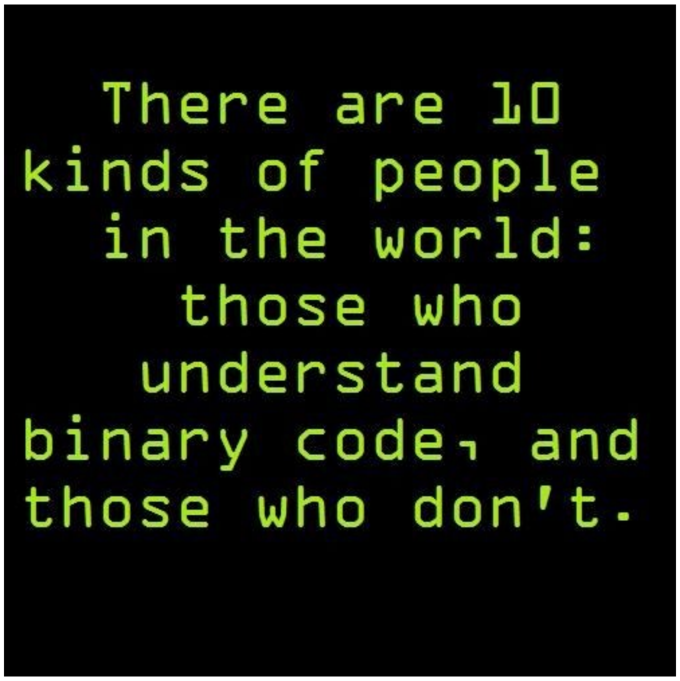
CS 218 – Assignment #9

Purpose:	Learn assembly language functions and standard calling convention. Additionally, become more familiar with program control instructions, functions handling, and stacks.
Due:	Wednesday (6/22)
Points:	175

Assignment:

Write the assembly language functions described below. You will be provided a C++ main program that calls the following functions.

- Void function, **bubbleSort()**, modified to sort the numbers into ascending order (small to large).
- Void function, **simpleStats()**, to find the minimum, median, and maximum for a list of numbers. *Note,* for an odd number of items, the median value is defined as the middle value.
- Value returning function, **iAverage()**, to computer and return the integer average for a list of numbers.
- Value returning function, **lstStats()**, to compute and return the variance and standard deviation for a list of numbers.



There are 10 kinds of people in the world: those who understand binary code, and those who don't.

In addition, write a function **readBinaryNumber()** that will read a binary number, in ASCII format, from the user. The routine should use the system service for reading data from the keyboard (into a buffer), call a routine to convert the ASCII input (from the buffer) into an integer, and return the integer. The number must be between 0 and MAXNUM (defined constant). The function should re-prompt for invalid/incorrect input. When the end of input is received (a return with no characters on the line), the function should return a NOSUCCESS code (indicating no more input).

In addition, the **readBinaryNumber()** function should call the **aBin2int()** function to convert the binary string to an integer. This function will be used on a future assignment.

All functions should use the stack for local variables. No static variables should be declared! All provided data items are *unsigned* integers (MUL and DIV instructions). The functions must be in a separate assembly file. The files will be assembled individually and linked together. Refer to the text for more information regarding functions.

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.
    - Note, only the functions file (**`ast9procs.asm`**) will be submitted.
- 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 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 loss of up to 3%.

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.

<u>Compile, Assemble, and Linking Instructions</u>
You will be provided a C++ main function that calls the functions. Your functions should be in a separate file (**`ast9procs.asm`**). The files will be compiled/assembled individually and linked together.

When compiling, assembling, and linking the files for assignment #9, use the provided **makefile** to assemble, and link. *Note*, **only** the functions file, **`ast9procs.asm`**, will be submitted. The submitted functions file will be assembled and linked with the provided main. As such, do not alter the provided main.

<u>Example Execution:</u>
The following is an example execution demonstrating various error handling:

```
ed-vm% ./main
--------------------------------------------------
CS 218 - Assignment #9

Enter Binary Number: 1000
Enter Binary Number: 1001
Enter Binary Number: 1010
Enter Binary Number: 1011
Enter Binary Number: 1100
Enter Binary Number: 0001
Enter Binary Number: 0010
Enter Binary Number: 0011
Enter Binary Number: two
Error, re-enter: 010
Enter Binary Number: 10102
Error, re-enter: 1010101
Enter Binary Number:
000101010101010101010101010101010101010101010101010101010
Error, re-enter: 100101
Enter Binary Number: 0101x10
Error, re-enter: 1010100
Enter Binary Number:             10101
Enter Binary Number: 00000000000000000011
Enter Binary Number:

--------------------------------------------------
Program Results

Sorted List:
         1           2           2           3           3
         8           9          10          11          12
        21          37          84          85

Statistical Results:
   Length        =  14
   Minimum       =  1
   Maximum       =  85
   Median        =  9
   Average       =  20
   Variance      =  10708
   Std Deviation =  27.656051

ed-vm%
```