Project 3 Scores Have Been Released
  - Out Of 100, The Average Was 76.8085 But Factoring Out Code That Did Not Build, The Average Was 84.95
  - Half The Scores Were 89 Or Higher But Factoring Out Code That Did Not Build, The Median Was 93

![Project 3 Scores Chart]

Project 4 Come Due Wednesday
  - Late Work Being Accepted Until Thursday Morning 7AM
  - Discussion Board Questions
    ○ String array[ 5 ] = { "1", "2", "3", "4", "5" };  
      // some code
      array = { "1", "2", "3", "4", "5" };  // ILLEGAL!!
    ○ String array[ 5 ] = { "1", "2", "3", "4", "5" };  
      // some code
      countIntegers( array, 5 );  /// 5
      countIntegers( anotherArray, 5 );  /// 2
    ○ What is meant by an empty array?
countIntegers( array, 0 );  // not safe to access any
// part of the array

○ While sorting is not required, if you sort the array that
will be a correct answer to the divide function.
And yes, there are other valid solutions that don't
require a completely sorted answer

- Any Other Questions?
  ○ Atleast Create All The Assigned Functions

```c
int divide( string array[ ], int n, string divider )
{
    return( 0 );
}

Assert( divide( array, n, "value" )
```

Course Content
- Bank Account class
- But What If Something Goes Wrong....

/// DRIVER CODE...
bankAccount muffin( "muffin", -100.00 );
// RIGHT HERE..... Should not continue....

cout << "The bankAccount for " << muffin.getName() << " has " << muffin.getBalance();
muffin.deposit( 10 );
muffin.withdraw( 10000000 );
muffin.withdraw( -100000 );

/// CLASS CODE
bankAccount::bankAccount( )      ///// public
{
    /// role.... Initialize all your member variables....
my_name = "";
my_balance = 0.00;
}

bankAccount::bankAccount( string name, double balance )
///// public
{
    ///< role.... Initialize all your member variables....
    my_name = name;
    my_balance = balance;
    if (balance < 0 )
    {
        cout << "An error happened..." << endl;
    }
}

- Introducing std::logic_error
    #include <stdexcept>

    std::logic_error

    - message : string

    + logic_error( message : string )
    + what( ) : string

    ///// BETTER CLASS CODE
    bankAccount::bankAccount( string name, double balance )
    ///// public
{ 
  if (balance >= 0) 
  { 
    /// role.... Initialize all your member varialbes....
    my_name = name;
    my_balance = balance;
  }
  else 
  { 
    std::logic_error 
    e( "you can't have a balance less than 0!" );
    throw( e );  /// like a return communicating failure
  }
}

/// Better DRIVER CODE...
try
{
  bankAccount muffin( "muffin", -100.00 );
  /// fail....
  /// throwing logic_error
  ///   catch logic_error
  ///   die dead in tracks...
  cout << "The bankAccount for " << muffin.getName() 
        << " has " << muffin.getBalance();
  muffin.deposit( 10 );
  muffin.withdraw( 1000000 );
} catch( logic_error error ) {
  cout << "something went wrong.... " << error.what( )
  << endl;
} catch( overdrawnAccount variable ) {
  variable;
}
POINTER VARIABLES....

Int a = 12;
Int * ptrInt;   /// unusable... xcccccccc
ptrInt = NULL;  /// unusable sentinel
ptrInt = nullptr; /// test for checking
ptrInt = &a;
Cout << *ptrInt;

a = 13;
/// same thing done.... Using ptr variable ptrInt
*ptrInt = 13;  /// dereference walks the arrow

// first heap command

int * ptr;
ptr = new int( 12 );  /// dynamic variable
    /// do not die off until you say so...
*ptr = 13;
/// give this memory back....
delete( ptr );
delete ptr;

/// deal: whatever you new, you must delete

// same example again: classtype rather than int

bankAccount b;
b.deposit( 100 );
bankAccount * ptr;
ptr = & b;
Ptr -> deposit( 100 );
(*ptr).deposit( 100 );       /// same thing....

///// -> will crash your program if the pointer you arrowing
///// is null or not defined....

bankAccount * b = new bankAccount( "muffin", 100 );
b->withdraw( 100 );
delete b;
delete( b );

///// a set of items in advance how big that blob
int array[ 10 ];

Int l = 10;
Int array[ l ];  /// could not be based on a variable...
// illegal
// new and delete  break this constraint
///  dynamic array

Int I = 10;
Cin >> I;

/// dynamic array
int * array = new int[ i ];
array[ 0 ] = 12;
array[ 1 ] = 13;
/// give it back...
delete [ ] array;  /// [] required