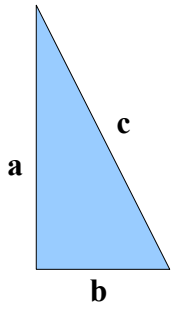
CS 218 – MIPS Assignment #1

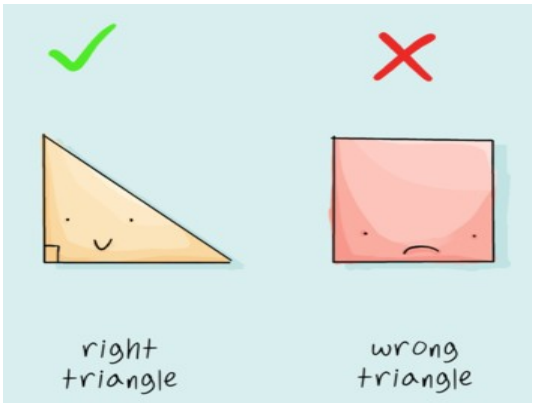
Purpose: Become familiar with RISC Architecture concepts, the MIPS Architecture, and SPIM (the MIPS simulator).

Points: 45

Assignment



Write a MIPS assembly language program to calculate the semi-perimeter for each right triangle in a series of right triangles. The sides should be read from word-sized ***aSides[]***, ***bSides[]***, and ***cSides[]*** arrays. The result must be stored into the word-sized ***semiPerims[]*** array.



The formula for the semi-perimeter of a right triangle is as follows:

$$semiPerim[n] = \frac{aSides[n] + bSides[n] + cSides[n]}{2}$$

After the semi-perimeters are computed, the program should find the minimum, maximum, middle value, sum, and average for the right triangle semi-perimeters. The provided data is sorted and will remain so after the calculations. *Note,* for an odd number of items, the middle value is defined as the middle value. For an even number of values, it is the integer average of the two middle values.

The program must display the results to the console window with 8 numbers per line. The output should look something like the following (with the correct answers displayed):

```
MIPS Assignment #1
Program to calculate the semi-perimeter of each right triangle in a
 series of right triangles.  Also finds min, mid, max, sum, and average
 for the semi-perimeters.

  104   110   117   124   131   139   147   153
  162   174   194   205   215   218   231   237

      . . .   output truncated  . . .


Semi-Perimeters min = ?
Semi-Perimeters mid = ?
Semi-Perimeters max = ?
Semi-Perimeters sum = ?
Semi-Perimeters ave = ?
```

Submission

- All source files must assemble and execute with QtSpim/SPIM MIPS simulator.
- Submit source file
  - Submit a copy of the program source file via the on-line submission
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time (at a maximum rate of 5 submissions per hour).
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given assignment. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All source files must include your name, section number, assignment, NSHE number, and program description. The required format is as follows:

```
#  Name: <your name>
#  NSHE ID: <your id>
#  Section: <section>
#  Assignment: <assignment number>
#  Description: <short description of program goes here>
```

Failure to include your name in this format will result in a reduction of points.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Assemble	-	Failure to assemble will result in a score of 0.
Program Header	3%	Must include header block in the required format (see above).
General Comments	7%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	90%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Data Declarations

Use the following data declarations:

```
aSides:   .word     121,  123,  131,  139,  141,  149,  153,  157,  163,  169
          .word     201,  207,  212,  215,  223,  227,  231,  236,  241,  245
          .word     251,  252,  262,  264,  271,  273,  287,  289,  293,  299
          .word     301,  305,  312,  315,  326,  328,  332,  337,  341,  343
          .word     401,  408,  411,  413,  421,  424,  431,  434,  445,  448
          .word     453,  454,  460,  462,  474,  475,  486,  487,  491,  492
          .word     501,  504,  515,  517,  524,  525,  535,  537,  543,  548
          .word     551,  553,  563,  567,  577,  579,  582,  588,  593,  595

bSides:   .word      75,   81,   83,   87,   89,   91,   94,   97,   99,  101
          .word     107,  111,  120,  121,  137,  141,  157,  167,  177,  181
          .word     191,  199,  202,  209,  215,  219,  223,  225,  231,  242
          .word     244,  249,  251,  253,  266,  269,  271,  272,  280,  288
          .word     291,  299,  301,  303,  307,  311,  321,  329,  330,  331
          .word     332,  351,  376,  387,  390,  400,  411,  423,  432,  445
          .word     469,  474,  477,  479,  482,  484,  486,  488,  492,  493
          .word     557,  587,  599,  601,  623,  624,  625,  626,  627,  628

cSides:   .word      13,   17,   21,   23,   33,   39,   47,   53,   63,   79
          .word      81,   93,   99,  100,  103,  107,  109,  111,  121,  127
          .word     132,  137,  142,  149,  154,  161,  167,  178,  186,  197
          .word     206,  212,  222,  231,  246,  250,  254,  278,  288,  292
          .word     303,  315,  321,  339,  348,  359,  362,  374,  380,  391
          .word     400,  404,  406,  407,  424,  425,  426,  429,  448,  492
          .word     501,  513,  524,  536,  540,  556,  575,  587,  590,  596
          .word     634,  652,  674,  686,  697,  704,  716,  720,  736,  753
          .word     782,  795,  807,  812,  817,  827,  837,  839,  841,  844


semiPerims:
          .space    320

length:   .word     80

sMin:     .word     0
sMid:     .word     0
sMax:     .word     0
sSum:     .word     0
sAve:     .word     0
```

Note, the **`.space 320`** directive reserves 320 bytes (which will store 80 words).