CSI 116 91 – Introduction to Programming

Spring 2017

Final Project

Your task is to write a fully functional C program and present it to me on the last day of class before the Final Exam. After handing in your source code, I will compile it on my computer and check the results for different inputs.
Please seek help as soon as you can. Your program will be graded only if it compiles. I will not review the source code, so I won't be able to tell you what is wrong with it.

1 Description

Your program should read floating point values from the input, up to (and including) 100 values. You should then print the median of the values read by your program.
Your program should stop reading input values if

1. It reads -999 from the input or
2. It reads 100 values from the input.

If the number of values entered by the user is equal to 100, you should stop when reading the 100th value from the input and calculate the median.
Your program *must* not print any instructions to the screen. It should only read floating point values until a value equal to -999 is presented by the user or until 100 values are read from the input.
Sample run:

```
32 2 17 -4 87.45 1 98.65 -999
Median is: 17
```

2 What is the median?

In statistics and probability theory, a median is the value separating the higher half of a data sample, a population, or a probability distribution, from the lower half. The median of a finite list of numbers can be found by arranging all the observations from lowest value to highest value and picking the middle one (e.g., the median of 3, 3, 5, 9, 11 is 5). If there are an even number of observations, then there is no single middle value; the median is then usually defined to be the mean of the two middle values (the median of 3, 5, 7, 9 is $(5 + 7) / 2 = 6$), which corresponds to interpreting the median as the fully trimmed mid-range. The median is of central importance in robust statistics, as it is the most resistant statistic, having a breakdown point of 50%: so long as no more than half the data are contaminated, the median will not give an arbitrarily large or small result[1].
The following example has two lists, A and B. List A has 15 values. According to the description above, we first sort the list and find the median by extracting the value in the middle of the list, on position 8. List B has 6 values (an even number of values) and we can't find a value exactly in the middle. Following the description above, we have to pick the two middle values and calculate the average between them, which is $(6 + 8)/2 = 7$.

[1] https://en.wikipedia.org/wiki/Median

List A	0	1	1	1	2	2	3	**3**	3	3	3	8	14	22	23	$\xrightarrow{\text{position } 8}$	**3**
Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
List B	1	4	**6**	**8**	10	14										$\xrightarrow{(6+8)/2}$	**7**

3 And how do we sort the input values?

Sorting is an extremely important problem in Computer Science. Many problems found on a daily-basis can be solved by sorting. You can choose the sorting algorithm you want, but I strongly suggest you use the one below or the one in the book (it's somewhere in Chapter 7).
If you decide to use the sorting algorithm below, copy it into your program and call the `bubble_sort` function to sort the elements of the array after you finished reading the numbers from the input. Then you can extract the elements to compute the median.

```
void bubble_sort(float list[], int n)
{
   float c, t;
   int d;

   for (c = 0 ; c < ( n - 1 ); c++)
   {
      for (d = 0 ; d < n - c - 1; d++)
      {
         if (list[d] > list[d+1])
         {
            /* Swapping */
            t         = list[d];
            list[d]   = list[d+1];
            list[d+1] = t;
         }
      }
   }
}
```