**Homework 1**

**Due date:** Thursday, January 24th, 2013 by 9:30pm.

**Ex. 1. a.** Inside your home directory in your Linux account (the default directory when you log on), create a folder for this class with the command  **mkdir c243**  Change your working directory to this folder using the command  **cd c243**  Repeat this step to create a directory called **ass1** and change your working directory to it.

**b.** Download the following files into the folder you have created for this homework: **[Makefile](file:///Users/mscheess/mike/postsabteaching/c243s13/C243_Dana/Homework/p1/Makefile)  [my\_array.h](file:///Users/mscheess/mike/postsabteaching/c243s13/C243_Dana/Homework/p1/my_array.h)**[**my\_array.cc**](file:///Users/mscheess/mike/postsabteaching/c243s13/C243_Dana/Homework/p1/my_array.cc)[**main.cc**](file:///Users/mscheess/mike/postsabteaching/c243s13/C243_Dana/Homework/p1/main.cc)

These files are available on Oncourse, in Assignments, Ass1.

**c.** Compile the program with the command  **make**.  Test the program with the command **array**.

**Ex. 2. a.** Add some functions to the class **My\_array** to do the following operations and with the given prototypes:

* Input the elements of the array from the console. This function asks for the size of the array first, and call an appropriate function to initialize the array with the given size (you will have to figure out which class method you need to call). Then it must ask the user to input all the elements of the array one by one and read them from the console.
  + 1. **void input();**
* Initialize the elements of the array with random values between 0 and a given maximum limit provided as a parameter. The size of the array will be provided as a second parameter. The function must reinitialize the array like before, except for the case where the size is given as 0. In that case we'll assume that the user does not want to reinitialize the array. To make things simpler, a default value of 0 will be given for this parameter.

**void randomize(int limit=100, int the\_size=0);**

* Compare the elements of the array with another one and decide if they are equal or not (2 functions). One of the operators must be explicitly implemented with a for loop comparing the elements of the arrays one by one. The other operator should simply return the opposite of the first one by calling it (one line of code).

**bool operator==(My\_array &data);**

**bool operator!=(My\_array &data);**

**Note.** If the arrays don't have the same size, they are not equal no matter what the elements are. This is a simple test since we know the size of the arrays explicitly and it should be done before the for loop.

* Compute the Hamming distance between two arrays, by counting the number of elements in the two arrays that are different. For example, the distance between the following arrays

**2 1 6 3 4**

**5 1 6 8 4**

is 2 because the first and fourth elements of the two arrays are different. If the two arrays are equal, the distance between the should be 0. Prototype:

**int hamming(My\_array &data);**

**Note.** If the arrays are not of the same size, in the Hamming distance you should compute the distance as described for the parts of the arrays that are common (on the length of the smaller of the two arrays), and then add to it the difference between the sizes of the arrays. For example, for the following arrays:

**2 1 6 3 4 5 9 0**

**5 1 6 8 4**

the distance would be computed by comparing the first 5 elements of the two arrays, which gives us 2 as before, and then adding the difference between the sizes of the arrays, which is 8-5=3. So the total distance is 2+3=5. This function should only return the value and not output it.

**b.** Modify **main** to test the three functions and at least one of the operators. For this, in **main** do the following:

* declare three array objects in **main**;
* call the function that inputs the array from the user to initialize one of the three arrays;
* randomize the second and third arrays with a limit of 10 and the size entered by the user at the previous step by calling the appropriate class method;
* compare the first and second arrays using one of the operators you defined and output an appropriate message;
* compute and output the Hamming distance between the second and third arrays and output the result.

Remove or comment out any part of the original main that is not relevant to your program.

**Submit** to Oncourse Dropbox all the files (.cc and .h) that you've modified, probably all of them.

Turn in hardcopies of all modified files.

Both must be done by 9:30pm on the due date.