COMP2080 - Data Structures and Algorithms

Tutorial/Lab 1 and 2

Learning Objectives:

1. Review basic Object Orient Programming
2. Review of basic array operations and application.
3. Discuss, analyse and apply different array based methods for storing and accessing data
4. Implement the Selection sort algorithm and analyze its performance
5. Implement the Insertion sort algorithm and analyze its performance
6. Implement the Linear Search algorithm and analyze its performance
7. Implement the Binary Search algorithm and analyze its performance

Review of Objects, Arrays and selection sort- Part 1

The main objective is to implement the class given below. We will discuss and analyze the implementation of each function as they are created.

The UnOrderedArray class:

UnOrderedArray
- m_array : int[] - maxSize : int - numElements : int
+ UnOrderedArray (int size) + addLast (int item): boolean + removeItem () : boolean + efficientRemoveItem(int index): boolean + linearSeach(int item): int + selectionSortAsc(): void + listItems (): void

State information:

m_array - An integer array.

maxSize – An integer recording the maximum size of the array.

numElements - An integer representing the number of items in the array.

State information:

Function signature	Description
UnOrderedArray (int **size**)	Sets the ***maxSize*** to **size** Creates and array of size ***maxSize*** called ***m_array*** Sets the ***numElements*** to 0
boolean addLast (int **item**)	Adds **item** at the end of the array if there is space and returns ***true***. Returns ***false*** otherwise.
boolean removeItem(int **item**)	Removes the first occurrence of **item** from the array. If found, the item must be removed by shifting down all items after it down by 1. The function returns ***true*** if the removal was successful (if found) and ***false*** otherwise (not found)
boolean efficientRemoveItem(int **item**)	Removes the first occurrence of **item** from the array. **The strategy to be discussed in the tutorial.**
int linearSearch(int **item**)	Returns the location of the first occurrence of item in the array of found and -1 if not found.
void listItems()	Prints a list of all of the items in the array
void selectionSortAsc()	Sorts the array in ascending order using selection sort.

Implementation and analysis of insertion sort and binary search- Part 2

Function signature	Description
int BinarySearch(int **item**)	Returns the location of the first occurrence of item in the array of found and -1 if not found.
void insertionSortAsc()	Sorts the array in ascending order using insertion sort.
void bubbleSort()	Sorts the array in ascending order using bubble sort.

Modify and adapt the techniques learnt to develop a solution to the scenario outlined below:

A company is having an event. Each employee is assigned an identification number (an integer). The list of ***n*** employees participating in this event is stored in an array called ***eventEmployeeList*** in ascending order. The IT department manager would like to know how many of the employees in his department are participating in this event. The identification numbers of the ***m*** employees in the IT department are stored in another array called ***itDepartment*** in no particular order.

Write a method called **isParticipating** which accepts ***eventEmployeeList***, ***n*** and an identification number, and returns a true value if the identification number is found in the ***eventEmployeeList*** array or false value if it is not found. Your method must use an efficient search technique.

Write a method called **numParticipating** which accepts ***eventEmployeeList*** , ***n***, ***m*** and ***itDepartment*** and returns the number of employees that are participating. You must use the method **isParticipating**.