Objective:

- Usage of constructor/destructor.
- Targets the object transition by reference.
- Character array manipulation.

Task-1: *Array ADT*

The Array ADT that we discussed today.

Private Members:

- `int *data;`
  pointer to an array of integers
- `int capcity;`
  capacity of array pointed by data
- `bool isValidIndex( int index )`
  return true if index is within bounds otherwise false.

Public Members:

The class 'Array' should support the following operations

- `Array( int cap = 0 );`
  Sets 'cap' to 'capacity' and initializes rest of the data members accordingly.
  If user sends any invalid value (-ve value) then sets the cap to zero.
- `~Array()`
  Free the dynamically allocated memory.
- `int & getSet(int index);`
  insert value at given index of array.
- `int getCapacity()`
  returns the size of array.
- `void reSize ( int newcapacity )`
  resize the array to new capacity. Make sure that elements in old array should be preserved in the new array if possible.

Task-2: *CString ADT*

A class which will provide basic functionalities related to strings.

Note: You are not allowed to use any C/C++ library functions related to strings.

```
class CString
{
    char * data;
    int size;
public:
```

`CString ();`	Initializes data and size to 0.
`CString (char c);`	Initializes data with char c
`CString(const char *);`	Initializes the data with received string by allocating memory on heap.
`~CString ();`	You know what to do.
`void input();`	Takes input from console in calling object.
`char & at(int index);`	*Index:* Receives the index for string. *Return Value:* reference of array location represented by index

`bool isEmpty( );`	Tells whether string is empty or not *Return Value*: return true if string empty otherwise true.
`int getLenght();`	Returns length of the string
`void display();`	Prints the string on console
`int find(CString * subStr, int start=0 ) ;`	Find the substring in the calling CString object. By default, search starts from 0 index. Return the count of occurrences found in calling object.
`void insert(int index, CString * subStr);`	Insert the substring at given index in calling object.
`void remove(int index, int count=1);`	Remove the characters (how many? Given in count) starting from index
`int replace(CString * old, CString * newSubStr );`	Find all the occurrences of old substring and replace it with new substring. Return the count of occurrences found in calling object.
`void trimLeft();`	Removes all the white space characters on the left of string
`void trimRight();`	Removes all the white space characters on the right of string
`void trim();`	Removes all the white space characters on both left and right sides of string
`void makeUpper();`	Change all the alphabets to uppercase
`void makeLower();`	Change all the alphabets to lowercase
`void reverse();`	It reverses the string stored in the calling object
`void reSize(int);`	You know what to do.
`int compare(CString & s2 );`	Compare the calling and receive object string. It should behave just like strcmp

};

We shall do many revisions on CString class. That's an initial and amateur version of CString class. You will soon receive an updated version of this class.