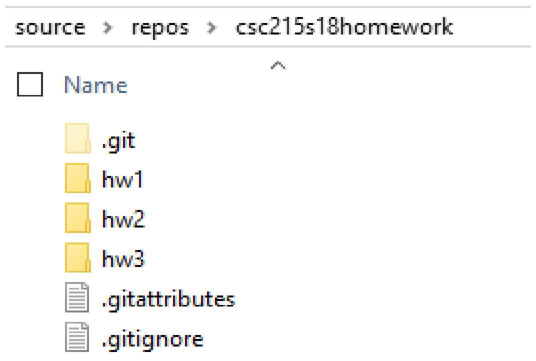
Homework 3 – Testing Hw 2

Due Monday 1/29 at class time. I will pull the repos at 3:00pm.

Create another project in your homework repository (csc215s18homeworks).

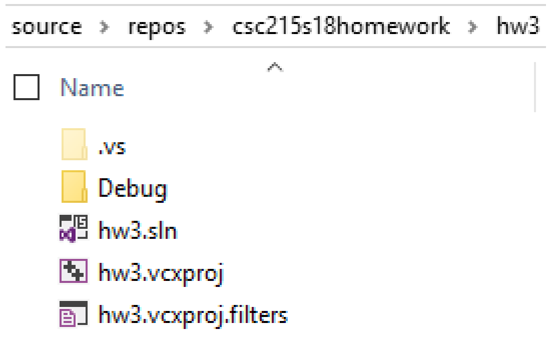
Project Name: **hw3**.

Source filename: hw3.cpp


source > repos > csc215s18homework

Name

.git
hw1
hw2
hw3
.gitattributes
.gitignore

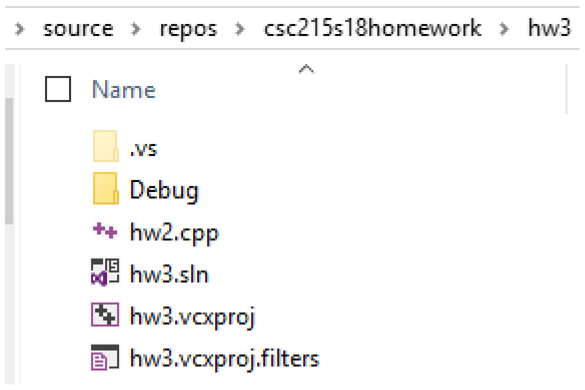

source > repos > csc215s18homework > hw3

Name

.vs
Debug
hw3.sln
hw3.vcxproj
hw3.vcxproj.filters

Copy hw2.cpp from hw2 project to this project and rename it as **dates.cpp**

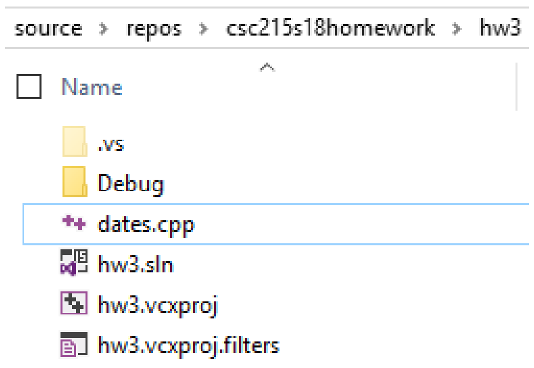
Right click the file hw2.cpp → select rename and type dates.cpp. Make sure you are in hw3 directory.


> source > repos > csc215s18homework > hw3

Name

.vs
Debug
hw2.cpp
hw3.sln
hw3.vcxproj
hw3.vcxproj.filters

→


source > repos > csc215s18homework > hw3

Name

.vs
Debug
dates.cpp
hw3.sln
hw3.vcxproj
hw3.vcxproj.filters

Add this file to your hw3 solution

From the solution Explorer, right click "Source Files" → "Add" → "Existing item"

Select **dates.cpp** from the csc215s18homeworks/hw3 directory.

