Homework 7
Due Monday February 12, 2018
C++ Strings and Catch

Project Name:  hw7
File names: hw7.cpp, stringplus.cpp and stringplus.h

You may only write two functions for this assignment.  You may not write any other functions.

# Setup:

Download hw7.cpp from D2L – already has prewritten test cases.

```
#define CATCH_CONFIG_MAIN
#include <string>
#include "stringplus.h"
#include "..\catch.hpp"
using std::string;
```

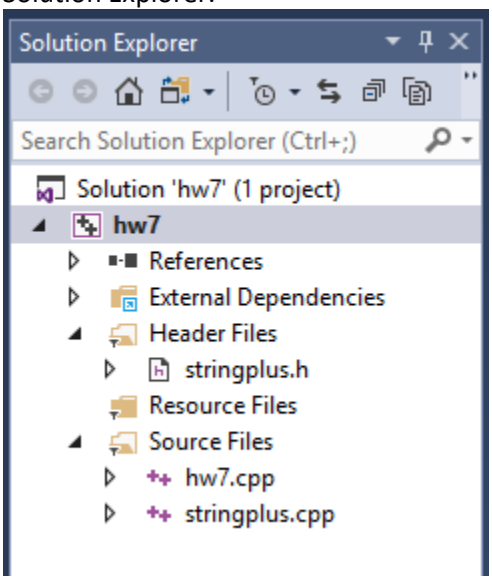
Stringplus.h – add new source file to solution.

```
#include <string>
#include <cctype>
using namespace std;

void trimString( string &s, char method );
void extractStringNames( string fullname,
                         string &first, string &middle, string &last);
```

Stringplus.cpp – add new header file to solution

```
#include "stringplus.h"
void trimString( string &s, char method )
{
    // your code goes here

}
void extractStringNames( string fullname,
                         string &first, string &middle, string &last)
{
    // your code goes here
}
```

Solution Explorer.



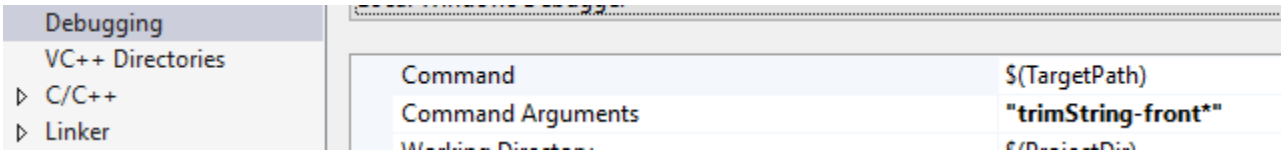
# Prototypes:

```
void trimString( string &s, char method );
void extractStringNames( string fullname,
                         string &first, string &middle, string &last);
```

You will develop your code while testing against pre written test.
The following scheme was used when naming the test cases:

| | | |
|---|---|---|
| trimString-front | extractStringNames-fml | extraCr |
| trimString-end | extractStringNames-fl | |
| trimString-both | extractStringNames-lfm | |
| trimString-invalid | extractStringNames-lf | |

You can specify what tests you want by providing an argument (filter) on the command line.
Example, tor run all the tests that deal with removing whitespace from the front of the string you would provide an argument of "trimString-front*". Yes the double quotes are required if the filter has a space in it.
From Visual studio, right click your project name and select properties. In the new window, click debugging and provide the argument in the right pane.




To make these functions easier, you will pick whether to work on the "trimString-front" or the "trimString-end" first. You will get all of the test cases to pass, and then switch to another filter. After that is working, switch to another filter, and so on. To run all the test case, change the filter to "trimString*".

For the Name names from a c++ string, pick a filter between extractStringNames-fml and extractStringNames-lfm. Make it pass all the test and then switch to the version that is missing the middle name.
Example: "extractStringNames-lfm*" and then work on “extractStringNames-lf *" ← note the space
"extractStringNames-fml" and then work on "extractStringNames-fl*“

After you think you have your function working, run them against all test cases for that function.
Example "trimString*" and "extractStringNames*".

# Trim String Function:

trimString will trim all whitespace from a c++ string based on the method given. The method controls what end the whitespace will be removed from. An invalid method should not alter the given string at all. The trim function will never alter the middle of the string, it will only remove from the front and end.

Method

'F' or 'f' will remove any whitespace characters from the front of the string.
'E' or 'e' will remove any whitespace characters from the end of the string.
'B' or 'b' will remove any whitespace characters from the both ends of the string.

Algorithm:

If the method is invalid return.
If the method is an F or a B, remove all whitespace from the front of the string.
If the method is a E or a B, remove all whitespace from the end of the string.

Consider using an 2 if statements for the actual trim.

** If ( method == 'f' || method == 'b' ) and the statement if( method == 'e' || method == 'b' )

This will minimize your code size.

Front - Use a loop to walk forward in the string and find the first non whitespace character and then remove the characters from the c++ string.

End – Use a loop to walk backwards in the string and find the first non whitespace character and then remove the characters that are after it.

Functions I would look at using:

isspace from the cctype library.

erase from the c++ string library.

```
string str;
Str.erase( );
```

You may not write a loop that copies the data over the data you wish to remove. Make one call to erase n number characters from the string.

# Extract String Names Function:

extractStringNames will parse the first and last names from a given c++ string. The string may have the middle name also. The format of the names will be one of the following:

Last, First
Last, First Middle
First Last
First Middle Last.

Any amount of **spaces or tabs** can be at the front, middle or end of the string.
I would consider using your trim function to remove excessive whitespace from the strings.

**With the trim function.**
Trim the string.
Find the first whitespace character moving forward through the string.
Extract this substring and then erase this substring from the name
Call the trim function again.
Find the first whitespace character moving forward through the string.
Extract this substring and then erase this substring from the name.
Call the trim function again.
If the string is not empty, extract the remaining characters to the third string.

**Without the trim function**
Find the first non-whitespace character. Then continue from this spot and find the first whitespace character.
Extract the substring between the two spots.
Continue looking for the next non-whitespace character from where the above loop was left off.
Then find the first whitespace characters after it.
Extract the substring between these two spots.
Continue looking for the next non-whitespace character from where the above loop left off.
If You find a non-whitespace character and did not go past the end of the string. Extract this substring to the third string.

I guarantee that I will pass in a string that contains the first and last names, the middle name is optional. Any amount of space and tab characters can be at the front, middle or end of the string. I will guarantee you that at least on space or tab separates the first, middle and last names.

Examples " \t Roger \t Schrader \t\t\t "
"\t\t Schrader,\tRoger\tLewis \t"

Functions I would look at using:

String str;
trimString
str.find()
str.substr()
str.erase()

**Extra Credit –** If you get your code to pass all the trimString tests and all the extractStringName tests you may work on the extra credit. The filter "extraCr*" will allow you to run these test cases. The format of the string does not change but any whitespace character can be found between the names and you are not guaranteed a space will be between the names. You will only get the extra credit if your code passes all the tests for trimming the string and extracting the names.

Example:

# Grading breakdown.

| | |
|---|---|
| trimString function – | 1 point front (must pass all trimString-front* tests) |
| | 1 point end (must pass all trimString-end* tests) |
| | 2 points both (must pass all trimString* tests) |
| extractString Names – | 2 points fml & fl (must pass all extractStringNames-f* tests) |
| | 2 points lfm & lf (must pass all extractStringNames-l* tests) |
| Extra Credit – | 2 points (passess all trimString*, extractStringNames* and extraCR* tests) |

**The only libraries you may use are <string> and <cctype>**
**Make sure and pull your repo to update.**
**Make sure and push your homework to the remote repository.**