1806ICT Programming Fundamentals

Workshop Week 5: Arrays

<u>Please complete these exercises before your workshop.</u>

1. Write a program that reads in the size of an array and the array elements (all of which are integers). The program then computes the sum of all the elements in the given array. Note: If the input is 3 1 4 8, then the array has 3 elements {1, 4, 8}.

   Sample Run:

Input	Output
3 1 4 8	13
5 2 4 6 8 13	33

2. Write a program that reads in the size of an array and the array elements (all of which are integers). The program then prints out the minimum and maximum values in the given array.

   Sample Run:

Input	Output
3 4 8 5	Min = 4, Max = 8
5 14 2 5 13 9	Min = 2, Max = 14

3. Write a program that reads in the size of an array and the array elements (all of which are integers). The program then prints out an array that contains the exact same numbers as the original given array, but with the array elements rearranged such that all the even numbers come before all the odd numbers. Other than the latter requirement, the numbers can be in any order. Challenge: Instead of using a new array to store the rearranged numbers, try to rearrange the numbers in the original given array itself.

   Sample Run:

Input	Output
4 3 4 5 6	6 4 5 3
5 14 2 5 13 9	14 2 5 13 9

4. Write a program that reads in 20 integer numbers, each of which is between 1 and 10, inclusive. Use an array to store the numbers as they are being read in, if and only if that number is not a duplicate of a number already read. Print out the array containing non-duplicate numbers.

   Sample Run:

Input	Output
1 2 3 4 5 6 7 8 9 10 9 8 7 6 5 4 3 2 1 2	1 2 3 4 5 6 7 8 9 10
1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2	1 2

5. Write a program that reads in the size of an array and the array elements (all of which are integers). Your program must rearrange the numbers in the array such that every 3 is followed by a 4. Do not move the 3's, but every other number may move. The given array contains the same number of 3's and 4's. Every 3 has a number after it that is not a 3 or 4, and a 3 appears in the array before any 4.

   Sample Run:

Input	Output
4 1 3 1 4	1 3 4 1
4 3 2 2 4	3 4 2 2
7 1 3 1 4 4 3 1	1 3 4 1 1 3 4

6. Write a program that reads in the size of an array and the array elements (all of which are integers). Say that a "clump" in the given array is a series of 2 or more adjacent elements of the same value. Your program will print out the number of clumps in the given array.

   Sample Run:

Input	Output
6 1 2 2 3 4 4	2
5 1 1 2 1 1	2
7 1 1 1 1 1 1 1	1

7. Write a program that reads in the size of an array and the array elements (all of which are integers). Print a version of the given array where each zero value in the array is replaced by the largest odd value to the right of the zero in the array. If there is no odd value to the right of the zero, leave the zero as a zero.

   Sample Run:

Input	Output
4 0 5 0 3	5 5 3 3
4 0 4 0 3	3 4 3 3
6 7 0 4 3 0 2	7 3 4 3 0 2

8. Write a program that reads in the size of an array and the array elements (all of which are integers). Find the number of indexes in the array such that the sum of elements at lower indexes is equal to the sum of elements at higher indexes. For example given an array {-7, 1, 5, 2, -4, 3, 0}, one of such indexes is index 3 because

   -7 + 1 + 5 = -4 + 3 + 0

   Given the same array, another of such indexes is index 6 because

   -7 + 1 + 5 + 2 – 4 + 3 = 0

   Therefore, the number of such indexes in the given array would be 2.

9. Write a program that performs the addition of two matrices of the same size. An example showing the addition of two matrices is given below:

$$\begin{pmatrix} 3 & 6 & 1 \\ 2 & 1 & 4 \\ 5 & 2 & 3 \end{pmatrix} + \begin{pmatrix} 4 & 1 & 3 \\ 2 & 5 & 5 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 7 & 7 & 4 \\ 4 & 6 & 9 \\ 6 & 4 & 6 \end{pmatrix}$$

   Use two 2-D arrays to store the elements for the two matrices.

10. Suppose that you are going to play 18 holes of golf with two friends. Write a program to randomly generate the input scores for each hole (for you and both your friends), and then at the end of the round (i.e. after hole number 18), add up the scores and report on various statistics for each of you. Use the rand( ) function to randomly generate the scores for each hole (each score should be between 1 and 10, inclusive)

    Your program should use arrays to store 18 scores for an 18 holes round. One 1-D array should store the so-called 'par' score for each hole, where par is the number of shots a player is expected to take for a given hole. You can use the rand( ) function to randomly generate the par score for each of the 18 holes (par is either 3 or 4 or 5 shots). You should also use a 3-by-18 2-D array, to keep your own score and the scores for each of your two friends.

    For each player, the program should report on the following: the total score for the round, the average number of shots per hole, and how many of each of the following were achieved:

- Hole in 1
- Par
- Birdie (one under par)
- Eagle (two under par)
- Albatross (three under par)
- Bogey (one over par)
- Double bogey (two over par)
- Triple bogey (three over par)

Your program should also report the average score for each of the 18 holes. For example, for hole number 10, if you took 4 shots, one friend 6 shots and another 8 shots, the average for that hole would be 6.

Before you start coding, do a top-down design for the program. You should use at least two functions (one for inputting the scores, and one for printing the post-round report).

You should also take into consideration that your program should be easy to test. In particular, you might find it a hassle that there are so many holes! Use a #define to specify the number of holes and test your program initially with a three holes course. Only run tests over all 18 holes when you are convinced that your program works independently of the number of holes defined.