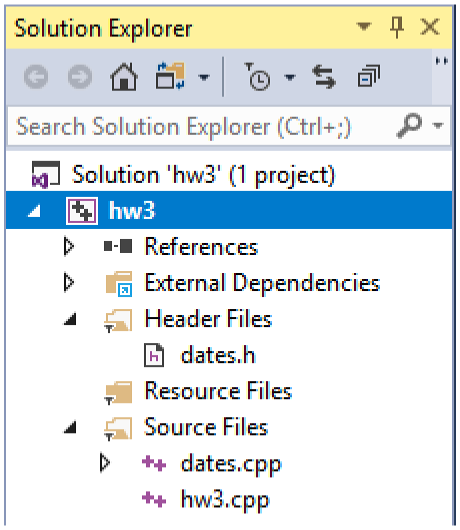
Add a header file name dates.h

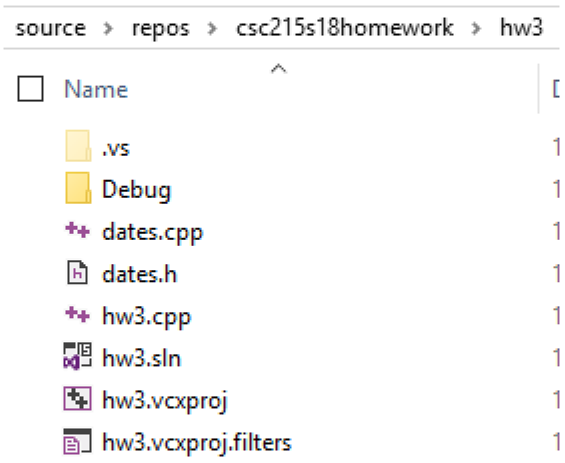
Add 2 more new files to the solution both created in the folder hw3.

Right Click "Source Files" → "Add" → "New Item"
Select "C++ File (.cpp)"
Create a cpp file named hw3.cpp

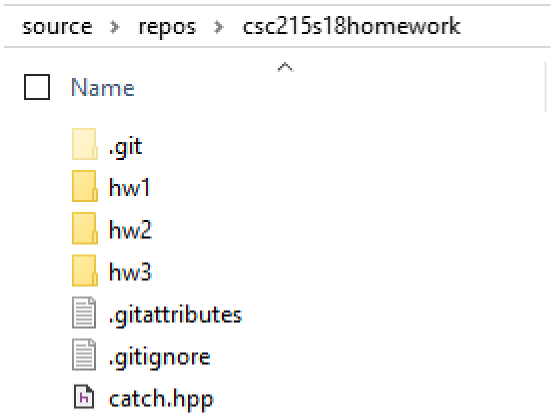
Right Click "Header Files" → "Add" → "New Item"
Select "header File (.h)"
Create another file named dates.h

Your Solution Explorer and hw3 should look like the following:





Download catch.hpp from the homework folder on d2l and save it to the csc215s18homeworks directory.

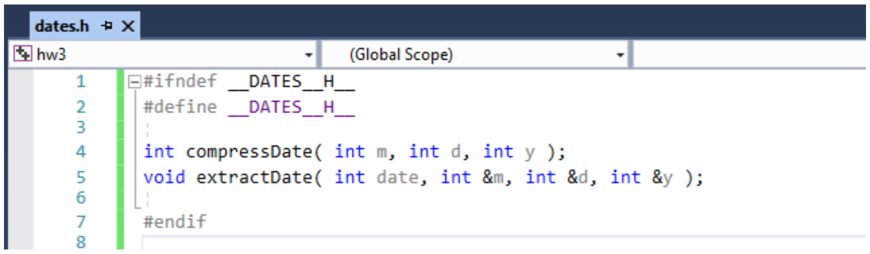


**dates.h**

Double click the dates.h file in your solution explorer. Erase any auto generated content if it exists.
Place the type the following code into dates.h

```
#ifndef __DATES__H__
#define __DATES__H__
int compressDate( int m, int d, int y );
void extractDate( int date, int &m, int &d, int &y );
#endif
```

The contents of dates.h should look like the following image. Save and close this file.

