Homework5 – 10 Points

Due Thursday, February First at class time

Project name: hw5
Source Files: hw5.cpp, mystring.cpp, mystring.h
Put the homework template header on hw5.cpp

You will write two functions and test cases using catch.hpp. The file setup is very similar to hw3.cpp.
You will need to add these three files to your solution in Visual Studio.

mystring.h

```
hw5                                                    (Global Scope)
1   #ifndef __MYSTRING__H__
2   #define __MYSTRING__H__
3
4   #define _CRT_SECURE_NO_WARNINGS
5   #include <cstring>
6   #include <cctype>
7
8   void extractNames( char str[], char first[], char middle[], char last[] );
9   void trim( char cstr[], char method);
10
11  #endif
```

mystring.cpp

```
hw5                                                    (Global Scope)
1   #include "mystring.h"
2   void trim( char cstr[], char method)
3   {
4       // your code goes here -- no input or output in function
5   }
6
7   void extractNames( char cstr[], char first[], char middle[], char last[] )
8   {
9       // your code goes here -- no input or output in function
10  }
11
```

hw5.cpp

```
hw5                                                    (Global Scope)
1   #include "mystring.h"
2   #include <cstring>
3   #define CATCH_CONFIG_MAIN
4   #include "../catch.hpp"
5
```

The first function will be a trim function:

```
Prototype:  void trim( char cstr[], char method)
```

Description: this function will remove whitespaces from the front, the end or both depending on that method is passed to the function. If an invalid method is passed, you will not modify the string parameter at all.

Method:

The method character can be upper or lower case.
If a 'b' is passed in for the method, you will trim all whitespace from the front and back of the cstring.
If a 'f' is passed in for the method, you will trim all whitespace from the front of the cstring only.
If a 'e' is passed in for the method, you will trim all whitespace from the end of the cstring only.

Cstring: can be any array of characters that has a null terminator including an empty string

Write many test cases.
2 Example test case:

```
TEST_CASE ( "trim: give a string 'no white spaces either side' method = 'B'"
            "      expect no changes to the string")
{
    char cstr[100] = "no white spaces either side";
    char answer[100] = "no white spaces either side";

    trim( cstr, 'B' );
    REQUIRE( strcmp( cstr, answer ) == 0 );
}

TEST_CASE ( "trim: give a string ' \t\nwhite spaces at front.' method = 'f' "
            "      expect 'white spaces at front.'")
{
    char cstr[100] = "  \t\n\fwhite spaces at front.";
    char answer[100] = "white spaces at front.";

    trim( cstr, 'f' );
    REQUIRE( strcmp( cstr, answer ) == 0 );
}
```

When writing your test case, make sure you pass in strings with different whitespace in the front, end, and both. Be sure and test that 'b' and 'B', 'f' and 'F', 'e' and 'E' work with the trim function.

The second function will be an extract Names function.

Prototype:
`void extractNames( char cstr[], char first[], char middle[], char last[] );`

Description: this function will extract a person's last name, first name and the middle name if it exists from a c style string. You must copy the data from the c string into the appropriate array first, middle or last. No section of the name will contain spaces. You are guaranteed that the name will only contain at least the first and last name. Middle might be present, and it might not.

Example Names:

"Last, First Middle"
"Last, First"
"First Middle Last"
"First Last"

You must use the strchr to determine if a comma is present within the person's full name.
You must use the strtok function to find each token within the string.
You must use the strcpy function to transfer data from the fullname c string to the first, middle and last c strings.

I will guarantee you that an empty string will not be passed.

Write many test cases:
2 Example test case for c strings

```
TEST_CASE( "extractNames: Several spaces arround names\n"
           "              given '  Firstname  Lastname  '\n"
           "              expect first=Firstname last=Lastname\n"
           "                     middle should be empty")
{
    char cstr[100] = "  Firstname  Lastname" ;
    char first[100], middle[100], last[100];
    char fAnswer[100] = "Firstname";
    char mAnswer[100] = "";
    char lAnswer[100] = "Lastname";

    extractNames( cstr, first, middle, last );

    CHECK(strcmp( first, fAnswer) == 0);
    CHECK(strcmp( middle, mAnswer) == 0);
    REQUIRE(strcmp( last, lAnswer) == 0);
}

TEST_CASE( "extractNames: Given: \"Schrader, Roger George\""
"              expect: first = 'Roger', middle = 'George', last = 'Schrader'")
{
    char cstr[100] = "Schrader, Roger George";
    char first[100], middle[100], last[100];
    char fAnswer[100] = "Roger";
    char mAnswer[100] = "George";
    char lAnswer[100] = "Schrader";

    extractNames( cstr, first, middle, last );

    CHECK(strcmp( first, fAnswer) == 0);
    CHECK(strcmp( middle, mAnswer) == 0);
    REQUIRE(strcmp( last, lAnswer) == 0);
}
```