NAME: **login id:**

Precept:

COS 126 Midterm 1 Programming Exam Fall 2012

This part of your midterm exam is like a mini-programming assignment. You will create two programs, compile them, download test data and run them on your laptop. Debug your code as needed. This exam is open book, open browser—but only our course website and booksite! Of course, no internal or external communication is permitted (e.g., talking, email, IM, texting, cell phones) during the exam. You may use code from your assignments or code found on the COS126 website. When you are done, submit your program via the course website using the submit link for Programming Exam 1 on the Assignments 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.

Even though you will electronically submit your code, *you must turn in this paper* so that we have a record that you took the exam and signed the Honor Code. *Print your name, login ID, and precept number on this page* (now), and write out and sign the Honor Code pledge before turning in this paper. *Note*: It is a violation of the Honor Code to discuss this midterm exam question with anyone until after everyone in the class has taken the exam. You have 90 minutes to complete the test.

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

Signature

Part 1	/14
Part 2A	/8
Part 2B	/8
TOTAL	/30

Your task in this exam is to write two programs related to the famous *derangement* problem:

> *Suppose that N graduating students throw their hats in the air, and each catches a hat. What is the chance that no student gets his own hat back?*

Mathematically, a *permutation* of size N is a rearrangement $p_1 p_2 p_3 \ldots p_N$ of the numbers 1 through N and a *derangement* is a permutation for which $p_i \neq i$ for any i. For example,

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
9	8	7	6	5	4	3	2	1

is *not* a derangement because $p_5 = 5$ and

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
3	7	6	9	8	2	1	5	4

is a derangement. Your job is to write code to identify and count derangements.

Part 1 (14 points). Write a program `HatsPart1` with the following API:

```
public class HatsPart1

public static boolean isD(int[] a)     // Is the permutation in
                                       // array a a derangement?

public static void main(String[] args)// Count how many of the input
                                      // permutations are derangements.
                                      // Output the first derangement
                                      // if it exists.
                                      // Output the derangement count.
```

Your public method `isD()` must return `true` if the `N`-item array passed as an argument represents a derangement, and `false` otherwise. For example, this code should print `false`:

```
int[] test = {9, 8, 7, 6, 5, 4, 3, 2, 1};
StdOut.println(isD(test)); // should print false
```

Your `main()` method must read from standard input an `int` value `N` followed by a sequence of permutations of the integers `1` through `N` (you may assume there will be one or more permutations). Input and examine each permutation so that you can identify and print the first derangement. Continue to examine the permutations in order to identify and count all the derangements. If there are no derangements, the first line of output should not be printed.

Your output should be in the same format as shown in the example.

Sample Runs. Two sample files are available at:

http://www.cs.princeton.edu/~cos126/docs/data/5perms9.txt
http://www.cs.princeton.edu/~cos126/docs/data/1000perms15.txt

```
% more 5perms9.txt
9
1  2  3  4  5  6  7  8  9
9  8  7  6  5  4  3  2  1
2  1  4  3  6  5  8  7  9
8  9  2  1  4  7  5  3  6
3  7  6  9  8  2  1  5  4

% java HatsPart1 < 5perms9.txt
First derangement: 8 9 2 1 4 7 5 3 6
Number of derangements: 2

% java HatsPart1 < 1000perms15.txt
First derangement: 12 8 7 5 9 10 1 14 13 2 4 3 11 15 6
Number of derangements: 358
```

Submission. Submit the single file `HatsPart1.java` via Dropbox at

    https://dropbox.cs.princeton.edu/COS126_F2012/Exam1

Be sure to click the *Check All Submitted Files* button to verify your submission.

Part 2 (16 points). *Be sure that you have successfully submitted* `HatsPart1.java` *before attempting this part of the exam. Read everything before starting to code.*

Assuming that all hats are labeled with owner's names, one way to get your hat back after graduation is to swap hats with the person whose hat you have, continuing until you get your hat. You are always guaranteed to get your hat back that way. For example, in the permutation

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
3	7	6	9	8	2	1	5	4

student `1` would get her hat back by swapping with student `3`, then `6` then `2`, then `7`.

The sequence `1 3 6 2 7` is an example of a *cycle*. The *length* of this cycle is 5 (including 7 back to 1). A person who already has her own hat forms a cycle of length 1. Every permutation consists of a set of 1 or more cycles. In this example, the other cycles are `4 9` and `5 8`, each of length 2.

Your next task is to determine the length of the *longest* cycle in each permutation from the input, and use that to determine the average longest cycle length for the given permutations. For example, in the permutation above the longest cycle length is 5. The lengths of the longest cycles in the five permutations in `5perms9.txt` are 1, 2, 2, 9, and 5, respectively, so the average length of the longest cycle in this set of permutations is 3.8.

Write a program `HatsPart2` with the API shown on the following page:

```
public class HatsPart2

public static int maxCycleLength(int[] a) // return the length of the
                                          // longest cycle in the
                                          // permutation in array a

public static void main(String[] args)// Compute and print the average
                                      // longest cycle length for the
                                      // input permutations.
```

The method `maxCycleLength()` takes as an argument an `N`-item `int` array containing a permutation and returns the longest cycle length, as described above. The `main()` method, using the same input format described in Part 1, computes the average max cycle length among the permutations from standard input, and prints that average as shown below.

Sample Runs. For our two sample files, your program must behave as follows:

```
% java HatsPart2 < 5perms9.txt
Average max cycle length: 3.8

% java HatsPart2 < 1000perms15.txt
Average max cycle length: 9.585
```

Hint. This part is worth 8 points for `maxCycleLength()` (Part 2B) and 8 points for everything else (Part 2A). Since Part 2A is much easier than Part 2B, start with a dummy implementation of `maxCycleLength()` that just returns 1 and then complete the rest of it *after* you have everything else working.

Submission. Submit the single file `HatsPart2.java` via Dropbox at

`https://dropbox.cs.princeton.edu/COS126_F2012/Exam1`

Again, be sure to click the *Check All Submitted Files* button to verify your submission.