Project 2 Is Due Tonight At 9 PM
- I Appreciate Your Patience Over The Next Few Days As The TAs And I Get Them Scored
Project 3 Has Been Released Today
- Due July 24th At 9 PM
- We Need To Discuss A Bit About Strings And String Handling
Midterm
- Scheduled for Discussion Section time on Friday July 21st
  o Please Attend The Section You Are Enrolled In
- Covers Most Everything Up Until Now
- No Books, No Computers, No Devices
- You Can Bring In 1 8 1/2 x 11 Sheet Of Paper With Notes
  o Can Be Both Sides
  o Can Be Typed Or Scribbled
- Worth 100 Points
- I Will Release Sample Problems Over The Weekend
  o We Will Discuss Them Next Wednesday
- No Bluebook Or Scantron Needed
- More Information About Its Form Monday, Once I Finalize It...

// code that walks a string letter - by - letter
// loop       process each letter

string s = "Hello";        /// 5
for( int i = 0; i < s.size( ); i++ )  /// counter is I
for( size_t i = 0; i < s.size( ); i++ )
// warnings   int is a signed quantity    + & -
// size of a string is an unsigned quantity    0 or more
{
    //0, 1, 2, 3, 4
    // first letter is in position 0
    // last letter is in position .size()-1
    // grab an individual letter from a string....
    char letter = s[ i ]; // square brackets offset to a particular
                          // no checking... at runtime bad value...
    char letter = s.at( i );  /// exact same thing    does bounds checking..
if (letter == "H")
{
    // is this letter in the string....
}
if (letter == 'H')
{
    // is this letter in the string...
}
switch( letter )
{
    case 'A':
    case 'E':
    case 'I':
    case 'O':
    case 'U':
    case 'Y':
    cout << " a vowel " << endl;
    break;
    default:
    cout << " not a vowel " << endl;
    break;
}
cout << "the letter at position " << l << " = " << letter << endl;
}

Goals: supplying you some code that turns
"51" ----> int value of 51
"1000" ---> 1000

'5' ----> 5
char five = '5';       ---> value 5
int value = five - '0';

Convert digit sets of characters into their int equivalent
"21" ----> value 21
    loop that works letter by letter
    accumulate each digit - '0';
basis 10   * 10


2 -------> value 2
1 -------> 2*10 + 1

"210"

// bullseye...
// "aim at the bullseye"
// even though it does not exist,
// predicting the results
#include <cassert>

int number( string s, int& pos );

int main( )
{
    assert( number( ":", 0 ) == -1);
    assert( number( "123a456", 0 ) == 123 );
    assert( number( "123", 1 ) == 23 );
    assert( number( "123", 2 ) == 3 );
    assert( number( "123", 3 ) == -1 );
    assert( number( "123", 4 ) == -1 );
    assert( number( "Howard", 0 ) == -1 );
    assert( number( "1900 Pico Boulevard", 0 ) == 1900 );
}

// work:   "123"  --->  123
// positively and negatively
//       "Howard Stahl"  --->  -1
// simple----> start from the beginning
// accept another parameter
int number( string s, int& pos )
{
    int result = 0;
    // walk letter-by-letter, '0' - '9'
    // not worry about a + -
    // starting from the beginning pos: 0
int pos = 0;
// digit character
// problem: be sure we don't walk off the string
// problem: no digit character...
// test for a non-digit....
if (s.length() > pos)
{
  if (s[ pos ] < '0' || s[ pos ] > '9')
  {
    // failure
    result = -1;
  }
  else
  {
    while( s[ pos ] >= '0' && s[ pos ] <= '9')
    {
      // accumulate into result the ongoing value...
      //  1900  1 9 0
      result = result * 10 + (s[ pos ] - '0');
      pos = pos + 1;
      if (pos == s.length()) break;
    }
  }
}
else
{
  result = -1;
}

return( result );

