CString Practice File

Its better to make a header and put all the function in it. You should name files as CString.h and CString.cpp

```
class CString {
private:
    char * data;
    int size;
public:
    CString(); // Must Do
    //Initialize data and set the size to zero
    CString(char ch); // Must Do
    //Initialize data by character and set the size to 1
    CString(const char * string); // Must Do
    //Intialize data by the character array and set the size equal to length
    CString(const CString & ref); // Must Do
    //Copy Constructor
    ~CString(); // Must Do
    //Release resources from the data members
    int getSize() {
        return this->size;
    }
    void input() {
        if (size == 0 || data == nullptr)
            resize(10);
        if(getLength()>0)
            remove(0, getLength());
        cin.getline(data,size);
    }
    char & at(int index); // Must Do
    //Index -> Recieves the index for the string
    //Return Value -
> Reference of the aarray location represented by the index
    bool isEmpty(); // Must Do
    //Tells whether the string is empty or not
    bool isFull(); // Must Do
    //Tells whether the string is full or not
    void resize(int newSize); // Must Do
    //Resizes the data array to the newSize, sets the size to new value
    int getLength(); // Must Do
    //Returns the length of the string
    void display(); // Must Do
    //Displays the content of the string
    int find(CString & subString, int start = 0); // Must Do
    //Find the substring in the calling CString object.
    //By default, search starts from index 0
    //Returns the count of occurences found in the calling object
    void insert(int index, CString & subString); // Must Do
    //Insert the substring at given index in the calling object
    void remove(int index, int count = 1); // Must Do
    //Remove the characters starting from index
    //Count represents the number of characters to be removed
    int replace(CString & oldString, CString & newString); // Must Do
    // Find all the occurences of old substring and  replace it with new subst
ring
    // Return the count of occurrences found in the calling object
    void trimLeft();
    //Removes all the white spaces to the left of the string.
    //Old String = "   start of the string"
    //New String = "start of the string"
    void trimRight();
    //Removes all the white spaces to the right of the string.
    //Old String = "start of the string    "
    //New String = "start of the string"
    void trim();
    //Removes all the white spaces to the left and right of the string.
    //Old String = "   start of the string   "
    //New String = "start of the string"
    void makeUpper();
    //Change all the alphabets to uppercase
    void makeLower();
    //Change all the alphabets to lowercase
    void reverse();
    //Reverse the string stored in data
    int compare(CString & newString);
    //Behaves exactly likes strcmp and compares the caalling object to referen
ced object
    //Return Values
    //  < 0 - > The first character that does not match has a lower value
    //          than the referenced object's character
    //    0 - > Both strings are equal
    //  > 0 - > The first character that does not match has a greater value
    //          than the referenced object's character
    void concatEqual( const CString & s2 );
    //Adds the value from CString s2 to the array of calling object
    void concatEqual( const char * const s2 );
    //Adds the value from const char * s2 to the array of calling object
    bool isEqual(const CString & s2 ); // Must Do
    // Checks if the passed string is equal to the calling object string or no
t
    bool isEqual(const char * const s2 ) ; // Must Do
    // Checks if the passed string is equal to the calling object string or no
t
    CString concat( const CString& s2 ) const ; // Must Do
    // Concatinates the string passed to the value of the calling object in a
NEW Cstring
    // and returns the reference to the NEW CString
    CString concat( const char * const s2 ) const; // Must Do
    // Concatinates the string passed to the value of the calling object in a
NEW Cstring
    // and returns the reference to the NEW CString
    CString & operator = ( const CString & ref ); // Must Do
    CString & operator = ( const char * ref); // Must Do
    // Assignment Operator: Assigns the value of ref to the calling object (th
is pointer)
    // and returns the reference to the calling object
    // You can use Copy constructor as well
    char & operator []( const int index); // Must Do
    // The array operator returns the character at a specific index in the giv
en CString array.
    // You can use content of at functionas well
    void operator +=( const CString& s2 );
    void operator +=( const char * s2 );
    //Adds the value from CString s2 to the array of calling object
    // You can use content of concateqaual function as well
    int operator ==( const CString& s2 ); // Must Do
    // Checks if the passed string is equal to the calling object string or no
t
    //You can use the content from isEqual too
    CString operator + ( const CString& s2 ) const ; // Must Do
    CString operator +( const char * const s2 ) const; // Must Do
    // Concatinates the string passed to the value of the calling object in a
NEW Cstring
    // and returns the object of the NEW CString
    // You can use the content from concat


    //Practice these as soon as you are done learning about them
//    CString tokenzie( const char * const delim );
//    CString operator () ( const char * const delim );
//    istream & operator >> (istream &, const CString &);
//    ostream & operator << (ostream &, const CString &);
//    operator int()
};
```