North South University
CSE-225L Fall-2017
Lab 04: Unsorted List (Array Based)

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 data[MAX_ITEMS];
        int currentPosition;

};

#endif
```

unsortedtype.cpp

```
#include "unsortedtype.h"

template <class ItemType>
UnsortedType<ItemType>::UnsortedType()
{
        length = 0;
        currentPosition = -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>::insertItem(ItemType
item)
{
        data[length] = item;
        length++;

}

template <class ItemType>
void UnsortedType<ItemType>::deleteItem(ItemType
item)
{
        int location = 0;

        while(item != data[location])
        {
                location++;
        }

        data[location] = data[length-1];
        length--;

}

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 == data[location])
                {
                  found = true;
                  item = data[location];
                }

                else
                {
                  location++;
                  moreToSearch = (location<length);
                }
        }

}

template <class ItemType>
void UnsortedType<ItemType>::resetList()
{
        currentPosition = -1;

}

template <class ItemType>
```

```
void
UnsortedType<ItemType>::getNextItem(ItemType&
item)
{
        currentPosition++;
        item = data[currentPosition];

}
```

<u>Tasks to be performed</u>:

Now, generate the driver file main.cpp and in that file, perform the following tasks (you cannot change anything in the given source code):

Task Description	Input Values	Expected Output	Allotted Marks
Create a list for integers	-	-	1
Check if the list is empty or not	-	List Empty	1
Insert 4 items in the list	23, -57, 25, 78	-	1
Print all the items in the list using any loop statement	-	23, -57, 25, 78	1
Add another item to the list and print the whole list	96	23, -57, 25, 78, 96	1
Print the length of the list	-	List Length = 5	1
Retrieve 96 and print whether 96 is found or not	-	Item 96 is found	1
Retrieve -69 and print whether -69 is found or not	-	Item -69 not found	1
Delete 25 and print the whole list	-	23,-57,96,78	1
Empty the list and check whether the list is full or not	-	List is not full	1