# CS 3304 – Data and Information Structures
# Assignment 2

**Due:** 03/06/2023, 23:59

**Submission:** Submit your solutions as a single **.zip** file through Blackboard. **.zip** file will contain

**- Q1.cpp** file (for Question 1) including the **onlyLowerCase** function and the main function.

**- MyList.h** file and **MyListDriver.cpp** file (for Question 2). An example **MyListDriver.cpp** is given in the question.

---

**Q1 (30 pts).** Write a C++ function named **onlyLowerCase** that accepts a string parameter and returns another string by removing all characters other than lower case letters in the parameter. For instance, if the parameter is "The young girl gave no clear response.", the function should return “heyounggirlgavenoclearresponse” . In your main function, read a file named “input.txt” and remove all characters except lower case letters in each line and print them into “output.txt” using **onlyLowerCase** function. An example “input.txt” and “output.txt” is given below:

**input.txt**

```
University of Houston - Downtown
The birch canoe      slid on the smooth planks.
Glue the sheet to the dark      blue background.
It’s easy to     tell the    depth of a well.
These days a chicken leg is a rare dish.
Rice is         often served in round bowls.
```

**output.txt**

```
niversityofoustonowntown
hebirchcanoeslidonthesmoothplanks
luethesheettothedarkbluebackground
tseasytotellthedepthofawell
hesedaysachickenlegisararedish
iceisoftenservedinroundbowls
```

**Q2 (70 pts).** In this question, you will write your own container class named **MyList** similar to vector class in C++. Your class must be a template class which will contain its elements in an array named **myArray**. Since this class will be a template class, you should include the function definitions at the end of the header file. You can also check the Stack example in Chapter 9. You should keep the size and capacity of your list in two separate attributes named **mySize** and **myCapacity**. Therefore, your class will have three data members: one generic dynamic array (you should use pointer representation as given in the Stack example) and two integers. In the list, the first elements will always be the minimum element. As the function members of your class write the following functions:

a. (10 pts) Write a constructor that accepts one integer parameter as the capacity of your list, creates a new array with the given capacity, and stores new array in **myArray**. The constructor should also set **myCapacity** to the parameter value and **mySize** to 0. If the parameter is not a positive number, you should set the capacity as 5 and create the array with 5 elements.
b. (5 pts) Write getters for size (**getSize**) and capacity (**getCapacity**).
c. (15 pts) Write **add** function that accepts an element having generic type, adds this element as the last element of the list, and swaps it with the first element if it is less than the minimum element (first element). For instance, if the list elements are [3, 12, 7, 9] and the parameter of add function is 2, it will be added as the last element and swapped with the first element. The elements will be [2, 12, 7, 9, 3] after addition. If the list elements are [3, 12, 7, 9] and the parameter of add function is 4, it will be added as the last element. The elements will be [3, 12, 7, 9, 4] after addition. You should also update the size (**mySize**) accordingly. The function should return nothing. If the capacity is full, you should
    - double the capacity by creating a new array,
    - copy existing elements into new array,
    - add the element into appropriate place,
    - destroy old array,
    - store the new array in **myArray**,
    - update **myCapacity**.
d. (5 pts) Write a function **getMin** that returns the minimum element in the list. If there is no element in the list, throw an exception.
e. (15 pts) Write **removeMin** function that does not accept any parameter and removes the first element from the list. Note that, the list must have the minimum element as the first element after removal. Therefore, you need to search the minimum element in the list and move it to the first position in the array. You should shift the remaining elements after removal. For instance, if the elements are [3, 14, 9, 12, 5, 7, 11], the list elements will be [5, 14, 9, 12, 7, 11] after calling **removeMin** function. You should also update the size (**mySize**) accordingly. The function should return the removed element (3 in the previous example).

f. (5 pts) Write a function named **display** to print the elements in the list separated by a comma. E.g., [13,11,8,9,12]. Note that you should not print all array content, you should print the elements in the list. For instance, when you create an integer list with capacity 100, **myArray** will contain 100 zeros, but you should not display them. Therefore, you should iterate your loop from 0 to **mySize**-1. The function should return nothing.

g. (5 pts) Write a function named **at** that accepts an integer parameter index and returns the element at the given index. If there is no element at that index, throw an exception.

h. (5 pts) Write a function named **lastIndexOf** that accepts an element having generic type as a parameter and returns the index of last occurrence of it in the list. The function should return -1 if the element is not included.

i. (5 pts) Write a function named **clear** that does not accept any parameter and clears the list by setting size as 0. The function should return nothing.

An example driver class (**MyListDriver.cpp**) and its output are given below.

```
#include <iostream>
#include "MyList.h"
using namespace std;

int main()
{
	MyList<int> a(3);
	MyList<double> b(-10);
	cout << "Size of a: " << a.getSize() << endl;
	cout << "Capacity of a: " << a.getCapacity() << endl;
	cout << "Size of b: " << b.getSize() << endl;
	cout << "Capacity of b: " << b.getCapacity() << endl;
	a.add(9); a.display();
	a.add(2); a.display();
	a.add(4); a.display();
	a.add(3); a.display();
	a.add(5); a.display();
	a.add(1); a.display();
	a.add(4); a.display();
	cout << "a.getMin(): " << a.getMin() << endl;
	cout << "a.at(3): " << a.at(3) << endl;
	cout << "a.removeMin(): " << a.removeMin() << endl;
	a.display();
	a.removeMin(); a.display();
	for (int i = 1; i <= 5; i++) {
		cout << i << ": " << a.lastIndexOf(i) << endl;
	}
	a.clear();
	a.display();
	cout << "Size of a: " << a.getSize() << endl;
	cout << "Capacity of a: " << a.getCapacity() << endl;
}
```

```
Size of a: 0
Capacity of a: 3
Size of b: 0
Capacity of b: 5
[9]
[2,9]
[2,9,4]
[2,9,4,3]
[2,9,4,3,5]
[1,9,4,3,5,2]
[1,9,4,3,5,2,4]
a.getMin(): 1
a.at(3): 4
a.removeMin(): 1
[2,9,4,3,5,4]
[3,9,4,5,4]
1: -1.
2: -1.
3: 0.
4: 4.
5: 3.
[]
Size of a: 0
Capacity of a: 12
```