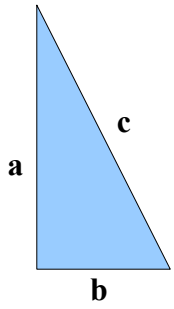
CS 218 – MIPS Assignment #2

Purpose: Become familiar with RISC Architecture concepts, the MIPS Architecture, and SPIM (the MIPS simulator).
Points: 50

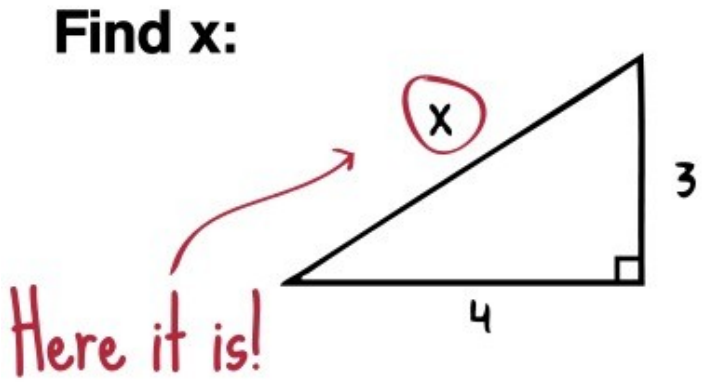
Assignment



Write a MIPS assembly language program to calculate the area of each right triangle in a series of right triangles.

The area of a right triangle is computed as follows:

$$areas[i] = \frac{aSides[i] \times bSides[i]}{2}$$



Once the values are computed, the program should sort the right triangle areas array in ascending order (small to large) using the insertion sort algorithm as follows:

```
insertionSort(array arr) {
   for i = 1 to length-1 do  {
      value = arr[i];
      j = i - 1;
      while((j ≥ 0) and (arr[j] > value)){
          arr[j+1] = arr[j];
          j = j - 1;
      }
      arr[j+1] = value;
   }
}
```

Submissions not based on this algorithm will not be scored.

After the data has been sorted, the program should find the minimum, maximum, median, sum, and average for the computed areas. 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 should display the results to the console window. The output should look something like the following (with the correct answers displayed and 8 numbers per line):

```
MIPS Assignment #2
  Right Triangle Areas Program:
  Also finds minimum, middle value, maximum, sum,
  and average for the areas.

  12656  12768  12820  12870  13162  13208  13572  13779
  13826  13937  13986  14022  14135  14278  14383  14384

      . . .   output truncated  . . .


Areas Minimum = ?
Areas Median  = ?
Areas Maximum = ?
Areas Sum     = ?
Areas Average = ?
```

Note, no additional code was required to justify the numbers. The formatting is a result of all numbers having the same digit count.

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.

MIPS Assignment #2 – Data Declarations

Use the following data declarations:

```
aSides:     .word     233,   214,   273,   231,   255
            .word     264,   273,   274,   223,   256
            .word     244,   252,   231,   242,   256
            .word     255,   224,   236,   275,   246
            .word     253,   223,   253,   267,   235
            .word     254,   229,   264,   267,   234
            .word     256,   253,   264,   253,   265
            .word     226,   252,   257,   267,   234
            .word     217,   254,   217,   225,   253
            .word     223,   273,   235,   261,   259
            .word     225,   224,   263,   247,   223
            .word     234,   234,   256,   264,   242
            .word     236,   252,   232,   231,   246
            .word     250,   254,   278,   288,   292
            .word     282,   295,   247,   252,   257
            .word     257,   267,   279,   288,   294
            .word     234,   252,   274,   286,   297
            .word     244,   276,   242,   236,   253
            .word     232,   251,   236,   287,   290
            .word     220,   241,   223,   232,   245

bSides:     .word     157,   187,   199,   111,   123
            .word     124,   125,   126,   175,   194
            .word     149,   126,   162,   131,   127
            .word     177,   199,   197,   175,   114
            .word     164,   141,   142,   173,   166
            .word     104,   146,   123,   156,   163
            .word     121,   118,   177,   143,   178
            .word     112,   111,   110,   135,   110
            .word     127,   144,   210,   172,   124
            .word     125,   116,   162,   128,   192
            .word     117,   114,   115,   172,   124
            .word     125,   116,   162,   138,   192
            .word     121,   183,   133,   130,   137
            .word     142,   135,   158,   123,   135
            .word     127,   126,   126,   127,   227
            .word     177,   199,   177,   175,   114
            .word     194,   124,   112,   143,   176
            .word     134,   126,   132,   156,   163
            .word     124,   119,   122,   183,   110
            .word     191,   192,   129,   129,   122

rtAreas:    .space    400

len:        .word     100

rtMin:      .word     0
rtMid:      .word     0
rtMax:      .word     0
rtSum:      .word     0
rtAve:      .word     0
```

Note, the **`.space 400`** directive reserves 400 bytes which will be used to store 100 words.