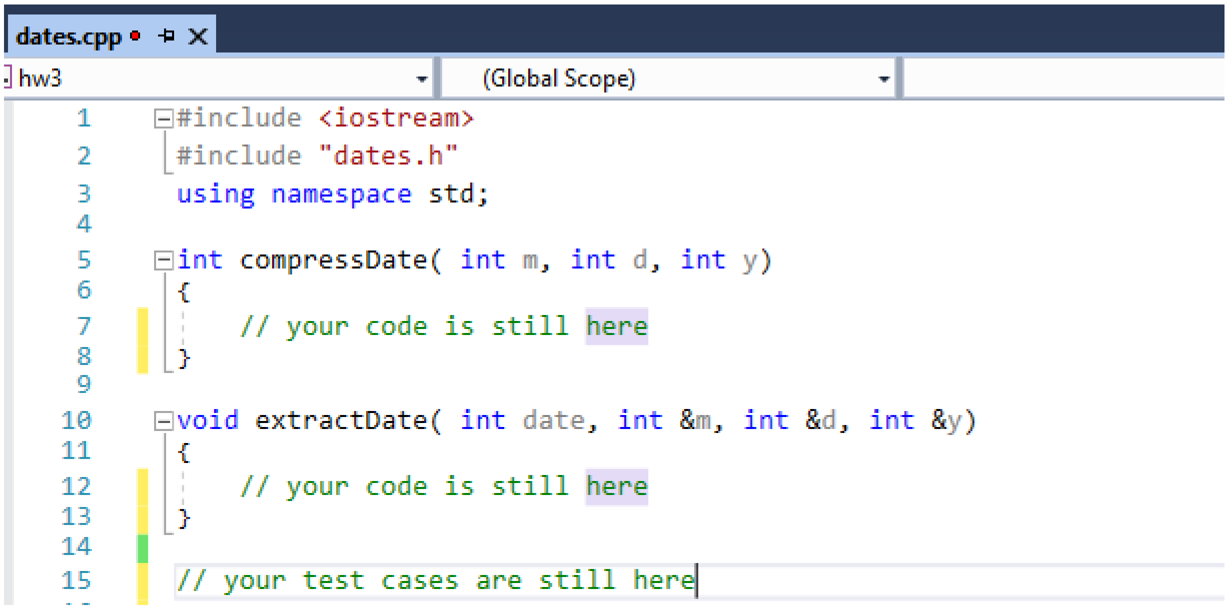


```
#ifndef __DATES__H__
#define __DATES__H__

int compressDate( int m, int d, int y );
void extractDate( int date, int &m, int &d, int &y );

#endif
```

**dates.cpp**

Delete your main function from this file.
Remove your two function prototypes form compressDate and extractDate.
Add a #include “dates.h” at the top of the file.

Your file should look like the following:

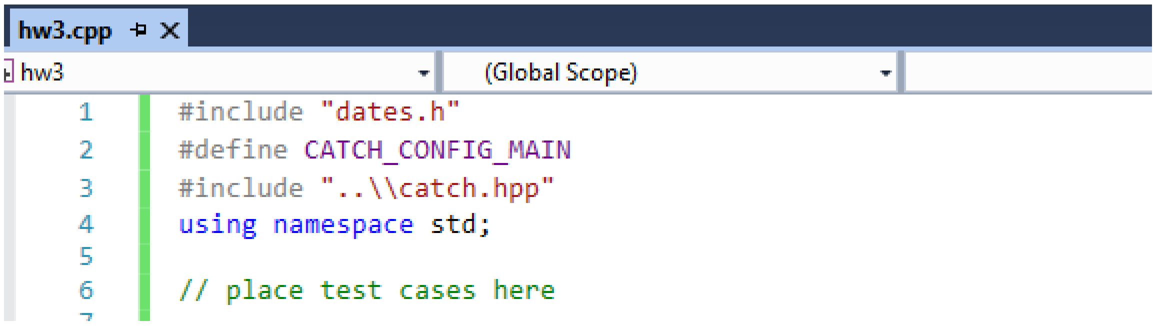


```cpp
#include <iostream>
#include "dates.h"
using namespace std;

int compressDate( int m, int d, int y)
{
    // your code is still here
}

void extractDate( int date, int &m, int &d, int &y)
{
    // your code is still here
}

// your test cases are still here
```

**hw3.cpp**
Testing framework
Place the following code into hw3.cpp

```cpp
#include "dates.h"
#define CATCH_CONFIG_MAIN
#include "..\\catch.hpp"
using namespace std;

// place test cases here
```

Hw3.cpp should look like the following:



```cpp
#include "dates.h"
#define CATCH_CONFIG_MAIN
#include "..\\catch.hpp"
using namespace std;

// place test cases here
```

# How to write Test Cases for catch

TEST_CASE (   string  )  – starts a unit test of some function

The string within the parenthesis describes what is being tested with given data and what you expect.  Each string Must be unique because it identifies a single test that is testing one item.

Then is the code snippet that runs your test case.

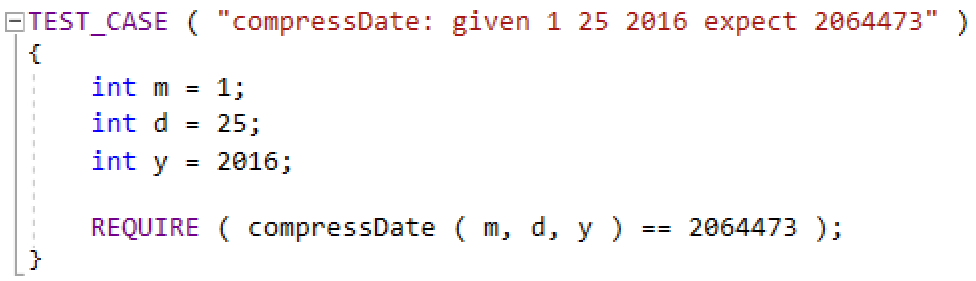
```
{
        // C++ code to preform the test goes here

        REQUIRE (   //single logical condition to see if it was correct );
}
```

In a nutshell, it looks very similar to a function.

