Lab Assignment #1

Goal: To setup all necessary programming tools (compilers, tools for accessing different platforms as needed, etc.) and to become comfortable using the C/C++ syntax in writing relatively simple programming code.

Due Dates:

Part I: Email, confirming your successful completion of tasks (a) and (b) of the Introduction below – **by Mon, Jan 09 @ 2pm**

Part II: Full project write-up (see below) + code submission (electronic) – **by Mon, Jan 16 @ 10pm**

Submission: A turn-in link will be made available on the course web site shortly, under a folder called 'Assignment Submission'.

Introduction

Your primary task in this assignment comprises the following three parts (due at different times, see below):
(a) to install an IDE for C++ (e.g., Visual Studio 2010 Professional) on a computer you intend to use regularly for project work this quarter, and successfully compile and run the (simplest) "Hello, World" program.
(b) to install tools for terminal access to Linux, and for uploading your files, and then successfully compiling and running simple C++ files (e.g., "Hello, World" code, again).
(c) to use the tools installed and configured in parts (a) and (b) for solving a relatively simple programming task, described further below.
The specific deliverables are identified in the 'Deliverables' section below.

Mechanics

You will be working individually for the coding and project description portion of this project assignment. You are, however, free to use the help of others, if you need it, in order to successfully install and configure the tools you will need to use. These tools will be needed for every programming assignment this quarter, so the effort now will pay off multiple times.

Product Requirements

Write a program that evaluates the strength of a hand for the card game "bridge".
The input for a single bridge hand will be a space-delimited sequence (provided on the command line, as follows: *name_of_your_executable sequence_of_cards_in_a_hand*) that represents a set of cards, with each card itself represented by a pair of characters: one character denoting the card's rank (e.g., 'A' for ace, 'K' for king, 'Q' for queen, 'J' for jack, 'T' for 10, '9' for 9, '8' for 8, etc.), and the other character denoting its suit (e.g., 'S' for spades, 'H' for hearts, 'D' for diamonds, and 'C' for clubs). Hence, a sample sequence might look like this:

AH KC 9C 3C 4H TD QS JS QH TC 7C 2S 8C

(representing respectively ace of hearts, king of spades, 9 of clubs, 3 of clubs, 4 of hearts, 10 of diamonds, etc.).

To evaluate a hand, use the following standard bridge values:
An ace of any suit is worth 4 points, a king is 3 points, a queen is 2 points, and a jack is 1 point.
A suit that has no cards (a.k.a., a void) is worth 3 points; a suit that has a single card of it (a.k.a., a singleton) is worth 2 points; a suit that has exactly two cards of it (a.k.a., a doubleton) is worth 1 point. Finally, all suits longer than 5 cards carry 1 point for each card after the 5^{th} one.
For example, the sample hand displayed above (consisting of 13 cards total) is worth a total of 15 points: 1 ace (4 pts.), 1 king (3 pts.), 2 queens (2*2 pts.), 1 jack (1 pt.), plus a single diamond (2 pts.) and 6 clubs (1 pt.).

Display the results for each hand in a layout sorted by suit, and by rank in descending order within each suit. In the example above, the output should look like this:

SPADES:	Q J 2
HEARTS:	A Q 4
DIAMONDS:	T
CLUBS:	K T 9 8 7 3
Points = 15	

Make your code robust, with as few assumptions about the input as possible, and as easy to change to accommodate new functionality as possible. Examples of types of potential additional functionality (*none of which you need to implement* for this assignment) include: correct handling of additional error conditions; evaluating the collective (two) hands of both bridge partners (as described in the Extra Credit part below); adding functionality for making bids based on the total number of points in both partners, according to a known bidding system; etc.
For instance, do not assume that there are exactly 13 cards in each hand; do not assume that all character pairs represent a valid card; do not assume that identical cards cannot show up in a given random hand; do not assume that the cards are in sorted order, either by rank or by suit. Some of those are erroneous conditions that need to be handled appropriately; others are simply alternative types of input that do not represent an error.

