**SET-252 C Programming #2**

#### Homework: SuperString – Final Project ( 30% of your final grade )

**Warning: You cannot consult with any student that has already completed this project.**

Warning: You cannot view, use or reference any code for this project written by students that have taken this course in the past.

**Step 1**

* Implement the following class header file (i.e. write the cpp file).
* You may add any other properties and/or methods that you need. You might want to check with me before adding them just to make sure you're on the right track.
* Write a main function that will test all the different methods. Make sure your main function is easy to read and understand. I recommend you use regions and procedures to break up the logic. You can easily comment out a call to previously made procedures so you can test only the current procedure that you are working on.
* You may use any string functions you want (e.g. strlen, strstr, etc).
* You may not use the built-in String class.
* In addition to the methods listed below, overload the following operators with all reasonable parameters (i.e. char, const char \*, CSuperString &): ==, !=, >, <, >=, <=.
* NONE of the class methods other than the constructors or the assignment operators ( = and += ) should change m\_pstrSuperString.
* You must assign the value back to the class instance to make it permanent. For example:   
  ssBuffer = ssBuffer.ToUpperCase( );
* Minus one letter grade for any memory leak. Minus two letter grades if there are a lot of memory leaks.
* Minus one letter grade if replace with substrings doesn't work correctly.
* Minus one letter grade if self assignment doesn’t work correctly (e.g. ssBuffer = ssBuffer.ToUpperCase( ); ).
* Minus one letter grade if you don't solve the double call problem. The following should display correctly:

printf( "Left( 2 ): %s, Left( 4 ): %s\n", ssBuffer.Left( 2 ), ssBuffer.Left( 4 ) );

Splitting the code up into two lines is not the solution.

* Add the following procedure to your tests and make sure it executes without errors.

void MuahahahTest1( )

{

CSuperString ssTest;

cout << “Muahahah Test #1: “ << ssTest << endl;

}

* Add the following procedure to your tests and make sure it executes without errors.

void MuahahahTest2( )

{

CSuperString ssTest = "I Love Star Wars and I Love Star Trek";

ssTest = ssTest.Replace( "Love","Really Love Love" );

cout << “Muahahah Test #2: “ << ssTest << endl;

}

* I recommend you start with just the following methods, get those working and code, test and run additional methods one at a time: constructor, initialize, assignment operator, deep copy, to string and print.
* Next, I’d recommend you get all the other constructors done.
* After that, I’d recommend you get all the “to <data type>” methods done.

// --------------------------------------------------------------------------------

// Class: CSuperString

// --------------------------------------------------------------------------------

class CSuperString

{

private:

char\* m\_pstrSuperString;

public:

// Constructors

CSuperString( );

// Parameterized/Copy constructors

// CSuperString ssBuffer( "I Love Star Trek" );

// CSuperString ssBuffer( 3.14159f );

CSuperString( const char \*pstrStringToCopy );

CSuperString( const bool blnBooleanToCopy );

CSuperString( const char chrLetterToCopy );

CSuperString( const short shtShortToCopy );

CSuperString( const int intIntegerToCopy );

CSuperString( const long lngLongToCopy );

CSuperString( const float sngFloatToCopy );

CSuperString( const double dblDoubleToCopy );

CSuperString( const CSuperString &ssStringToCopy );

// Destructor

virtual ~CSuperString( );

long Length( ) const;

// Assignment Operators

void operator = ( const char \*pstrStringToCopy );

void operator = ( const char chrLetterToCopy );

void operator = ( const CSuperString &ssStringToCopy );

// Extra credit for numeric parameters

// Concatenate operator

void operator += ( const char \*pstrStringToAppend );

void operator += ( const char chrCharacterToAppend );

void operator += ( const CSuperString &ssStringToAppend );

// Extra credit for numeric parameters

friend CSuperString operator + ( const CSuperString &ssLeft,

const CSuperString &ssRight );

friend CSuperString operator + ( const char \* pstrLeftSide,

const CSuperString &ssRightString );

friend CSuperString operator + ( const CSuperString &ssLeftString,

const char \* pstrRightSide );

long FindFirstIndexOf ( const char chrLetterToFind );

long FindFirstIndexOf ( const char chrLetterToFind, long lngStartIndex );

long FindLastIndexOf ( const char chrLetterToFind );

long FindFirstIndexOf ( const char \*pstrSubStringToFind );

long FindFirstIndexOf ( const char \*pstrSubStringToFind, long lngStartIndex );

long FindLastIndexOf ( const char \*pstrSubStringToFind );

// Do not change original string. For example:

// cout << ssBuffer.ToString( ) << endl; // I Love Star Trek

// cout << ssBuffer.ToUpperCase( ) << endl; // I LOVE STAR TREK

// cout << ssBuffer.ToString( ) << endl; // I Love Star Trek

const char\* ToUpperCase( );

const char\* ToLowerCase( );

const char\* TrimLeft( );

const char\* TrimRight( );

const char\* Trim( );

const char\* Reverse( );

const char\* Left ( long lngCharactersToCopy );

const char\* Right ( long lngCharactersToCopy );

const char\* Substring ( long lngStart, long lngSubStringLength );

const char\* Replace ( char chrLetterToFind, char chrReplace );

// Hard

const char\* Replace ( const char \*pstrFind, const char\* pstrReplace );

const char\* Insert ( const char chrLetterToInsert, long lngIndex );

const char\* Insert ( const char \*pstrSubString, long lngIndex );

// Subscript operator

char& operator [ ] ( long lngIndex );

const char& operator [ ] ( long lngIndex ) const;

const char\* ToString ( );

bool ToBoolean ( );

short ToShort ( );

int ToInteger ( );

long ToLong ( );

float ToFloat ( );

double ToDouble ( );

// cin >> ssBuffer;

// cout << ssBuffer;

friend ostream& operator << ( ostream &osOut, const CSuperString &ssOutput );

friend istream& operator >> ( istream &isIn, CSuperString &ssInput );

// Don’t forget the comparison operators!!!

};