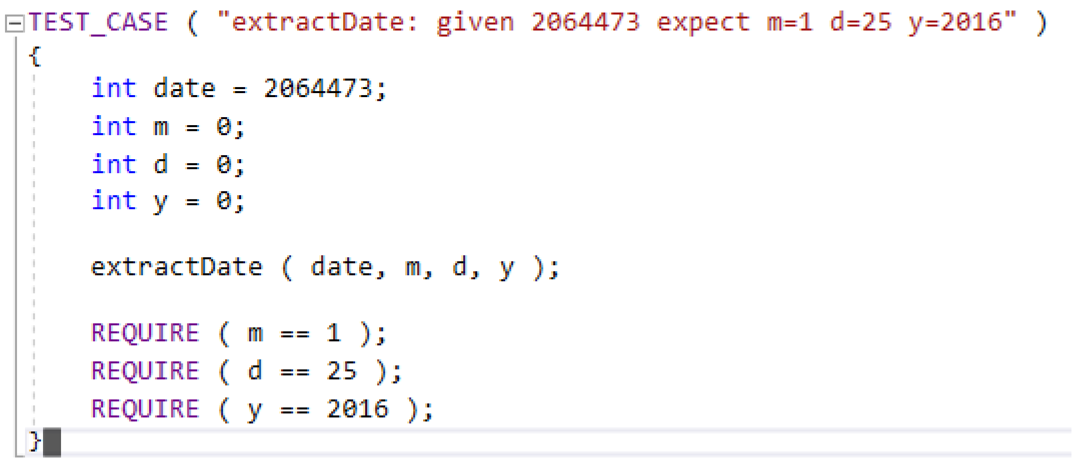
```
TEST_CASE (“functionName: given data expect answer”)
{
        int number =10;

        REQUIRE( functionName( number ) == false );
}
```

You can have whatever code snippet is needed to test your function.
These are known as unit tests.  The REQUIRE line what is known as an assertion and test the results from the code snippet.  You can have more than one assertion per test case.

Examples for my test case.



```
TEST_CASE ( "compressDate: given 1 25 2016 expect 2064473" )
{
    int m = 1;
    int d = 25;
    int y = 2016;

    REQUIRE ( compressDate ( m, d, y ) == 2064473 );
}
```

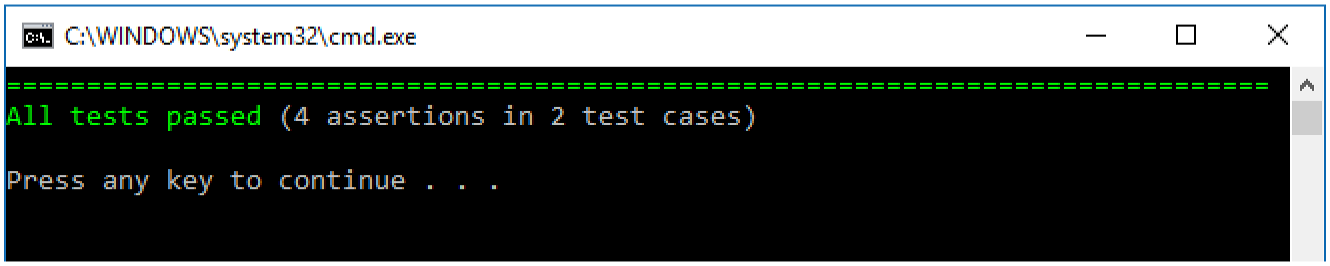


```
TEST_CASE ( "extractDate: given 2064473 expect m=1 d=25 y=2016" )
{
    int date = 2064473;
    int m = 0;
    int d = 0;
    int y = 0;

    extractDate ( date, m, d, y );

    REQUIRE ( m == 1 );
    REQUIRE ( d == 25 );
    REQUIRE ( y == 2016 );
}
```

Running your tests.

Since we now have a two test cases, Run your code without debugging.