Extra credit: Evaluate the collective hands of both partners in a game of bridge. The challenge here is that if two hands (one for each partner) are passed in as input to the program, the sum of the points of both hands (as evaluated by the procedure above) may not represent correctly the strength of the collective hand. For example, if one hand has a singleton king in a suit, and the other hand does not have the ace in that suit, that king may not be worth as much, since it may easily be captured by the opponents (with the ace one of them holds), and the short suit (singleton) of the king is not worth its additional points either, since it represents a weakness in this case (the king is more vulnerable), not a strength to be extra valued. There can be many such interesting combinations, involving two bridge hands – changing the perception of what actual hand strength means.
Your task for this extra credit, if you should choose to work on it, is to describe a few such situations of your choice, and develop code to accommodate them.
Note: The complexity of the extra credit part goes beyond the simple programming and design task, so only undertake extra credit tasks once you are finished with the main task of the assignment.

Tips:

- You do not need to be a bridge player (or even know the rules of the game) in order to be able to do this project. Do not let possible unfamiliarity with the game prevent you from making progress; it's quite common for programmers to know nothing about accounting or chemical compositions or building architectures or a number of other domains for which programming code is being written anyway.
- Start by understanding the domain well. Take out a deck of cards (or sketch some) and calculate a few sample hands of 13 cards each, according to the simple algorithm outlined above.
- **Ask questions**; I can answer; talk to your fellow classmates about what questions they have. Certain parts of the product description may be vague at present (which is quite common in initial product descriptions), and in need of clarification before you have a firm grasp of what it is exactly that needs to be built.
- Consider what classes you may need to have in your program in order to model the necessary data. What interfaces (i.e., public methods) would each class have to expose to its clients, and what methods might be better left private?
- We will discuss these and many other issues in our next class meeting. The more you have thought about the problem, the more value you will get out of such a discussion, and the more valuable I can be in answering questions and guiding you toward effective designs, etc.

Evaluation
In the evaluation of your project work, I will be looking to see not only that your code compiles and runs correctly with my test cases, but also that you have an associated write-up, in which you have:
(a) described your designs clearly, including classes and their interfaces (public methods), and justified those designs over alternatives you have considered;

(b) created a reasonable (even if incomplete) set of test cases to test your program for functional correctness as well as for robustness against various types of inputs;
(c) described clearly and concisely what you have been able to accomplish of the required tasks, and what not;
(d) listed the bugs you are aware of in your code (note: every code has bugs, and that's okay so long as they are not critical, preventing the main functionality from working);
(e) described what, if any, additional functionality you were able to build (based on the extra credit task) and which two-hand combinations it is able to handle well;
(f) a snapshot of your program running on a sample input, and producing its output;
(g) listed the number of hours it took you to work on this project, broken down by hours before coding (i.e., designs, clarifying requirements, etc.) and hours for coding and testing;
(h) organized and presented your write-up well.

Note that Extra credit should not be attempted unless you are reasonably certain you have completed (virtually) everything from the tasks in the main part of the assignment. Extra credit is nominally worth 10% over the points for the main parts.

Deliverables
For Part I, the deliverable is a single email to vrazmov@uwb.edu by the due date, in which you confirm that you were able to successfully install and use the necessary tools (C++ compiler; file transfer utility for Linux; terminal window utility for Linux) for running elementary programs like "Hello, World" without errors.

For Part II, I will expect to see the following deliverables by the due date:
(1) A write-up addressing all the questions from the Evaluation section above – this should be clear and concise; extra length is not a virtue. Use a common document format, to make sure that I have the software to open it and read your write-up.
(2) Your source code – including only your `*.cpp` and `*.h` files; no executables, no object files, no project files are necessary or desired.

Submission
There will be a submission link on the course web page shortly.
Please name your write-up files in a way that will be easy for me to identify them (e.g., *YourLastName*-css342-lab1-writeup.doc, where *YourLastName* is your last name).