The State University of New York, Korea
Computer Science

CSE 216  Handout 6: PS 5  October 19, 2021

This problem set is due at **11:55pm on Tuesday, October 26, 2021**. Don’t go by the due date that you see on Blackboard because it is in ET. Go by the one given in this handout.

**Be sure to include** a comment at the top of each file submitted that gives your name and email address.

Submit on Blackboard your solution files with the names specified in each problem. Multiple submissions are allowed before the due date.

**Note:** Please read the ”Cooperation vs. Cheating” (i.e., ”Academic Integrity”) section of the syllabus before you start this problem set and future problem sets.

**Naming things in C/C++**

If you have a multi-word name in C or C++, you may use one of the following two very commonly used naming conventions: `randomIntArray` or `random_int_array`. I used the former. If you prefer the latter, you may change the names to be in the second format and stay with that format.

**Problem 1 (60 points)**

Write a C program in a file named `arr.c` that solves the following problems and hand it in. You will have the needed functions and a `main` that tests the functions that solve the given problems.

1. (5 pts) Define a function, named `randomIntArray` with three formal parameters: an array to be filled with random numbers, the size of the array, and the upper limit of the random numbers to be created. If the third number is 100, then it would mean that the random numbers will be in the range of 0 and 99 inclusive.

2. (2 pts) In your `main` call `randomIntArray` with appropriate actual arguments to establish an array of random numbers of a certain size.

3. (2 pts) Define a function, named `print` to print the elements in an integer array of a certain size to the standard output device.

4. (2 pts) In your `main` call `print` to print the array established above.

5. (4 pts) Define a function, named `arrSum` with appropriate parameters to sum the elements in an integer array.

6. (2 pts) In your `main` print the average of the numbers in the array that you obtained earlier above. To compute the average, you must use `arrSum` that you defined earlier. Print the average to the standard output device. When you generate output to the screen, add some annotation so that the output will be meaningful.

7. (5 pts) Define a function named `contains` with appropriate parameters to search an element in an array of integers. It will return a value to be used as a boolean value.
8. (2 pts) In your main call `contains` with the array obtained earlier above to see if the array contains the integer 22 in it, and print the result in a meaningful format to the standard output device.

9. (5 pts) Define a function named `contains2` with appropriate parameters to search an element in an array of integers. The function returns the index of the array where the first occurrence of the number to be found is found. If the number is not contained in the array argument, it returns −1.

10. (2 pts) In your main call `contains2` with the array obtained earlier above to see if the array contains the integer 22 in it, and print the result (index found or −1) in a meaningful format to the standard output device.

11. (3 pts) Define a function named `countMultiplesOf` with at least two parameters: one an array of integers and the other an integer. This method returns the count of the integers in the array that are multiples of the second parameter. For example, 8 is a multiple of 2, but not a multiple of 3, and zero is a multiple of any number. I suggest that you also define and use an auxiliary function that tests if a number is a multiple of another and returns a value to be used as boolean.

12. (2 pts) In your main call `countMultiplesOf` with the array obtained earlier above and 7 as the second argument, and print the result to the standard output device.

13. (15 pts) Define a function named `buildHistogram` with at least one parameter: an array of integers, say `scores`. This function builds a histogram (frequency table). The histogram will also be an array itself, say it is named `hist`. This is how you build `hist`: First, find the largest element in `scores`, add one to it, and use the result as the size of `hist`. Now that `hist` is created we need to fill the array with some elements as follows: If `scores` contains 5 occurrences of 6, then 5 will be stored as the value in the array index 6 of `hist`. That is, `hist[6]` will hold 5. If you do that for each index of `hist`, then you have finished filling the array `hist`. Feel free to add any auxiliary functions.

14. (2 pts) In your main call `buildHistogram` with the array obtained earlier above. You might have the histogram as a local array in `main` if you structured your program that way. If you used a different structure, that is certainly fine as well.

15. (7 pts) Define a function named `printHistogram` with at least one parameter: array of integers. Call this function from the `main` with the histogram obtained in one of the previous steps and have this function print the histogram in the following format:

```
i: <n>: <n>’s for the index i of the array. n is the element of the histogram array in the index location i>
```

So, if the array holds the following values assuming the size of the histogram array is 8:

```
+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+
| 5 | 1 | 4 | 6 | 0 | 2 | 10 | 6 |
+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+---------------------------+
```

Then, the histogram printed should look like this:

```
0: 5: *****
1: 1: *
2: 4: *****
3: 6: *****
4: 0:
5: 2: **
6: 10: ********
7: 6: *****
```
Problem 2 (60 points)

In your solution of Problem 1 above you used the usual array notations, namely[]. In this problem you will rewrite your solution for Problem 1 using the pointer notation, namely *, only. That is, you are not allowed to use the array notation ([]) in this problem. Name your solution file for this problem arrptr.c and hand it in.