Department of Computer Science and Engineering

Assignment 4

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

Answer all the questions. **[20 + 20 + 30 + 30 = 100 Marks]**

1. Write a program to input n integers from the keyboard and store them within an array. Declare another array of size n and copy the contents of the first array into the second array as per the following rules:

   (a) For the elements in the even positions of the first array, into the even positions of the second array in the reverse order. Example: the element in the 1st even position of the 1st array should be placed in the last even position of the 2nd array. Similarly the element in the last even position of the 1st array should be placed in the 1st even position of the 2nd array. For more details see the example given below.

   (b) For all elements in odd positions of the first array, copy the values in the corresponding odd positions of the second array after multiplying that value with the rounded off value of the square root of the sum of squares of all elements to the right of that element in the first array. Round off the square root value to a perfect integer (lower integer in case of decimal part being less than 0.5 and next higher integer in case the decimal part is greater than or equal to 0.5). In case no numbers exist to the right hand side for an odd position number, copy the number as it is in the corresponding odd position of the second array.

   You can input the original array as you wish. Read the question very carefully along with the example shown and understand the logic. For output display the second array only.

   Example:
   (Please take the input and display the output as shown)

   Input:

2 6 7 4 9 3

Output:

28 3 70 4 27 6

[Reason: The even position numbers are copied in reverse order in even positions only. For the odd position numbers, say the number at position 1 i.e '2' of the input array, sum of squares of all elements to the right of '2' in the original array = 191. Square root of 191 is 13.82, so the rounded value is 14, now multiply the original number '2' with 14 to get 28, which is stored in the 1st position of the second array. Apply the same rule for all odd position elements. In case there are no elements to the right of the input array, then copy the value as it is in the second array.]

2. Write a C program to calculate the relative grades of 10 students. The steps to calculate the grade are given as follows:

   (a) Take the marks of the 10 students in an array marks[]. The maximum and minimum marks are 100 and 0 respectively. If the entered marks are outside the range, then error should be displayed, and the system will wait till you enter the marks within the range.

   (b) Calculate the average (μ) of the marks

   (c) Calculate the standard (σ) deviation of the marks, as per the following formula

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

   (d) Hence calculate the relative grades of the marks using the category: 10 if the mark is above (μ + 1.5*σ)
   9 if the mark is above (μ + 1.0*σ)
   8 if the mark is above (μ + 0.5*σ)
   7 if the mark is above (μ)
   6 if the mark is above (μ - 0.5*σ)
   5 if the mark is above (μ - 1.0*σ)
   4 if the mark is above (μ - 1.5*σ)
   Your input should be the marks of 10 students within an array and your output should be the grades of each of the students based on the marks that you have inputted. You may print the grade of each student in a new line and additionally make the output more readable, for example,
   Grade of Student 1 = 9
   Grade of Student 2 = 10
   Grade of Student 3 = 7 and likewise.....

3. Write a C program to perform the following game. You have N number of buckets and each bucket contains some specific number of balls. Take input the value of 'N' from the keyboard and the number of balls in each of the N buckets and store it in an array ball[] of size N. Perform successive iterations whereby at each iteration, you have to find the number of bucket(s) with

the least number of balls and also the least value (say the value be x), and remove 'x' balls from each bucket. You have to continue the game till no balls in any bucket are left. Display the number of bucket(s) with the least number of balls as well as the number of balls remaining in each bucket after you have extracted the least value, i.e x from each bucket. For displaying the number of balls remaining after each iteration, you do not need to resize the ball[] array or declare any other array, simply print the non-zero values within the array side by side after each iteration. You can use only one array to perform all the operations. Consider the example below.

Example:

(Please take the input and display the output as shown)

Input:

Enter Number of Buckets: 5
Enter Positive Number of Balls for Each 5 Buckets: 7 5 2 7 2 [this will be stored in the ball[] array]

Output:

(You would need to print the iteration number as well as shown)

———Iteration-1———

Number of buckets with least number of balls = 2

Numbers of balls in the Remaining buckets = 5 3 5

[Reason: Buckets 3 and 5 have the least number of balls, i.e 2 balls which is the value of x. Now remove x=2 balls from each bucket which provides the balls remaining in the other buckets. Since bucket 3 and 5 have 0 balls remaining you do not print it]

———Iteration-2———

Number of buckets with least number of balls = 1

Numbers of balls in the Remaining buckets = 2 2

[Reason: Continue from iteration 1. Bucket 2 has the least number of balls, i.e 3 balls which is the value of x. Now remove x=3 balls from each bucket which provides the balls remaining in the other buckets. Since bucket 2 have 0 balls remaining and buckets 3 and 5 already had 0 balls from the previous iteration, you do not print it]

———Iteration-3———

Number of buckets with least number of balls = 2

Numbers of balls in the Remaining buckets = NIL [END]

4. You might have observed queues in railway reservation counters. In a queue when the first person leaves the queue, the second person becomes the first member of the new queue. Write a C program to input n characters into a character array of size n. Consider the original queue to be this character array of size n. In each iteration, compute the number of vowels that are present within the current iteration's queue (say this number be 'k') and print all the elements of the queue by skipping the first k elements of the queue. This queue that you have printed becomes your new queue for the next iteration and continue this process until there are no more elements in the queue or there are no vowels remaining in the current queue being considered. For printing the new queue you do not need to perform any deletion or array resizing operation. Rather always keep track of the beginning index of the latest queue for each iteration and advance the index accordingly at the end of iteration. You may use only one array to implement the queue.

   Example:

   (Please take the input and display the output as shown)

Input:

Original queue: a, y, u, g, h, i, e, k, l

[NOTE: While taking input the character array one by one use scanf("<space> %c",&ar[i]) within the loop, where i is the index and ar is the character array. GIVE ONE SPACE BETWEEN THE opening quote and the '%c' within the scanf()]

Output:

(You would need to print the iteration number as well as shown)

——-Iteration-1———

New queue: h, i, e, k, l

[Reason: In the original queue there are 4 vowels, a, u, i and e. Hence k=4. Thus display the queue from after the first 4 elements of the original queue and the displayed queue becomes your new queue for the next iteration]

——-Iteration-2———

New queue: e, k, l

[Reason: Same as above, now considering the new queue from the previous iteration]

——-Iteration-3———

New queue: k, l

[END here, no more vowels]
