ECOO 2012
Programming Contest
Questions

Local Competition (Round 1)

After March 18, 2012

Problem 1: Sluggers

Two important stats in baseball are the team batting average and the team slugging average. Batting average is defined as the total number of hits (this includes 1 base hits, 2 base hits, 3 base hits and home runs combined) divided by the total number of times at bat ("at bats") for all players on the team. The team slugging average is defined using the following equation:

$$sa = \frac{A + 2 \times B + 3 \times C + 4 \times D}{E}$$

Where A is the number of 1 base hits, B is 2 base hits, C is 3 base hits, D is home runs, and E is the number of at bats for all players on the team. Both slugging and batting averages are always presented as decimals rounded to 3 places, leaving off the leading 0 (in theory batting averages can be as high as 1.000, and slugging averages as high as 4.000 but in practice they are both usually well below 1).

DATA11.txt (DATA12.txt for the second try) contains the raw data on the top 10 teams during a regular season of Major League Baseball. The first line is the season name, followed by 10 lines for each of the top 10 teams. Each of these lines starts with a team name (single word) followed by 7 integers: Games Played, At Bats, Runs, Hits (total), two-base hits, three-base hits, and home runs. One space character separates each item on each line.

Write a program to produce a report showing the batting and slugging averages for each team in the order they appeared in the input file, and formatted EXACTLY as shown below, including all punctuation and matching upper and lower case exactly. All spacing is done with a single space character. All batting and slugging averages will be less than 1. The final line shows the same averages for all 10 teams combined (computed from the sum of all at bats, hits, etc. for all 10 teams). Note that the two lines of "=" characters each contain 20 characters.

Sample Input

```
2011 Regular Season
Boston 162 5710 875 1600 352 35 203
NY_Yankees 162 5518 867 1452 267 33 222
Texas 162 5659 855 1599 310 32 210
Detroit 162 5563 787 1540 297 34 169
St.Louis 162 5532 762 1513 308 22 162
Toronto 162 5559 743 1384 285 34 186
Cincinnati 162 5612 735 1438 264 19 183
Colorado 162 5544 735 1429 274 40 163
Arizona 162 5421 731 1357 293 37 172
Kansas_City 162 5672 730 1560 325 41 129
```

Sample Output

```
2011 Regular Season
====================
Boston: .280 .461
NY_Yankees: .263 .444
Texas: .283 .460
Detroit: .277 .434
St.Louis: .273 .425
Toronto: .249 .413
Cincinnati: .256 .408
Colorado: .258 .410
Arizona: .250 .413
Kansas_City: .275 .415
====================
Big 10 Av: .267 .428
```

Problem 2: Decoding DNA

DNA is made up of two twisted strands that encode genes using long combinations of four bases: Adenine, Cytosine, Guanine and Thymine. The strands are complementary to one another, meaning that Adenine and Thymine are always opposite each other, and Cytosine and Guanine are always opposite each other, like this.

A double strand of DNA: ATCAAGGCCTATTCCGGGAAAGG
TAGTTCCGGATAAGGCCCTTTCC

In order for the information in a gene to be used, it has to be transcribed into a strand of RNA. During this process, a portion of one strand of DNA is transcribed – this portion is known as the transcription unit. The start of the sequence to be transcribed is signaled by a sequence of bases known as a promoter, and the end is signaled by a sequence known as the terminator. For our purposes, the promoter is the sequence TATAAT, which begins 10 bases before the start of the transcription unit, and the terminator consists of two distinct, complementary, reversed sequences of at least length 6 that cause the RNA molecule to coil back on itself and pinch off the transcribed strand. If TATAAT appears twice on a strand, only the first occurrence counts as the promoter. An example is shown below.

AGATTA**TATAAT**GATAGGATTTAGATTGACCC**GTCATGCA**AGTCCA**TGCATGAC**AGC

In this example the promoter and terminator sequences are boldfaced, and the transcription unit is underlined. The resulting RNA will be complementary to the transcription unit, except that in RNA Uracil takes the place of Thymine. For this example, the result looks like this:

CCUAAAUCUAACUGGG

DATA21.txt (DATA22.txt for the second try) will contain five single strands of DNA, one on each line. Write a program to output the RNA sequence that results from the transcription process. The sequences should be numbered starting at 1, with a colon and a single space character following the number, as shown below.

