CS 121 – Week 3 Worksheet

1. Assume you are working on a calendar application that works from year **1970** and above with a software team and created the following function for validating an inputted date. You've realized that the code looks rather hard to read and doesn't display errors, and would prefer to refactor it before showing the rest of your team. **In pseudo-code, code comments, or jotted notes, explain how you would rewrite the following code segment to improve its cleanliness and readability.**

This question was mostly open-ended and was to practice what you would do with rough drafts of code. There is no “totally” right answer, and I'd like to hear your guys' suggestions for code cleanliness.

My thoughts on this exercise were:

1. Change the returns to a boolean flag that checks for validity (which is returned in the end)
2. Add error cout/cerr statements per “flag = false”
3. Use an enum for the months (this is entirely subjective)
4. Make a boolean function for leap year and integer function for months (to get end of month)
5. Add checks for days in the final else statements (again, subjective)
6. Change “not” logic for year check
7. Change logic for days (to AND logic)
8. Change month/day checks to booleans as well.

An example of these changes are in the file “validate-week.cpp”

2. Write a function that, given a string, outputs (to console) whether or not it is a palindrome.

   Some notes before working on this:
   - A palindrome is a word or phrase that looks the same when written backwards. *Racecar*, *a but tuba*, and *b* are palindromes, but *palindrome* and *computer* are not.
   - Assume *using namespace std;* is noted before the definition of this function
   - A string's characters can be accessed by subscript (like an array). For instance, to access the second letter of a string called *str* would be *str[1]*.
   - You might want to format the string before checking if it is a palindrome.
   - You can make any number of extra functions to complete this problem.
   - The function is **only** outputting whether or not the word is a palindrome.

My (example) solution of this problem is in the file “palindrome-soln.cpp”.

If you use the “algorithm” C++ library the string scrubbing can be cleaner (namely use the “find” function). As this is out of the scope of the class I'd rather not talk about it, but if you search “how to erase characters in a string C++” on Google you can find some information about it.

3. Explain the difference between implicit and explicit type conversion, and give two example scenarios of when you would perform type conversion.

Taken from the words “implicit” and “explicit”, implicit type conversion is when data type conversion is implied, whereas the other is where the type conversion is written in manually.

Examples of implicit and explicit type conversion:

```
char ch = 'A';
short y = ch;       // implicit type conversion (uses the ASCII value of 'A', which is 65)
int x = 1.8;        // another implicit type conversion (float to int)

cout << char(x);  // type of explicit call
                  // other examples: char(x), (char)x, and static_cast<char>(x)
```

Example scenarios:

Calculator program (adding two numbers and one is an int whereas another a double)

Library system (have an array of book listings, each element is accessed by a char/ASCII)

- Example: “Everybody Poops” would be 'E' and index #5 (book_list['E' – 65])

More Info Here: http://www.cplusplus.com/doc/tutorial/typecasting/