integer value `M` from the command line and reads from standard input a sequence of integer values, stopping and printing out the number of values read as soon as some value has been found to occur `M` times (or, if the input is exhausted, print a message indicating that no such value was found).

```
% more Birthday.java
public class Birthday
{
  public static void main(String[] args)
  {
    int[] counts;
    // YOUR CODE HERE
  }
}
```

Input. Take the integer value `M` from the command line and a sequence of integer values from standard input that represent birthdays and are guaranteed to be between 0 and 365 inclusive (one for each of the 366 possible dates, including February 29). You do not need to check whether the values are legal.

Output. As soon as your program has found a value that has occurred `M` times, it should print a message as shown below with the number of values read, the value of `M`, and the value with `M` occurrences. If no value occurs `M` times in the standard input stream, print a message saying so. For example, your program must behave *exactly* as follows for the file `20birthdays.txt` (which you will find in the project folder).

```
% more 20birthdays.txt
101 123 342 2 3 45 101 31 89 73 123 84 83 65 23 99 123 245 289 302

% java-introcs Birthday 2 < 20birthdays.txt
7 birthdays examined to find 2 occurrences of 101

% java-introcs Birthday 3 < 20birthdays.txt
17 birthdays examined to find 3 occurrences of 123

% java-introcs Birthday 4 < 20birthdays.txt
No birthday occurs 4 times
```

Further testing. Run your program for the file `1000birthdays.txt` (also in the project folder) with `M` equal to 5, 6, 7, 8, and 9 to get the answers 273, 528, 589, 670, and 989, respectively.

Submit `Birthday.java` using the `Submit!` link on the **Exams** page

Do not attempt Part 2 until you have submitted your solution to Part 1.

You cannot get credit for Part 2 unless your solution to Part 1 is correct.

There is no partial credit for Part 2.

Part 2 (0.5 points). What's the largest number of matches on a birthday that we can expect from this class? Assume that we have 300 students and that they have random birthdays. To compute an estimated answer, add code to `BirthdayPart2.java` that performs the following experiment 1000 times: Generate 300 random birthdays, count the number of times each value occurs, and find the maximum count. After performing the 1000 experiments, your program must print the average of the 1000 maximum counts found (just print a `double` value, as in the code given, nothing else).

Submit `BirthdayPart2.java` using the `Submit!` link on the **`Exams`** page

```
% more BirthdayPart2.java
public class BirthdayPart2
{
  public static void main(String[] args)
  {
    double average;

    // YOUR CODE HERE

    StdOut.println(average);
  }
}
```