The first program is written in C, and creates a binary file, along with a text file with some statistics in it about the binary file.

The second program is an assembly program which will include the binary file you created with your first program. Your assembly program will compute some values using your binary file, and then print the results. The results will be compared to your first program’s text file stats output to check for correctness.

Your binary file will contain N(0 < N < 50) sections of data, where N specified by the user (ie. How many sections of data do you want?). Each section will consist of three chunks of data. The first chunk will be a random (100 < NUM\_BYTES < 500) number of bytes, followed by a random (100 < NUM\_SHORTS < 500) number of shorts, followed by a random (100 < NUM\_LONGS < 500) number of longs. The byte values themselves will be unsigned and random (50 < BYTEVAL<150) . The short values will also be random (1000 < SHORTVAL < 2000). The long values will also be random (3000 < LONGVAL < 5000). The range of NUM\_BYTES, LONGS, and SHORTS should be provided in an input text file (config.txt) that your C program reads. In class, I will provide input for your program – for example, I will say “OK, make your binary files, and make NUM\_BYTES be between 100 and 100 (ie, 100)”, etc., and you will create the binary on the fly.

Your stats file will show the number of pieces of data in each chunk, along with the average value, rounded to 2 decimal places. An example output might look like:

User chose 35 sections.  
Section 1  
238 bytes, average value 101.73  
401 shorts, average value 1487.25  
104 longs, average value 4345.32  
…  
Section 35  
123 bytes, average value 101.32  
481 shorts, average value 1581.15  
342 longs, average value 4125.12

The assembly program will then be run and ask the user for a section number to use in its computations. The assembly program will spit out the averages for the three chunks in the section specified by the user. The values should match the text output given by the C program.

FAQ  
How should we approach this problem?  
Carefully. Get your program running without randomness first. Hand pick the values in the chunks and check your math, especially the rounding. Then add the randomness.  
Are there any prizes?  
Yes, we will time your assembly programs by asking each person to tweak their random ranges so that each student has the same number of elements in their chunks. That way you can customize your binary file to make your assembly program run faster if desired. Winner will get the standard prize.