Electrical and Computer Engineering Department
University of Maryland
College Park, MD 20742-3285

Glenn L. Martin Institute of Technology ♦ A. James Clark School of Engineering

Dr. Charles B. Silio, Jr.
Telephone 301-405-3668
Fax 301-314-9281
silio@umd.edu

Programming Project 1
(Due: Mon., Mar. 2, 2009)

Write debug and test a C-language main program called proj1.c that processes an arbitrary number of sets of data with an arbitrary number of elements in each set and the specified output for each set. The input data consist of a file named **proj1_data** and if the executable load module for your program were compiled using the command **gcc -Wall proj1.c -o proj1**, then you would execute your program in the following way: **./proj1 < proj1_data** thus directing the program to read the file **proj1_data** as its input.

proj1_data is constructed in the following way: The first set of data consists of a set (positive or negative) decimal integers each terminated with a linefeed, much in the same way that data are input from the keyboard to the program array.c (in fact, array.c is a good starting point, but must be modified and added to significantly). We will define the SIZE of the array to be read in as **#define SIZE 20** so that if more than 20 decimal numbers are in a particular data set, you are to detect that fact, print out an error message noting that and then continue processing by looking for the start of the next set which immediately follows an asterisk *; you will find either an asterisk or an end-of-file (EOF) character. Following the asterisk is a new data set of decimal numbers to be processed. The occurrence of an EOF symbol indicates that your task is finished and that you should complete processing of the last data set, print out the results, and then return (return 0) to the operating system. Sample test data is found in the file **proj1_data**.

Your output should appear as follows (which shows output for only the first set of data):

```
Set Number: 1
Entered integer number 0: was 36
Entered integer number 1: was 27
Entered integer number 2: was 55
Entered integer number 3: was -3
Entered integer number 4: was 42
Entered integer number 5: was 66
Entered integer number 6: was -5
The sum is: 218
The average is: 31.1429
The sorted list is 0: -5
The sorted list is 1: -3
The sorted list is 2: 27
The sorted list is 3: 36
The sorted list is 4: 42
The sorted list is 5: 55
The sorted list is 6: 66

Set Number 2:
   etc.
```

In other words the output is to identify the data set number, print out the array elements in the order they were read and their sum (ala array.c), and in addition, their average (e.g., the float of sum divided by the float of count where count is the count of the number of validly entered decimal integers forming the sum); then sort the array elements into ascending order (e.g. using bubble sort).

You will need to test if scanf has read a correct entry by using **n = scanf("%d",...)** and then testing n to see if n==1 (in which case a decimal integer was read), or if n == EOF (in which case cntrl-D was read indicating end of file), or you might need to read the input again as a character e.g. n=scanf("%c", ...) (and if n=1 this time check to see that the character is an asterisk '*'). If the count of decimal digits read in exceeds 20, then don't calculate a sum or average, or sort the input; instead print an error message and keep reading without storing the remaining decimal integers in that set until an asterisk is found. Then identify and start processing a new set of decimal integer data.

-continued-

The following code segment may assist you in figuring out how to write a bubble-sort:

```
int array[SIZE] , int n;
int this, next, temp;
for( this = 0; this < n ; this ++ )             /* outer for loop */
 for( next = this + 1; next < n ; next ++ )     /* inner for loop */
  if( array[ this ] > array[ next ] )           /* if out of order  */
  { temp = array[ this ] ;
    array[ this ] = array[ next ] ;             /* swap elements */
    array[ next ] = temp ;
  }
```