CSE225L – Data Structures and Algorithms Lab
Lab 04
Unsorted List (array based)

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

unsortedtype.h

```
#ifndef UNSORTEDTYPE_H_INCLUDED
#define UNSORTEDTYPE_H_INCLUDED

const int MAX_ITEMS = 5;

template <class ItemType>
class UnsortedType
{
    public :
        UnsortedType();
        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 // UNSORTEDTYPE_H_INCLUDED
```

unsortedtype.cpp

```
#include "UnsortedType.h"

template <class ItemType>
UnsortedType<ItemType>::UnsortedType()
{
    length = 0;
    currentPos = -1;
}
template <class ItemType>
void UnsortedType<ItemType>::MakeEmpty()
{
       length = 0;
}
template <class ItemType>
bool UnsortedType<ItemType>::IsFull()
{
    return (length == MAX_ITEMS);
}
template <class ItemType>
int UnsortedType<ItemType>::LengthIs()
{
    return length;
}
template <class ItemType>
void UnsortedType<ItemType>::ResetList()
{
    currentPos = -1;
}
template <class ItemType>
void
UnsortedType<ItemType>::GetNextItem(ItemType&
item)
{
    currentPos++;
    item = info [currentPos] ;
}
template <class ItemType>
void
UnsortedType<ItemType>::RetrieveItem(ItemType&
item, bool &found)
{
    int location = 0;
    bool moreToSearch = (location < length);
    found = false;
    while (moreToSearch && !found)
    {
        if(item == info[location])
        {
            found = true;
            item = info[location];
        }
        else
        {
            location++;
            moreToSearch = (location < length);
        }
    }
}
template <class ItemType>
void UnsortedType<ItemType>::InsertItem(ItemType
item)
{
    info[length] = item;
    length++;
}
template <class ItemType>
void UnsortedType<ItemType>::DeleteItem(ItemType
item)
{
    int location = 0;
    while (item != info[location])
        location++;
    info[location] = info[length - 1];
    length--;
}
```

Now generate the **Driver file (main.cpp)** where you perform the following tasks:

Operation to Be Tested and Description of Action	Input Values	Expected Output
• Create a list of size 5		
• Insert four items	5, 7, 6, 9	
• Print the list		5 7 6 9
• Print the length of the list		4
• Insert one item	1	
• Print the list		5 7 6 9 1
• Retrieve 4 and print whether found or not		Item is not found
• Retrieve 5 and print whether found or not		Item is found
• Retrieve 9 and print whether found or not		Item is found
• Retrieve 10 and print whether found or not		Item is not found
• Print if the list is full or not		List is full
• Delete 5		
• Print if the list is full or not		List is not full
• Delete 1		
• Print the list		7 6 9
• Delete 6		
• Print the list		7 9