

Programming Practice Exam 1

This practice test has 2 programming parts. You have 90 minutes. The exam is open book, open note, and open booksite. You may use code from your programming assignments or the Introduction to Programming in Java booksite. No communication with any non-staff members is permitted. **If this were a real exam, you would have to write out and sign the Honor Code pledge before turning in the test, so you should do so to practice taking this time into account. Write out and sign the Honor Code pledge before turning in the test:**

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

Signature

Problem	Score
0	/1
1	/22
2	/7

Total	
-------	--

Name:

NetID:

Precept:
(circle your precept)

P01	12:30	Dave Pritchard
P01A	12:30	Donna Gabai
P01B	12:30	Pawel Przytycki
P02	1:30	Tom Funkhouser
P02A	1:30	Allison Chaney
P02B	1:30	Pawel Przytycki
P02C	1:30	Vivek Pai
P02D	1:30	Siddhartha Chaudhuri
P03	2:30	Tom Funkhouser
P03A	2:30	Allison Chaney
P04	3:30	Vivek Pai
P04B	3:30	Shilpa Nadimpalli
P05	7:30	Shilpa Nadimpalli
P06	10am	Lennart Beringer
P07	1:30	Dave Pritchard
P07A	1:30	Kevin Lee
P07B	1:30	Siyu Liu
P08	12:30	Donna Gabai
P08A	12:30	Judi Israel
P09	11am	Judi Israel

0. **Cover Page** Write your name and netid, select your precept section, and write out the honor code on the cover (now!) and sign it when you complete the test.

1. Sort

Your first task is to write a Java class in `SortAndDraw.java` whose `main` method implements a *linear*-time filter that takes from standard input a sequence of integers, each between 0 and 99, and prints the sequence in sorted order to standard output. Linear time means that each item on the input should be examined only once.

You may assume that there are not more than $2^{31} - 1$ repetitions of any distinct integer, but may make no further assumption about the number of integers in the set.

Sample Run. Here is our input and output for three sample runs:

```
% more file1.txt
98 2 3 1 0 0 0 3 98 98 2 2 2 0 0 0 2

% java SortAndDraw < file1.txt
0 0 0 0 0 0 1 2 2 2 2 2 3 3 98 98 98

% more file2.txt
99 98 97 96 95 94 93 39 40 41 42 43 44 45 46 47 74 73 72 71 17 18 19 20

% java SortAndDraw < file2.txt
17 18 19 20 39 40 41 42 43 44 45 46 47 71 72 73 74 93 94 95 96 97 98 99

% more file3.txt
0 0 0 99 99 99 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50

% java SortAndDraw < file3.txt
0 0 0 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 99 99 99
```

2. **Draw** Your second task is to expand your class to display the data in a histogram on `StdDraw`. The histogram's x axis should run from 0 to 100 (since 99 is the maximum range of the data), and the y axis should run from 0 to `max+1`, where the symbol `max` stores the maximum number of appearances of any integer in the sequence. You should use these three commands to set up your drawing window (changing `max` to match your chosen variable name):

```
StdDraw.setXscale(0,100);
StdDraw.setYscale(0,max+1);
StdDraw.line(0,0,100,0);
```

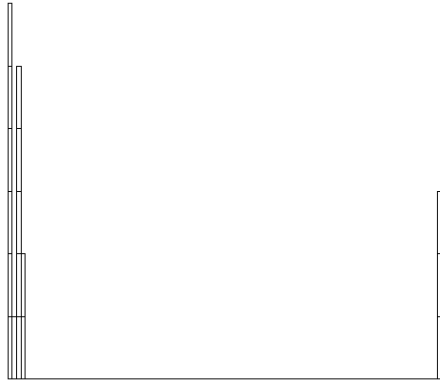
To draw the data in the histogram, you should draw a 1-by-1 black unfilled square to represent each integer in the sequence. The square should be centered at $(x+.5, y+.5)$, where x is the integer, and y is the running count of times that integer appears (starting with 0). For instance, if there are 2 instances of 6 in the sequence, then you should draw a square centered at $(6.5, 0.5)$ and a square centered at $(6.5, 1.5)$.

Recall that `StdDraw.square()` takes three arguments: the x position, the y position, and half the side length, and produces an unfilled square.

Sample Run. Here is our input, output, and drawing for the same three sample runs:

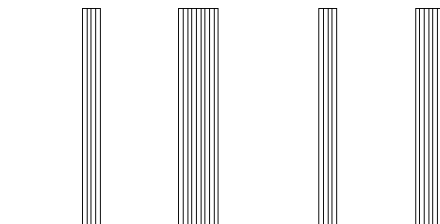
```
% more file1.txt  
98 2 3 1 0 0 0 3 98 98 2 2 2 0 0 0 2
```

```
% java SortAndDraw < file1.txt  
0 0 0 0 0 0 1 2 2 2 2 2 3 3 98 98 98
```

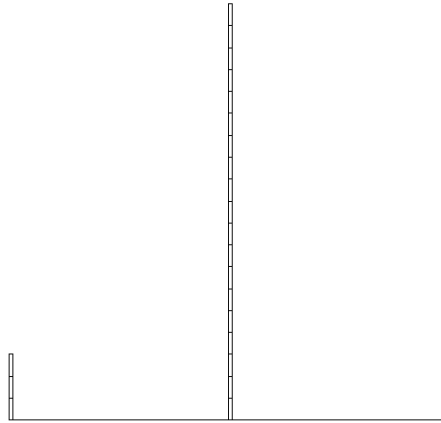


```
% more file2.txt  
99 98 97 96 95 94 93 39 40 41 42 43 44 45 46 47 74 73 72 71 17 18 19 20
```

```
% java SortAndDraw < file2.txt  
17 18 19 20 39 40 41 42 43 44 45 46 47 71 72 73 74 93 94 95 96 97 98 99
```



```
% more file3.txt
0 0 0 99 99 99 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
% java SortAndDraw < file3.txt
0 0 0 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 99 99 99
```



Testing. You can test by either making a small text file of integers like the ones shown above, and using input redirection, or by entering numbers via the keyboard. When you type from the keyboard, you can indicate an end-of-file in Windows Command Prompt by typing `return CTL-z return`. In Mac Terminal an end-of-file is `return CTL-d return`.

Submission. If this were a real exam, you would submit the file `SortAndDraw.java` via Dropbox, the same as you do for your assignments.

As you do with your assignments, click the Check All Submitted Files button to verify your submission when you submit real exam code. This will show you results for a limited set of some very basic test cases. However, this is not available for the practice exam.