Dayata, Wayne Matthew A. February 14, 2021

**CIS 1201 Assignment 3: Problem Involving Arrays**

**Problem Title:** Class Standings

**Introduction/Scenario:** Your Math teacher likes to monitor the performance of her classes in the latest assessment she gave. With great concern of her class, she does not want to stop at listing her students’ scores, but then she also wants to determine a few more interesting statistics behind the scores to know the overall status as a class and use these to compare with the other classes she handles. However, she finds these computations too lengthy to be manually done, so she asks for your help to ease her job by providing her a program that computes these.

**Task:** Given the number of students and their corresponding scores followed by the passing score of the test, construct the code of function getStats() that computes for and returns the following values (in order) in a set: mean score, median score, variance, and percentage of students passed.

Note: Try making the code efficient by limiting the number of operations and traversals.

Remarks: In my solution, the array is first sorted in ascending order (using bubble sort – pushing large elements to the right) to obtain the median while summing up the elements for the mean. In the next loop/traversal, the variance will then be computed at the same time searching the array to obtain the starting index containing the passing score. Other searching/sorting algorithms can be implemented.

|  |  |  |
| --- | --- | --- |
| **Function Header:** | **float\* getStats (int size, int A[], int pass)** | |
| **Sample Function Call:** | **int numStud = 6, passing = 45;**  **int scores[] = {51, 36, 49, 43, 60, 46};**  **float\* result = getStats (numStud, scores, passing);** | |
| **Initial State of Execution Stack:** | | |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | size |  | A |  | pass | | 6 |  | A100 |  | 45 | | D100 |  | D200 |  | D300 | | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | numStud |  | scores |  |  |  | | 6 |  | A100 | 0 | 51 | A100-A103 | | B200 |  | B100 | 1 | 36 | A104-A107 | |  |  |  | 2 | 49 | A108-A10B | | passing |  | result | 3 | 43 | A10C-A10F | | 45 |  |  | 4 | 60 | A110-A113 | | B300 |  | B400 | 5 | 46 | A114-A117 | | | | **A.R. of getstats()**  **A.R. of main()** |

|  |  |
| --- | --- |
| **Function Definition:** | **float\* getStats (int size, int A[], int pass){**  //variable declaration and dynamic memory allocation  **int x, y, temp;**  **float sum = 0;**  **float\* retval = (float\*) malloc (4 \* sizeof(float));**    //bubble sort with element addition  **for (x = 0; x < size-1; x++) {**  **for (y = 0; y < size–x-1; y++) {**  **if (A[y] > A[y+1]) {**  **temp = A[y];**  **A[y] = A[y+1];**  **A[y+1] = temp;**  **}**  **}**  **sum += A[size-x-1];**  **}**  **sum += A[0];**  //mean and median  **retval[0] = sum/size;**  **if(size % 2 != 0) {**  **retval[1] = A[size/2];**  **} else {**  **retval[1] = (float) ( A[size/2] + A[size/2 - 1] ) / 2;**  **}**  //variance computation and index with passing score linear search  **sum = 0;**  **for (x = 0; x < size && A[x] < pass; x++) {**  **sum += ( A[x] – retval[0] ) \* ( A[x] – retval[0] );**  **}**  **for (y = x; y < size; y++) {**  **sum += ( A[y] – retval[0] ) \* ( A[y] – retval[0] );**  **}**  //variance and passing rate  **retval[2] = sum/size;**  **retval[3] = (float) 100 \* ( size – x ) / size;**  //return array  **return retval;**  **}** |

|  |  |
| --- | --- |
| **Updated Execution Stack:** | |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | size |  | A |  | pass | | **6** |  | **A100** |  | **45** | | D100 |  | D200 |  | D300 | |  |  |  |  |  | | x |  | y |  | temp | | **~~0 1 2 3 4~~**  **~~0 1~~ 2** |  | **~~0 1 2 3 4 0 1 2 3~~**  **~~0 1 2 0 1 0~~**  **~~2 3 4~~ 5** |  | **~~51 60 49 51~~ 49** | | D400 |  | D500 |  | D600 | |  |  |  |  |  | | sum | | |  | retval | | **~~0.0 60.0 111.0 160.0 206.0 249.0 285.0~~**  **~~0.0 132.25 20.25 2.25 2.25 12.25~~ 156.25** | | |  | **F100** | | D700 | | |  | D800 | | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | numStud |  | scores |  |  |  | | **6** |  | **A100** | 0 | **~~51 36 36~~ 36** | A100-A103 | | B200 |  | B100 | 1 | **~~36 49 43~~ 43** | A104-A107 | |  |  |  | 2 | **~~49 43 49~~ 46** | A108-A10B | | passing |  | result | 3 | **~~43 51 46~~ 49** | A10C-A10F | | **45** |  | **F100** | 4 | **~~60 46 51~~ 51** | A110-A113 | | B300 |  | B400 | 5 | **~~46 60 60~~ 60** | A114-A117 | | | **A.R. of getStats()**  **Heap Memory**   |  |  |  | | --- | --- | --- | | 0 | **47.500** | F100-F103 | | 1 | **47.500** | F104-F107 | | 2 | **54.250** | F108-F10B | | 3 | **66.667** | F10C-F10F |   **A.R.**  **of main()** |