CSE225L – Data Structures and Algorithms Lab
Lab 05
Sorted List (array based)

In today's lab we will design and implement the List ADT where the items in the list are sorted.

sortedtype.h

```
#ifndef SORTEDTYPE_H_INCLUDED
#define SORTEDTYPE_H_INCLUDED

const int MAX_ITEMS = 5;
template <class ItemType>
class SortedType
{
    public :
        SortedType();
        void MakeEmpty();
        bool IsFull();
        int LengthIs();
        void InsertItem(ItemType);
        void DeleteItem(ItemType);
        void RetrieveItem(ItemType&,
bool&);
        void ResetList();
        void GetNextItem(ItemType&);
    private:
        int length;
        ItemType info[MAX_ITEMS];
        int currentPos;
};
#endif // SORTEDTYPE_H_INCLUDED
```

sortedtype.cpp

```
#include "sortedtype.h"
template <class ItemType>
SortedType<ItemType>::SortedType()
{
    length = 0;
    currentPos = - 1;
}
template <class ItemType>
void SortedType<ItemType>::MakeEmpty()
{
    length = 0;
}
template <class ItemType>
bool SortedType<ItemType>::IsFull()
{
    return (length == MAX_ITEMS);
}
template <class ItemType>
int SortedType<ItemType>::LengthIs()
{
    return length;
}
template <class ItemType>
void SortedType<ItemType>::ResetList()
{
    currentPos = - 1;
}
template <class ItemType>
void
SortedType<ItemType>::GetNextItem(ItemType&
item)
{
    currentPos++;
    item = info [currentPos];
}
template <class ItemType>
void SortedType<ItemType>::InsertItem(ItemType
item)
{
    int location = 0;
    bool moreToSearch = (location < length);

    while (moreToSearch)
    {
        if(item > info[location])
        {
            location++;
            moreToSearch = (location < length);
        }
        else if(item < info[location])
            moreToSearch = false;
    }
    for (int index = length; index > location;
index--)
        info[index] = info[index - 1];
    info[location] = item;
    length++;
}
template <class ItemType>
void SortedType<ItemType>::DeleteItem(ItemType
item)
{
    int location = 0;

    while (item != info[location])
        location++;
    for (int index = location + 1; index < length;
index++)
        info[index - 1] = info[index];
    length--;
}
template <class ItemType>
void SortedType<ItemType>::RetrieveItem(ItemType&
item, bool& found)
{
    int midPoint, first = 0, last = length - 1;
    bool moreToSearch = (first <= last);
    found = false;
    while (moreToSearch && !found)
    {
        midPoint = (first + last) / 2;
        if(item < info[midPoint])
        {
            last = midPoint - 1;
            moreToSearch = (first <= last);
        }
        else if(item > info[midPoint])
        {
            first = midPoint + 1;
            moreToSearch = (first <= last);
        }
        else
        {
            found = true;
            item = info[midPoint];
        }
    }
}
```

Generate the **Driver file (main.cpp)** and perform the following tasks:

Operation to Be Tested and Description of Action	Input Values	Expected Output
• Create a list of size 5		
• Print length of the list		0
• Insert five items	5 7 4 2 1	
• Print the list		1 2 4 5 7
• Retrieve 6 and print whether found		Item is not found
• Retrieve 5 and print whether found		Item is found
• Print if the list is full or not		List is full
• Delete 1		
• Print the list		2 4 5 7
• Print if the list is full or not		List is not full