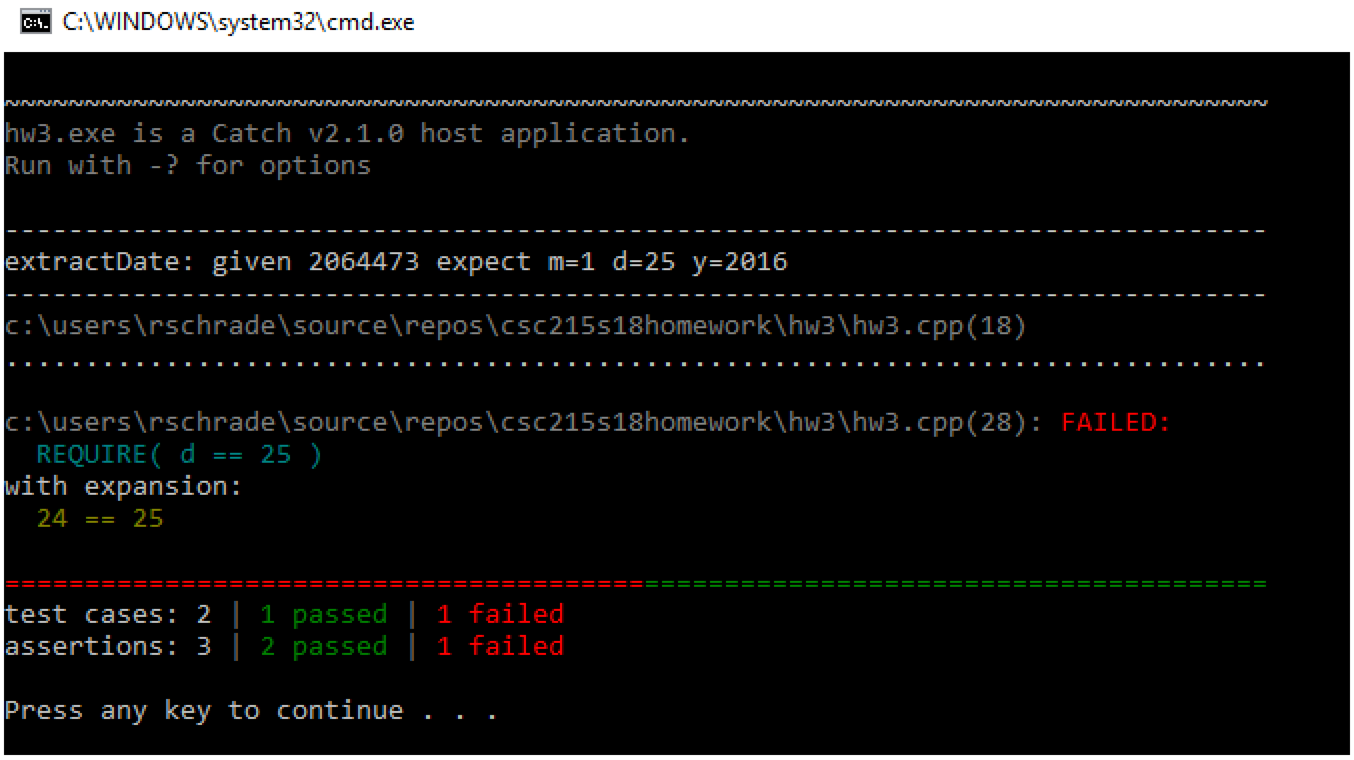
You should see a window like the following:



```
===============================================================================
All tests passed (4 assertions in 2 test cases)

Press any key to continue . . .
```

Place your three test cases into the TEST_CASE functions to do unit testing. Remember, the example I gave does not count. You must come up with at least 3 test cases for each function. See if you can come up with something that will cause your functions to fail.

Run your test cases and make sure they pass. If one fails, it should look something like this:



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
hw3.exe is a Catch v2.1.0 host application.
Run with -? for options

-------------------------------------------------------------------------------
extractDate: given 2064473 expect m=1 d=25 y=2016
-------------------------------------------------------------------------------
c:\users\rschrade\source\repos\csc215s18homework\hw3\hw3.cpp(18)
...............................................................................

c:\users\rschrade\source\repos\csc215s18homework\hw3\hw3.cpp(28): FAILED:
  REQUIRE( d == 25 )
with expansion:
  24 == 25

===============================================================================
test cases: 2 | 1 passed | 1 failed
assertions: 3 | 2 passed | 1 failed

Press any key to continue . . .
```

Reading a failed test case (I changed my date number to make it fail.)

The white line of text with dashes above and below tell you what test case failed.

It shows you that the d was required to be 25 to pass the overall test. (REQUIRE( d == 25 ))

Then it shows you the values it actually used for the require ( 24 == 25 ).

Now we can figure out if our function is wrong or was the testcase wrong.

Put the program header at the top of the hw3.cpp file and commit to local repository

Push the homework repository to the remote server by 2:00 pm on Monday.