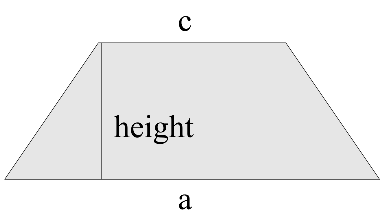
CS 218 – MIPS Assignment #1

Purpose: Become familiar with RISC Architecture concepts, the MIPS Architecture, and QtSpim (the MIPS simulator).
Points: 45

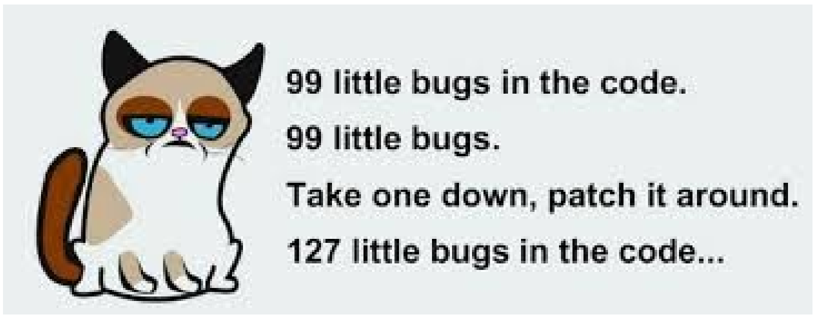
Assignment:

Write a MIPS assembly language program to calculate the area of each trapezoid in a set of trapezoids. The sides should be read from word-sized ***aSides*** , ***cSides***, and ***heights*** arrays. The result must be stored into the word-sized ***tAreas*** array. Below is the formula to calculate the area of a trapezoid:



$$tAreas[n] = \left(heights[n] \times \frac{(aSides[n] + cSides[n])}{2} \right)$$

After all the trapezoid areas have been calculated, the program should find the minimum, middle value, maximum, sum, and average for the trapezoid areas array.


99 little bugs in the code.
99 little bugs.
Take one down, patch it around.
127 little bugs in the code...

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

```
MIPS Assignment #1
  Program to calculate area of each trapezoid in a series of trapezoids.
  Also finds min, med, max, sum, and average for the trapezoid areas.

    1    4    9    16    25    36    49    64
    81    100    2346    4294    6588    9035    10656    12684
    14651    18512    21390    24231    31668    35690    39559    44454
    49848    53872    56330    62746    66920    71295    106800    113120

    [... display all numbers ...]

Trapezoid areas min = 1
Trapezoid areas med = ?
Trapezoid areas max = ?
Trapezoid areas sum = ?
Trapezoid areas 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.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. 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       1,    2,    3,    4,    5,    6,    7,    8,    9,   10
          .word      15,   25,   33,   44,   58,   69,   72,   86,   99,  101
          .word     107,  121,  137,  141,  157,  167,  177,  181,  191,  199
          .word     202,  209,  215,  219,  223,  225,  231,  242,  244,  249
          .word     251,  253,  266,  269,  271,  272,  280,  288,  291,  299
          .word     369,  374,  377,  379,  382,  384,  386,  388,  392,  393
          .word    1469, 2474, 3477, 4479, 5482, 5484, 6486, 7788, 8492, 1493

cSides:   .word       1,    2,    3,    4,    5,    6,    7,    8,    9,   10
          .word      32,   51,   76,   87,   90,  100,  111,  123,  132,  145
          .word     206,  212,  222,  231,  246,  250,  254,  278,  288,  292
          .word     332,  351,  376,  387,  390,  400,  411,  423,  432,  445
          .word     457,  487,  499,  501,  523,  524,  525,  526,  575,  594
          .word     634,  652,  674,  686,  697,  704,  716,  720,  736,  753
          .word    1782, 2795, 3807, 3812, 4827, 5847, 6867, 7879, 7888, 1894

heights:
          .word       1,    2,    3,    4,    5,    6,    7,    8,    9,   10
          .word     102,  113,  122,  139,  144,  151,  161,  178,  186,  197
          .word     203,  215,  221,  239,  248,  259,  262,  274,  280,  291
          .word     400,  404,  406,  407,  424,  425,  426,  429,  448,  492
          .word     501,  513,  524,  536,  540,  556,  575,  587,  590,  596
          .word     782,  795,  807,  812,  827,  847,  867,  879,  888,  894
          .word    1912, 2925, 3927, 4932, 5447, 5957, 6967, 7979, 7988, 1994

tAreas:   .space   280

len:      .word    70

taMin:    .word    0
taMid:    .word    0
taMax:    .word    0
taSum:    .word    0
taAve:    .word    0
```

Note, the `.space 280` directive reserves 280 bytes which will store 70 words.