1801ICT Individual Assignment – Milestone 1
Due end of Week 6 (100 marks)

This assignment requires you to solve problems and write software independently, and to integrate various aspects of C++ programming that you have learned this trimester. You must write ALL the C++ program code. Lecturer and tutors are happy to answer questions on this assignment.

Code and document with comments in your program.

Requirements:

- You are responsible for designing and implementing the class which addresses the requirements of the assignment.
- You should design your algorithms and your overall classes so that your code is efficient and clean.
- You have been supplied with a main program (**Alist.cpp**) which will test your class.
- **The version of C++ used for marking will be C++ 14. You must use C++ features where there is a choice between C and C++.**
- **You MUST submit one zip file containing your program source file (Alist.cpp) ONLY. The zip file MUST be named <Your last name>-<snumber>-A1.zip. So student Joe Bloggs, s1234567 would submit the following file:**

  bloggs-s1234567-A1.zip

  Within the zip file, your C++ source file is your updated versions of the supplied file Alist.cpp (do not change the name of this file).

  If you do not submit your assignment in the required format then it will not be marked.

SUBMISSION [TO BE SUBMITTED END OF WEEK 6 VIA L@G]

If you are having issues with doing this assignment:

ASK QUESTIONS, THE SOONER THE BETTER.

Note 1: The ability to ask intelligent questions is an important employability skill which is well worth developing. It shows that you do understand part of the issue and that you are being positive about understanding the remainder of the issue. Simply stating you do not understand something basically shows a negative attitude and that you want someone else to "spoon-feed" you. Much better to phrase it as "I understand this and that but cannot link from there to how this bit works". This shows that you have a positive attitude by at least trying.

Note 2: For most program specifications of more complicated problems, it is impossible to write the specification so that all possible questions are answered. So, invariably, you must have questions about what is required for this assignment.

ASSIGNMENT MILESTONE 1 SPECIFICATIONS

All programs should be submitted online by the due date and time. If you need any help, please ask as we can give you hints and suggestions.

For this task, you cannot use any list, array, vector, stack, queue, dequeue, or any such classes from any existing C/C++ libraries (except for 1 constructor). We are reinventing the wheel here to learn how to make wheels. So we do not want you to use any existing wheels. We hope this is clear, if you are not, please ask the course lecturer/tutor and do not make any assumptions. Please stick to the specification and do not use any advanced things that could be available online or offline from various sources. **Do things yourself from scratch!**

- There is a template *array*< > in C++ that comes with Standard Template Library (STL). This is **not** allowed to be used.

- You can use dynamic memory allocation and deallocation.

  ```
  int * a = new int[200];
  ```

  and then you have to deallocate.

  ```
  delete [ ] a;
  ```

- No linked list is needed in any part of this project.

1. My Customised List

This task is to write a class that implements objects having various states and transitions. The objects considered in this case are lists. You are to implement a custom defined list with a set of specific properties.

- Implement a list class called Alist in C++ to hold integer items. It will have essential methods (default constructor, copy constructor, move constructor, copy assignment, move assignment, and destructor), dynamic memory allocation and deallocation, extend, shrink, insert, remove, sort, unsort, save and search methods. The list will also have other parameterised constructors, setter and getter methods and overloaded operators as required by the supplied main program.

  You have been supplied with a file called Alist.cpp which contains a main program for testing your class. You must add your Alist class specification and Alist method implementations into this file. Your updated Alist.cpp file is the only file you will submit for this question.

- The list will have a capacity at any time, with a default capacity of 8 if not specified initially. However, with an *extend* method, the capacity is increased by default capacity and with a *shrink* method, the capacity is reduced by default capacity if no items are to be discarded during shrinking. There will be a getter method to access the capacity. Using the capacity, memory allocation and deallocation will be performed. Note when memory is allocated using the new capacity, you might need to copy item from the old memory to the new memory.

- The list will store integers internally in an array. The array is allocated dynamically so that the extend and shrink operations can be implemented easily. The list will also have an attribute to show how many items are currently in the list. Implement setter and getter methods to provide read only access to the number of items, and the items at given indexes. Overload the C++ operators as required by the main program.

- There will be a Boolean attribute internally to indicate whether the list is sorted in non-decreasing order or not. The value of the flag could be provided at initialisation and can be modified later by an appropriate setter method. The setter method will also ensure the array is sorted, if not already sorted. You can call any *sort* function from the C++ library for this. To unsort an array swap each item with another randomly selected item. A getter method will let us know the current value of the attribute.

- If we insert an integer to the list, it will add the integer at the end of the array, if the list is not sorted. If the list is sorted, the integer will be inserted in the array such that the array remains sorted in ascending order. You cannot call any sort function here. Start from the end of the list and move items one position towards the end until you find the suitable position for the integer, and then put the integer there. The insert operation may have to increase the capacity by the default capacity of the list to add the integer.

- If we search for an integer, then we get the index if the integer is found, else we get a value equal to the capacity of the list. The search in the array will however depend on whether the list is sorted. If the array is not sorted then check each integer in the array. If the array is sorted then a binary search will run.

- If an integer is to be removed from the list, then first search to check whether the integer exist in the list. If it exists, then the integer is to be removed and true will be returned. If the integer does not exist in the list then false will be returned. When an integer is removed from the array and the array is not sorted, then bring the integer at the end to fill in the gap. If the list is sorted and you are to remove an integer then move all the trailing integers by one position to fill in the gap so that the array still remains sorted. You cannot call any sort function here. If the number of elements in the list is less than (capacity – default capacity) then the list capacity should be reduced by default capacity.

- The output from the sample solution is :

```
Sizes: 104 104 144
Counts: 100 100 100
Double remove: 1 0
Search present, not present: 50 -1
Search (should be 20 57): 20 57
Extend (should be 112): 112
Shrink (should be 104): 104
Remove (should be 8): 8
Resize (should be 50 50): 50 50
Invalid index -20
Invalid index 200
0 2 4 6 8 10 12 14 16 18
0 2 4 6 10 12 14 16 18
```

Note 1 : As a guide, the class in the sample solution contains: 7 constructors, 10 operator overloads, 1 destructor, 4 private functions to eliminate duplicate code and 14 public functions.

Note 2 : It is the case that, given your 1806ICT background, you can start this assignment in Week 1.

Marking Scheme

Requirement	Mark
Essential methods (constructors, destructor)	10
Insertion operation	7
Remove operation	7
Search operation	6
Other methods as required by the main program, including operator overloading	40
Coding standard (e.g. use of functions, appropriate variable names, program layout, use of comments)	10
Program efficient and correct (e.g. meets all requirements, no duplicated code)	20