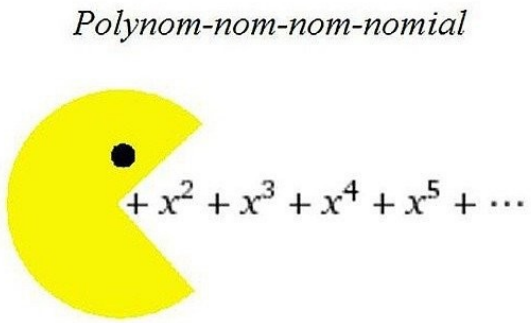
CS 218 – Assignment #4

Purpose: Learn to use arithmetic instructions, control instructions, compare instructions, and conditional jump instructions.
Points: 75

Assignment:

Write a simple assembly language program to find the minimum, estimated median value, maximum, sum, and integer average of a list of numbers. Additionally, the program should also find the sum, count, and integer average for the odd numbers. The program should also find the sum, count, and integer average for the numbers that are evenly divisible by 9. Do **not** change the data types (double-words) as defined below.



Declare the values:

```
lst         dd      4224, 1116, 1542, 1240, 1677, 1635, 2420, 1820, 1246, 1333
            dd      2315, 1215, 2726, 1140, 2565, 2871, 1614, 2418, 2513, 1422
            dd      1119, 1215, 1525, 1712, 1441, 3622, 1731, 1729, 1615, 2724
            dd      1217, 1224, 1580, 1147, 2324, 1425, 1816, 1262, 2718, 1192
            dd      1435, 1235, 2764, 1615, 1310, 1765, 1954, 1967, 1515, 1556
            dd      1342, 7321, 1556, 2727, 1227, 1927, 1382, 1465, 3955, 1435
            dd      1225, 2419, 2534, 1345, 2467, 1615, 1959, 1335, 2856, 2553
            dd      1035, 1833, 1464, 1915, 1810, 1465, 1554, 1267, 1615, 1656
            dd      2192, 1825, 1925, 2312, 1725, 2517, 1498, 1677, 1475, 2034
            dd      1223, 1883, 1173, 1350, 2415, 1335, 1125, 1118, 1713, 3025
length      dd      100

lstMin      dd      0
estMed      dd      0
lstMax      dd      0
lstSum      dd      0
lstAve      dd      0

oddCnt      dd      0
oddSum      dd      0
oddAve      dd      0

nineCnt     dd      0
nineSum     dd      0
nineAve     dd      0
```

You may declare additional variables if needed. All data is *unsigned*. As such, the DIV/MUL would be used (not IDIV/IMUL). The JA/JAE/JB/JBE must be used (as they are for unsigned data).

Since the list is not sorted, we will estimate the median value. Since the list length is even, the estimated median will be computed by summing the first, last, and two middle values and then dividing by 4. Be sure to divide the length by 2 (and not hard-code the indexes). There will be a 10% penalty for hard-coding indexes.

Note, no template is provided. Create the program source file, **`ast04.asm`**, based on the previous assignments.

Submission:

- All source files must assemble and execute on Ubuntu with **`yasm`**.
- Submit source files
  - 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 loss of up to 5%.

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	5%	Must include header block in the required format (see above).
General Comments	10%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	85%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Debugger Commands:

Due to the looping, when debugging assignment #4, you should learn to set breakpoints within the program.

Create an input file for the debugger. Some useful commands might include:

```
x/100dw &lst
x/uw &length
x/uw &lstMin
x/uw &estMed
x/uw &lstMax
x/uw &lstSum
x/uw &lstAve
x/uw &negCnt
x/uw &negSum
x/uw &negAve
x/uw &nineCnt
x/uw &nineSum
x/uw &nineAve
```

The commands should be placed in a file (such as '**a4in.txt**') so they can be read from within the debugger. The debugger command to read a file is "**source <filename>**". For example, if the command file is named '**a4in.txt**',

```
(gbd) source a4in.txt
```

Refer to the debugger input files from the previous assignments for examples. This will include outputting the results to a file.