Homework 3

100 Points

Pointers, Arrays and Dynamic Allocation of Memory

Project: TEST SCORES STATISCTICS

Write a program that creates and sorts dynamically allocated arrays. The program performs the following steps:

- Read integers from `scores.txt` into a dynamically allocated array. The first number in `scores.txt` represents the number of students, followed by the students' scores (see next pages)
- Sort the array in ascending order using Insertion Sort.
- Display the sorted array 10 numbers per line.

  HINT: Write a function that takes 3 parameters, as shown below:
  `printArray(scores, size, 10);`
- Calculate and display the following:
  - Average
  - Largest score
  - Lowest score
  - Standard deviation // see next page

Honors Project: Modify the above project to read from file name-score pairs. For each student taking a test, the file contains the name followed by the test score on the next line. When the sorted list of scores is displayed, each student's named should be displayed along with his or her score. Read integers from `scores_h.txt` into two dynamically allocated arrays. Display the sorted array 3 pairs name-score per line.

HINT: Write a function that takes 4 parameters, as shown below:
`printArray(scores, names, size, 3);`

Grading:

(15 Points) Read integers from `scores.txt` into a dynamically allocated array using a pointer instead of an index // see example on the next page
(10 Points) Insertion Sort using pointer
(10 Points) Display the sorted array 10 numbers per line using a pointer
(10 Points) Calculate average using a pointer
(10 Points) Calculate lowest using a pointer
(10 Points) Calculate highest using a pointer
(15 Points) Calculate standard deviation using a pointer
(10 Points) Display results: average, lowest, highest, standard deviation
(10 Points) main() – calls all of the above (8 FUNCTIONS)

UPLOAD one .zip file that contains the source file (screen output included), and the output file: **sorted.txt**

Example:

```
// A - print an array using an index
int i;

for (i = 0; i < size; i++)
    cout << list[i];


// B - print an array using a pointer
int *pCurr;
int *pLast = list + size - 1;

for (pCurr = list; pCurr <= pLast; pCurr++)
    cout << *pCurr;
```

To practice pointers, in this assignment you will use the second style (**B**).

STANDARD DEVIATION

To calculate the standard deviation of n scores, first you have to calculate the average of the n scores:

$avg = (s_1 + s_2 + s_3 + \ldots + s_n) / n$

Then you have to calculate the following sum:

$sum = (s_1 - avg)^2 + (s_2 - avg)^2 + (s_3 - avg)^2 +\ldots+ (s_n - avg)^2$

Finally, the standard deviation can be calculated now as the square root of
`sum / (n-1).`

Create the input file **`scores.txt`** using the following scores:

```
46
85 100 90 85 97 92 72 88 79 86 97 89 67 96 84 96 93 77 56 77 85 100 84
 92 88 67 97 86 95 94 73 68 76 80 99 78 87 96 85 64 93 81 92 93 74 65
```

HONORS:
Create the input file **`scores_h.txt`** using the following scores:

```
11

Ron Palmer
85
Becky Warren
100
Joe Looney
90
Geri Palmer
85
Lynn Presnell
97
Holly Gaddis
92
Sam Wiggins
72
Bob Kain
88
Tim Haynes
79
Warren Gaddis
86
Jean James
97
```