NAME: **login ID:**
precept:

COS 126 Midterm 1 Programming Exam, Fall 2009

This part of the exam is like a mini-programming assignment. You will create two programs. For each, compile it, create test data for it and run it on your laptop. Debug it as needed. This exam is open book, open browser. 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 Precept Exam 1 on the Assignments page.

Grading. Each 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.

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 50 minutes to complete the test.

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

Signature

1	/20
2	/10

October 22, 2009

Part 1. Write a Java program `Scores.java` that reads COS 126 grade data from standard input, takes weights from the command line, and computes numerical midterm grades.

The first line of the input has 6 integers giving the maximum scores for the four programming assignments, written exam, and programming exam. Each subsequent line of the input contains a string (student login) followed by 6 integers (grades for 4 programming assignments, written exam grade, programming exam grade). For example, you will find a link to the sample file `input.txt` on the Assignments page of the course website:

```
    20 20 20 20 60 30
pdf 16  8 16 20 30 15
abe 18 19 19 16 49 25
luc 14 15 20 20 59 25
pat 17 18 19 20 58 29
tex 18 19 19 16 49 25
...
```

The weights to be given to the grades come from the command line. For example, programming assignments are weighted equally and count 50% for the midterm grade and the exam counts 50% (with written and programming parts weighted equally), so each program counts 12.5% and each part of the exam counts 25%. Use the weights `12.5 12.5 12.5 12.5 25.0 25.0` on the command line. For these weights, the grade for `pdf` is (16/20)*12.5 + (8/20)*12.5 + (16/20)*12.5 + (20/20)*12.5 + (30/60)*25 + (15/30)*25 = 62.5.

For each assignment or exam, your program will compute a weighted grade by dividing the student grade by the maximum for that assignment, then multiplying by the corresponding weight from the command line. For each student, compute the grade by adding the six weighted grades, then print the student login and the grade (as a percentage, rounded **down** to the nearest integer between 0 and 100). After you have printed the login and midterm grade, there is no reason to keep that student's data. For `input.txt`, you should get the following results:

```
% java Scores 12.5 12.5 12.5 12.5 25.0 25.0 < input.txt
pdf 62
abe 86
luc 88
pat 94
tex 86
...
```

You will need to have `StdIn.java` or `StdIn.class` in your directory in order to compile and test your program.
Submit your program `Scores.java` via the link on the Assignments page.

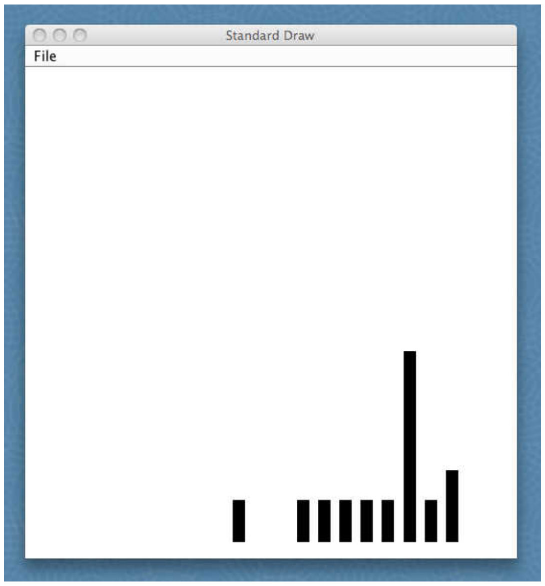
Part 2. Write a Java program `Histogram.java` that reads the output of your program from Part 1 as input and uses `StdDraw` to draw a histogram of the grades.

A histogram is a bar graph where we divide the grades into ranges and plot the number of grades in each range. Your histogram should have 21 bars, for the ranges 0-4, 5-9, 10-14, ... 90-94, 95-99, and 100. You may use `StdDraw.setPenRadius(.025)` to get reasonably wide bars.

For example, in your output from Part 1, you can see that there is 1 score in the ranges 45-49, 60-64, 65-69, 70-74, 75-79, 80-84, 6 scores in the range 85-89, 1 score in the range 90-94, and 2 scores in the range 95-99, so

```
% java Scores 12.5 12.5 12.5 12.5 25 25 < input.txt > avgs.txt
% java Histogram < avgs.txt
```

should produce this histogram:



If you did not get Part 1 working at all, there is a link on the Assignments page of the course website to the sample file `output.txt` which matches the results of Part 1.

You will need to have `StdDraw.java` or `StdDraw.class` in your directory in order to compile and test your program, as well as `StdIn.java` or `StdIn.class.`

Submit your program `Histogram.java` via the link on the Assignments page.