CIS 130 - Intro to Programming - Spring 2007
Homework Assignment #10 - **INDIVIDUAL** assignment

Homework #10 DUE**: BEGINNING** of class, Wednesday, May 2, 2007

Purpose: get practice with arrays, for-loops, and file input/output.

Note that use of the design recipe is still required for all functions, including C++ functions! But, use **C++ types** in **contracts** for C++ functions, and use == for examples for non-void C++ functions. (describe output for a specific example for void C++ functions)

1. Consider **line_of_X** from HW #9. You could use this to create a kind of horizontal bar chart, calling it for each of a set of values. And what is an array but a set of values?

Write a C++ function **bar_chart** that expects two arguments, an array of integers and its size. It prints to the screen a horizontal bar chart with the help of **line_of_X**, printing a line of X's the length of each array value. Your solution is expected to use appropriately use a **for-loop** and is expected to call **line_of_X**.

For example, if you have **const int NUM_MSRS = 7;** and **int measures[NUM_MSRS] = {3, 1, 6, 2, 8, 4, 5};**, then **bar_chart(measures, NUM_MSRS)** would cause the following to be printed to the screen:

```
XXX
X
XXXXXX
XX
XXXXXXXX
XXXX
XXXXX
```

2. You, of course, need to test **bar_chart**. So, write a **main** function whose purpose is to test **bar_chart** --- it should be in a file named **test_bar.cpp**, and should call the function **bar_chart** in **bar_chart.cpp/bar_chart.h**.

It should call **bar_chart** at least 3 times, for arrays of at least 3 different sizes. When all is working fine, **bar_chart.cpp**, **bar_chart.h,** and **test_bar.cpp** will be ready for submission.

3. Now, you should have gotten a little array-declaration practice in **test_bar.cpp**. But, what if the user would like to print a simple bar chart based on data from a file?

file_bar should take a file name, and how many values should be read from that file, as its arguments. It should then take the steps necessary to read values from that file and use **bar_chart** to print a horizontal bar chart based on those values. (What will you have to construct to be able to call **bar_chart**?) Use of a for-loop is required, as is use of **bar_chart**. Don't forget to close the file when you are done!

For example, if **stuff.txt** contains:

```
3
1
5
2
```

...then calling **file_bar("stuff.txt", 4)** should cause the following to be written to the screen:

```
XXX
```

```
X
XXXXX
XX
```

4. Write a **main** function whose purpose is to test **file_bar** --- it should be in a file named **test_file_bar.cpp**, and should call the function **file_bar** in **file_bar.cpp/file_bar.h**.

   It should call **file_bar** at least 2 times, for files of at least 2 different sizes. When all is working fine, **file_bar.cpp**, **file_bar.h, test_file_bar.cpp**, and whatever input files you created for testing purposes will be ready for submission. (Start these input files' names with **hw10** and end them with the suffix **.txt** so that the **get_hw10** tool can find them... 8-) For example, **hw10stuff.txt**, **hw10nonsense.txt**)

5. Oh - now you find out your users want a function that will help them to interactively create the kind of file that **file_bar** expects. **put_ints** will take a file name and a number of desired values as its argument -- it will then ask the user for that many integers, writing each to the specified output file. You are expected to appropriately use a for-loop in your function.

6. Write a main function whose purpose is to test **put_ints** --- it should be in a file named **test_put_ints**, and should call the function **put_ints** in **put_ints.cpp/put_ints.h**.

   It should call **put_ints** at least 2 times, for at least two different files and two different quantities of integers. When all is working fine, **put_ints.cpp**, **put_ints.h,** and **test_put_ints.cpp**, will be ready for submission.

7. Now write a main function in a file named **save_and_show.cpp** that asks the user how many integers they have to deal with, and where they should be saved. It then uses **put_ints** to obtain those values and write them there. Then it calls **file_bar** to display to the screen a bar chart based on the now-saved data.

When you are happy with these functions, you can either submit the .cpp, .h, and .txt files mentioned above, OR you can use the following quickie-tool to build a file containing all of them (a tar file) and submit that one file instead (IF you have named your functions PRECISELY as given above...):

...if you are interesting in the quickie tool, then type the following at the cs-server prompt:
~st10/get_hw10

...give the name of a directory you want built, and when done, if all 12+ files are listed on-screen as being in your new file, then you can submit the file whose name it tells you at the end.

(Note: you STILL use ~st10/130submit to submit this homework! But it is your choice if you submit the 12+ .cpp and .h files in the usual way, OR use ~st10/get_hw10 and submit the single file it builds containing (hopefully) your 12+ .cpp, .h, and . txt files.)