Parameter Passing Schemes
**CALLER**
The pile of code that invokes the function

**CALLEE**
The pile of code that gets called

<table>
<thead>
<tr>
<th>Pass-By-Value</th>
<th>Declared As:</th>
<th>Called As:</th>
<th>&quot;Very Safe&quot; Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>void foo(int i);</td>
<td>foo( 12 );</td>
<td>- Nothing The Callee Does</td>
</tr>
<tr>
<td></td>
<td>void foo( int );</td>
<td>foo( 'A' );</td>
<td>Affects The Caller's Variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>foo( true );</td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>foo( j+1 );</td>
<td>- The Only Communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>foo( j );</td>
<td>Between The Caller And The</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Callee Will Be With The Return</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Argument, If Not void And</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Captured By The Caller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- What Is Sent Is An &quot;R-Value&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Of The Desired Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- What Arrives Is An</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Independent Copy Of The</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Caller's Original Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- The Callee Can Change The</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Value But The Caller's Variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Will Not Change</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- INBOUND</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pass-By-Reference</th>
<th>Declared As:</th>
<th>Called As:</th>
<th>Means:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>void foo( int &amp; a );</td>
<td>foo( i );</td>
<td>&quot;unsafe&quot; can cause some</td>
</tr>
<tr>
<td></td>
<td></td>
<td>foo( j );</td>
<td>&quot;side effects&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;strict&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The Caller Can Change The</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Value Of The Caller's Variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>What Needs To Be Passed Is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The Exact Right Matching Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>l-value Of The Right Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Variable Of The Exact Right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No Copy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The &quot;Actual Thing&quot; Is What Is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Being Sent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If You Want The Caller To See</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The Changed Value, This Is The</td>
</tr>
</tbody>
</table>
Project 1 Original Code:

```cpp
int main()
{
    int postsReviewed;
    int fakePosts;
    int realPosts;
    cout << "How many internet posts were reviewed? ";
    cin >> postsReviewed;
    cout<<"How many of these post were fake news items?";
    cin >> fakePosts;
    cout<<"How many of these days were real news items?";
    cin >> realPosts;

    double pctFake = 100.0 * fakePosts / postsReviewed;
    double pctReal = 100.0 * realPosts / postsReviewed;
    cout.setf(ios::fixed); // see Savitch book
    cout.precision(1);
    cout << endl;
    cout << pctFake << "% were fake." << endl;
    cout << pctReal << "% were real." << endl;
    if (pctFake > pctReal)
        cout << "It was more fake than real." << endl;
    else
        cout << "It was more real than fake." << endl;

    return( 0 );
}
```

Sample Redo With Functions #1
#include <cstdlib>   /// for exit( )
void foo( );    // bit of orchestration
   // not communicating with the
caller
int main( )
{
    foo( );
    return( 0 );
}

void foo()
{
    int postsReviewed;
    int fakePosts;
    int realPosts;
    cout << "How many internet posts were
reviewed? ";
    cin >> postsReviewed;
    cout<"How many of these post were fake news
items?";
    cin >> fakePosts;
    cout<"How many of these days were real news
items?";
    cin >> realPosts;

    double pctFake = 100.0 * fakePosts / postsReviewed;
    double pctReal = 100.0 * realPosts / postsReviewed;
    cout.setf(ios::fixed);  // see Savitch book
    cout.precision(1);
    cout << "% were fake." << endl;
    cout << pctReal << "% were real." << endl;
    if (pctFake > pctReal)
        cout << "It was more fake than real."
        << endl;
    else
        cout << "It was more real than fake."
        << endl;
}

Sample Redo With Functions #2
July 12 Page 7
Sample Redo With Functions #2

```cpp
int posts( );
int fake( );
int real( );
void adjustcout( );
void output( double fake, double real );

int main( )
{
    // large task, breaking into pieces
    // code each part separately
    // orchestration calling all the pieces
    // stitching into the program
    int postsReviewed = posts( );
    int fakePosts = fake( );
    int realPosts = real( );

    double pctFake = 100.0 * fakePosts / postsReviewed;
    double pctReal = 100.0 * realPosts / postsReviewed;

    adjustcout( );
    output( pctFake, pctReal );
    return( 0 );
}

int posts( )
{
    int postsReviewed;
    cout << "How many internet posts were reviewed? ";
    cin >> postsReviewed;
    return( postsReviewed );
}

int fake( )
{
    int fakePosts;
    cout<<"How many of these post were fake news items?";
    cin >> fakePosts;
```
return( fakePosts );
}

int real( )
{
    int realPosts;
    cout<<"How many of these days were real news items?";
    cin >> realPosts;
    return( realPosts );
}

void adjustcout( )
{
    cout.setf(ios::fixed);
    cout.precision(1);
}

void output( double fake, double real )
{
    cout << endl;
    cout << fake<< "% were fake." << endl;
    cout << real << "% were real." << endl;
    if (fake > real)
        cout << "It was more fake than real." << endl;
    else
        cout << "It was more real than fake." << endl;
}

// declaration
Void swap( int& I, int& j );   /// reference to an int...
// alias

// get c++ not use pass-by value
// pass-by-reference
Void swap( int & I , int & j )
{
    int temp = I;
    I = j;
Main
Int a = 5, b = 10;
Swap( a, b );
Assert( a == 10 );
Assert( b == 5 );

Sample Redo With Functions #3 --- DOESN'T WORK! Do You See Why??

