

CPSC 601: Programming with C & Java – Spring 2009
Homework 4

February 24, 2009

Due date and time: 3 p.m. Tuesday March 3, 2009

This set of homework consists of three Exercise problems from the textbook Chapter 9. Your program should have a head comment including problem number, a short description of the problem, and your name, and an appropriate amount of comments throughout the program. Also, your program should have a well thought-out logic and the program should be neatly coded.

- 9.3 [20 points]
- A modified version of 9.4 [20 points]: Write a function that computes two sums from the elements of a *two dimensional* array of integers. Thus, your `sum` function should have a two dimensional array of integers as its first parameter. The rest is the same as in the text.
- A refined version of 9.7 [40 points]: You will write two versions of bubble sort in one program – `bubble1()` function that is an original version of bubble sort (cf. page 313 Section 9.6 in the text) and `bubble2()` function that is a modified version as described in the problem statement.

To measure the time and compare the two versions of bubble functions, use a counter to count the number of comparisons. At the end of each function, the function returns the number of comparisons.

Use symbolic constants (using `#define`) such that:

```
#define IN_ARR_SIZE 100 /* for the size of the arrays a1 and a2 */
#define NUM_RUN    10  /* for the number of iterations in main */
                        /* and the size of the 2nd dimension of */
                        /* cnt_arr array (see below)           */
#define NUM_FUN    2   /* for bubble1 and bubble2             */
```

and in the `main`, you should have the following array declarations:

```
int a1[IN_ARR_SIZE], a2[IN_ARR_SIZE]; /* arrays to be sorted      */
int cnt_arr[NUM_FUN][NUM_RUN]; /* to keep the number of comparisons */
                                /* after each call of each function */
```

The `main()` function body should execute `NUM_RUN` iterations of the following:

1. Generate `IN_ARR_SIZE` random numbers, storing them in the two arrays `a1` and `a2`. The contents of `a1` and `a2` must be the same.
2. Print out the content of one of the two arrays.
3. Call the two bubble functions with each array.
4. Keep the returned count from each function in `cnt_arr` so that, for example, `cnt_arr[0][2]` has the count value returned by `bubble1` function after 3rd call.
5. Now the contents of the two arrays `a1` and `a2` must be the same, sorted. Print out the content of one of the two arrays.

When all the iterations are completed and the `cnt_arr` array is all filled in, compute the average and find the minimum and the maximum of the values in each row of `cnt_arr`, and print out the two averages, best cases and worst cases to compare the running time of the two bubble functions.