2nd Sem. 2017-18
CS F211 Data Structures & Algorithms
Lab 1 (Week of 7th Jan. - 12th Jan.)
==========

Topic: Interface vs. Implementation

Learning Outcomes: The student should be able to (i) implement a given interface and (ii) use the interface without being dependent on the implementation. The student should be able to enable modular access while providing efficient implementation.

Implementation Platform: All implementation must be done in C on Linux using *gcc* to compile code.

Exercise 1: Define the data type **SeqList** (*to denote a Sequential Access List i.e. a list where elements are accessible only one after the other in a sequence*) with the following interface:

- SeqList newList() // creates an empty list
- SeqList insertInOrder(SeqList sl, Element e) // add e to sl in order of key assuming sl is ordered; return the modified list
- SeqList insertAtFront(SeqList sl, Element e) // add e at the front of the list; return the modified list
- SeqList insertAtEnd(SeqList sl, Element e) // add e at the end of the list; return the modified list
- SeqList delete(SeqList sl, Element e) // delete e from sl; return the modified list
- SeqList deleteAtFront(SeqList sl) // delete the first element from sl; return the modified list
- Element find(SeqList sl, Key k) // find the element e with key k in sl; return e

Use the header file SeqList.h for the interface.
[Hint: Efficient Addition at front and rear – will require two pointers – to the first and the last nodes. End of Hint.]

Exercise 2: Implement the interface (i.e. the operations listed above) in file SeqList.c using a linked list as the representation. Compile SeqList.c into a separate module.

Exercise 3 : (a) Write a procedure *merge* that takes two ordered sequential lists and merges them into a single ordered sequential list.
(b) Write a procedure *insertionSort* that takes an unordered sequential list and sorts the elements according to a key.
Note that both these procedures must use only the interface given above i.e. should not touch the underlying implementation. This also implies that no allocation / de-allocation is to be done in these procedures.
Implement these procedures in the file SeqListUses.c and compile this into a separate module.

Exercise 4: Write a main procedure to test the procedures insertionSort and merge. Implement this procedure in file SeqListTest.c. Compile this along with the other two modules to build an executable.