300103 Data Structures and Algorithms

Practical 2
(For Week 3)

Note: This practical will not be marked. However, attendance at Week 3's practical is compulsory because your tutor will explain the solution to Tutorial 1 and continue to mark Practical 1 if it has not been completed. Feel free to show us your solution to this practical. It is part of the training to Assignment 1.

Task 2.1
Given an array of *n* integers, write two programs to process a list of integers with different time complexity: one is in *O(n)* and the other in *O(log n)* or $O(n^2)$. The processes can be any. Test your programs using the following data:

1, 3, 8, 9, 12, 14, 22, 25, 33, 34, 38, 59, 61, 66, 68, 73, 75, 99, 101, 203, 454

Hint: *You may have one function for linear search and the other one for binary search or sorting*.

Task 2.2
Use the following class structure to create a class template with a number of member functions that implement the following functionalities:

1. Search for a particular element using linear search.
2. Sort the data in ascending order.
3. Search for a particular element after the data is sorted using binary search.

You may implement the algorithm by changing the code provided for Lecture 2 (download *CodeforLecture2.zip* on vUWS under Lecture 2). Test your program using two different data types, say *integer* and *character*.

```
template <class Type>
class Aggregate {
private:
      vector<Type> list;
public:
      Aggregate(Type* input, int length) {
            for (int i=0;i<length; i++)
                  list.push_back(input[i]);
      }

      bool seqSearch(Type item, int& loc);
      void bubbleSort();
      bool binarySearch(Type item, int& loc);
```

```
    void print();
};
```

Task 2.3
Rewrite your program for **Task** 1.4 so that it can generate 1000 lists of the numbers instead of 10 lists (store these lists in a vector as required in **Task** 1.4) and checks if any list in the vector is exactly the same as the following:

```
2 1 5 4 0 0 6 0 3
```

i.e, the number at each position is the same (irrelevant to the number of function calls to *rand()*).

Hint: *If each list in the vector is an object, you simply perform a sequential search over the vector.*

Task 2.4
Rewrite your program for **Task** 1.4 so that it prints a list in the format of 3*3 matrix. For instance, the list `2 1 5 4 0 0 6 0 3` is printed as:

```
 --- --- ---
| 2 | 4 | 6 |
| 1 | 0 | 0 |
| 5 | 0 | 3 |
 --- --- ---
```