// set the value of 3 things, not just 1…
void input( int posts, int fake, int real );
void adjustcout( );
void output( double fake, double real );

int main( )
{
    int postsReviewed;
    int fakePosts;
    int realPosts;
    // input only variables
    input(postsReviewed, fakePosts, realPosts );

    double pctFake = 100.0 * fakePosts / postsReviewed;
    double pctReal = 100.0 * realPosts / postsReviewed;

    adjustcout( );
    output( pctFake, pctReal );
    return( 0 );
}

void input( int postsReviewed,
int fakePosts,
    int realPosts     )
{
    cout << "How many internet posts were reviewed? ";
    cin >> postsReviewed;
    cout<"How many of these post were fake news items?";
    cin >> fakePosts;
    cout<"How many of these days were real news items?";
    cin >> realPosts;
}

void adjustcout( )
{
    cout.setf(ios::fixed);
    cout.precision(1);
}

void output( double fake, double real )
{
    cout << endl;
    cout << fake<< "% were fake." << endl;
    cout << real << "% were real." << endl;
    if (fake > real)
        cout << "It was more fake than real." << endl;
    else
        cout << "It was more real than fake." << endl;
}

Sample Redo With Functions #4 --- MUCH BETTER! Do You See Why??

// set the value of 3 things, not just 1...
void input( int& posts, int& fake, int& real );
void adjustcout( );
void output( double fake, double real );

int main( )
int postsReviewed;
int fakePosts;
int realPosts;
// input only variables
input(postsReviewed, fakePosts, realPosts);

double pctFake = 100.0 * fakePosts / postsReviewed;
double pctReal = 100.0 * realPosts / postsReviewed;

adjustcout();
output(pctFake, pctReal);
return( 0 );

void input( int& postsReviewed,
            int& fakePosts,
            int& realPosts )
{
    cout << "How many internet posts were reviewed? ";
cin >> postsReviewed;
    cout<<" postsReviewed;"
cin >> fakePosts;
    cout<<" How many of these post were fake news items?";    
cin >> fakePosts;
    cout<<" How many of these days were real news items?";    
cin >> realPosts;
}

void adjustcout( )
{
    cout.setf(ios::fixed);
    cout.precision(1);
}

void output( double fake, double real )
{
    cout << endl;
    cout << fake << "% were fake." << endl;
    cout << real << "% were real." << endl;
}
if (fake > real)
    cout << "It was more fake than real."
    << endl;
else
    cout << "It was more real than fake."
    << endl;
}

A Bit More About String
- Each string is a set of characters. For machines using the ASCII code, each character is one of those listed in the table shown at: http://www.asciitable.com

string orderstring;
getline( cin, orderstring);
// C++ starts counting with 0
// spot 0 will be the first letter of the string
char firstLetter = orderstring[ 0 ];
// single letter
// unsigned int  not exactly correct...
size_t length = orderstring.size( );
size_t length = orderstring.length( );

// loop to walk the entire string...
for ( size_t i = 0; i < orderstring.size( ); i++ )
{
    char c = orderstring[ i ];
    cout << c << endl;
}

- Turning the string data into int equivalency
"123" ----> 123

#include <cassert>
#include <iostream>
#include <string>
using namespace std;
int number( string s, int startingposition );

int main ( )
{
    cout << number( "1900 Pico Boulevard", 0 )
        << endl;
    // prints: 1900

    assert( number("1912 Pico Boulevard", 0)==
            1912 ); // ----> 1912
    assert( number("1912 Pico Boulevard", 1 )==912 ); // ----> 912
    assert( number("1912 Pico Boulevard", 2 )
            == 12 ); // ----> 12
    assert( number("1912 Pico Boulevard", 3 )
            == 2 ); // ----> 2
    assert( number("1912 Pico Boulevard", 10 )
            == 0 ); // ----> 0

    return( 0 );
}

int number( string s, int startingposition )
{
    int result = 0;    // answer
    int length = s.length();
    if (length > startingposition &&
            s[ startingposition ] >= '0' &&
            s[ startingposition ] <= '9')
    {
        while( s[ startingposition ] >= '0' &&
            s[ startingposition ] <= '9' )
        {
            int value = s[ startingposition ] - '0';
            result = ( result * 10 ) + value;
            startingposition = startingposition + 1;
        }
        // move on down the road...
    }
    if (startingposition == length)
    {
        break;
    }
}
"123" result = 123 '3' ----> 3