Sample Input

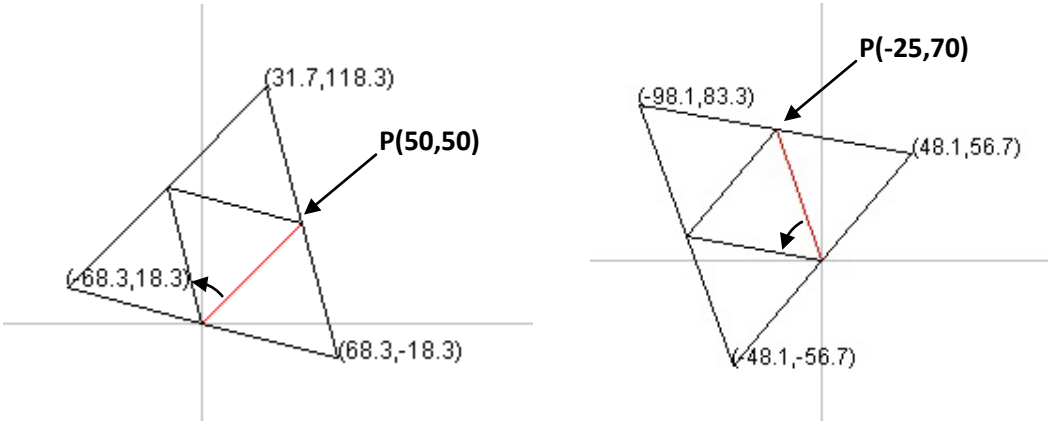
```
AGATTATATAATGATAGGATTTAGATTGACCCGTCATGCAAGTCCATGCATGACAGC
AGTCTTCAAGGGGATTCCCAGGTATATAATGCAGATCGCGACGAAATATCGGGCGGGATCCATACCGACCCAGCCGCCCGA
TATAATGGGGGAGAGACCGAGTGTTTAAGTCCCGAGGGATCGGGAGTGAGATTGAGGGAATTCCGGGAATCTCACT
```

Sample Output

```
1: CCUAAAUCUAACUGGG
2: UAGCGCUGCUUUAU
3: CUCUCUGGCUCACAAAUUC
```

Problem 3: Triangles

Who doesn't love a good equilateral triangle? Equilateral triangles have a lot of symmetry – so much so that if you fix one point of an equilateral triangle to the origin, then all you need is one other corner point to begin generating a tiling of identical equilateral triangles. In the examples below, the points labeled P were given as input, and the large equilateral triangles, each made up of 4 smaller, identically-sized ones, were generated automatically. Note that in these tilings, the third point of the innermost triangle is always counter-clockwise from the given point P, as shown by the arrow.



DATA31.txt (DATA32.txt for the second try) will contain 5 test cases. Each test case consists of two integers $\mathbf{P_x}$ and $\mathbf{P_y}$, the x and y coordinates of point **P**. Your job is to compute the coordinates of the three corners of the large equilateral triangle constructed as described above. These coordinates should be rounded to one decimal place and reported on a single line. Each point should be bracketed and spaced exactly as shown, and there should be a space separating each point on the line. The order in which you report the three points on the line does not matter.

Sample Input

```
50 50
-25 70
-35 -23
35 -23
5 6
```

Sample Output

```
(-68.3,18.3) (68.3,-18.3) (31.7,118.3)
(-48.1,-56.7) (48.1,56.7) (-98.1,83.3)
(37.4,-18.8) (-37.4,18.8) (-32.6,-64.8)
(2.4,41.8) (-2.4,-41.8) (72.4,-4.2)
(-7.7,1.3) (7.7,-1.3) (2.3,13.3)
```

Problem 4: Boggled

In the game of Boggled, players are given a 4x4 array of letters and about 2 minutes to list as many words as they can find in the array of letters. A word only counts if it is 3 or more letters long and you can trace at least one path through the letters to spell the word without using any letters twice.

In the example at left (depicting the offline version of the game) the two styles of arrows trace the words SALT and MACE.

For the purposes of scoring, 3- and 4-letter words are worth 1 point, 5-letter words are worth 2 points, 6-letter words are worth 3 points, 7-letter words are worth 4 points, and anything longer is worth 11 points. Paths can travel in any direction, including diagonally. There is no penalty for listing words that are too short or not present on the board and there is no penalty for listing the same word twice, but none of these are worth any points. (Of course in the real game, words would also have to be found in some standard dictionary, but we're not going to worry about that here.)

DATA41.txt (DATA42.txt for the second try) will contain 5 test cases. The first 4 lines of each test case will be a Boggled board, followed by an integer n on a line by itself, and then a list of n words that represent a single player's turn. Your program must output that player's score along with the number of legal and illegal words in the list. Illegal words are either not found on the board, too short, or repetitions of words higher up in the list. If a word is illegal for more than one reason, it is recorded once in the most important category of illegal words, with too short being most important, followed by repetition, and then not found. The format of the output must be exactly as shown below.

Sample Input

```
HIWH
PXBY
ASQK
PVES
15
SAX
SAP
WHIP
WHY
PAP
PAX
VASE
VASES
BY
HIP
HIPS
SAVE
SAVES
PAVE
PAVES
LSUN
DAGE
EBUI
WERJ
37
SUN
SUNG
LANGUAGE
SAGE
BAG
SAG
BEE
WED
WEDS
BED
BEDS
BALD
BALDS
BUG
BUGS
LAGS
LAG
SAD
BEAD
BEADS
EAD
EADS
LAD
LADS
GUN
GAS
BAG
BAGS
BAD
BADE
JIG
JIGS
JIGSAW
RIG
RIGS
RUG
RUGS
```

Sample Output

```
Your score: 16 (13 good, 1 not found, 1 too short, 0 repeated)
Your score: 36 (34 good, 2 not found, 0 too short, 1 repeated)
```