Dana Vrajitoru
C243 Data Structures

Homework 1

Due date: Thursday, September 3, 2015.

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
if you have not done this already. Change your working directory to this folder using the command
cd c243
Repeat this steps to create a directory called hw1 or whatever name you want for it and change your working directory to it.

b. Download the following files into the folder you have created for this homework:
Makefile
MyArray.h
MyArray.cc
main.cc

If you work from a terminal in your Linux account, you can copy them directly doing something like
cp /home/danav/public_html/teach/c243/p1/Makefile ./
and then a similar command for each of the files.

From a Linux system other than our labs, you can also copy them with a command like
wget http://www.cs.iusb.edu/~danav/teach/c243/p1/Makefile

These files are also available on Canvas, in Files, Homework 1.

c. Compile the program with the command
make
Test the program with the command
array
or if it doesn't work,
./array

Note that when you run the program, you should see an error message about illegal access to an element of the array. This is done on purpose to test the feature.

Ex. 2. a. Add some functions to the class MyArray 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.

```
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 theSize=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==(MyArray &data);


bool operator!=(MyArray &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(MyArray &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 the main to test the three functions and at least one of the operators. For this, in the main do the following:

- declare three array objects in the main;
- call the function that inputs the array form 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 Canvas, under Assignments, Homework 1: all the source files (.cc and .h). If you modify the Makefile, you have to submit it too.