HW 00: Grading on the Curve

We're going to extend our grader code from the lab to compute letter grades using a curve. The idea is that we'll take a collection of grades, compute some statistics about the grades and then determine the letter grades based on those statistics.

The input will be collection of raw grades provided one line at a time. Each line is a collection of grades from one student. Each student's score is computed by dropping the **two** lowest grades and then taking the average.

Compute the average and the standard deviation of the student scores. The letter grades are assigned as follows. If the score is at least one standard deviation above the average, then the grade is an A. If the score is at least the average, then the grade is a B. If the grade is at least one standard deviation below the average then the grade is a C. If the score is at least 2 standard deviations below the mean then the grade is a D. If it's less than that, the grade is an F.

For example, if the average is 82.0 and the standard deviation is 5.5, then all grades 87.5 and above are A's, all grades 82.0 and above are B's, 76.5 and higher gets a C, 71.0 and higher is a D, and below 71.0 is an F.

The program should print the final letter grades, one per line, in the order corresponding to the input order.

Here is a very simple example input.

```
0 100 100 100 100
0   0 100 100 100
```

The output in this case should be the following.

```
B
B
```

Here's a more elaborate example.

```
80 80 80 100 100 72
12 15 60 40 65 35
70 50 80 60 69 45
68 50 82 59 71 45
```

Here's the expected output.

```
A
D
C
B
```

In this case, the average is about 69.9 and the standard deviation is about 14.14.

Here is one more example input.

```
1 2 50
3 4 90
100 2 2
0 0 0
0 70 1
1 100 1
3 70 2
70 1 1
0 30 0
```

```
C
B
A
F
B
A
B
B
D
```

Reading the input

The skeleton code contains code to read in the input. The code makes a list of lists. Each item in `rawscores` is a list of scores for one student.

The code `for line in sys.stdin:` loops over the input one line at a time. The code `scores_as_strings = line.split()` splits the line of input into a list of strings. The input is read as a string, so the code converts each item to a `float`. The code `scores = [float(x) for x in scores_as_strings]` creates a new list, calling `float(x)` for each `x` in the `scores_as_strings` list. Finally, the new list is appended to the list of raw scores.

```
import sys

rawscores = []
for line in sys.stdin:
    # Split the line of numbers into a list of strings.
    scores_as_strings = line.split()
    # Convert the scores to floats.
    scores = [float(x) for x in scores_as_strings]
    rawscores.append(scores)
```

The input is read from something called **standard input**. By default, the standard input is the keyboard. It can get pretty tedious to type in inputs in by hand. If you run your code from a terminal, you can *redirect* stdin from a file. If you put a sample input in a file called `input00.txt`, then you can run the code with that input as follows.

```
$ python curve.py < input00.txt
```

Testing

You will save yourself lots of time by breaking your code into pieces that you can test individually. For example, consider hard coding some inputs in your code as you are developing it. In the end, you will want to compute some examples by hand to test that you are getting the right answers.

What to Submit

You will need to submit both `grader.py` and `curve.py`. Do not submit test code.