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 unsorted.

sortedtype.h

```
#ifndef SORTEDTYPE_H
#define SORTEDTYPE_H

const int SIZE = 5;

template <class T>
class SortedType
{
private:
    T *data;
    int currentSize;
    int pointTo;

public:
    SortedType();
    ~SortedType();
    int Length();
    bool IsFull();
    void MakeEmpty();
    void Insert(T value);
    void Delete(T value);
    void Search(T value, bool &found);
    void GetNext(T &value);
    void Reset();
};

#endif // SORTEDTYPE_H
```

unsortedtype.cpp

```
#include "sortedtype.h"

template <class T>
SortedType<T>::SortedType()
{
    data = new T[SIZE];
    currentSize = 0;
    pointTo = -1;
}

template <class T>
SortedType<T>::~SortedType()
{
    delete[] data;
}
```

```
template <class T>
int SortedType<T>::Length()
{
    return currentSize;
}

template <class T>
bool SortedType<T>::IsFull()
{
    return (SIZE == currentSize);
}

template <class T>
void SortedType<T>::MakeEmpty()
{
    currentSize = 0;
}

template <class T>
void SortedType<T>::Insert(T value)
{
    int i = 0;

    while(i < currentSize)
    {
        if (value > data[i])
        {
            i++;
        }
        else
        {
            for (int j = currentSize; j > i; j--)
            {
                data[j] = data[j - 1];
            }
            break;
        }
    }
    data[i] = value;
    currentSize++;
}

template <class T>
void SortedType<T>::Delete(T value)
{
    int i = 0;
    while (value != data[i])
    {
        i++;
    }
    while (i < currentSize)
    {
        data[i] = data[i + 1];
        i++;
    }
    currentSize--;
}
```

```
template <class T>
void SortedType<T>::Search(T value, bool &found)
{
    int midPoint;
    int first = 0;
    int last = currentSize - 1;
    found = false;

    while(first <= last)
    {
        midPoint = (first + last) / 2;

        if(value < data[midPoint])
        {
            last = midPoint - 1;
        }
        else if (value > data[midPoint])
        {
            first = midPoint + 1;
        }
        else
        {
            found = true;
            value = data[midPoint];
            break;
        }
    }
}

template <class T>
void SortedType<T>::GetNext(T &value)
{
    pointTo++;
    value = data[pointTo];
}

template <class T>
void SortedType<T>::Reset()
{
    pointTo = -1;
}
```

Generate the **driver file (main.cpp)** where you perform the following tasks. Note that you cannot make any change to the header file or the source file.

Operation to Be Tested and Description of Action	**Input Values**	**Expected Output**
Create a list of integers		
Print length of the list		
Insert five items	5 7 4 2 1	
Print the list		1 2 4 5 7
Search 6 and print whether found or not		Item is not found
Search 5 and print whether found or not		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
Delete 4		
Print the list		2 5 7
Write a class **timeStamp** that represents a time of the day. It must have variables to store the number of <u>*seconds*</u>, <u>*minutes*</u> and <u>*hours*</u> passed. It also must have a function to print all the values. You will also need to overload a few operators.		
Create a list of objects of class **timeStamp**.		

Insert 5 time values in the format ssmmhh	15 34 23
	13 13 02
	43 45 12
	25 36 17
	52 02 20	

Delete the timestamp 25 36 17		
Print the list		13:13:02 43:45:12 52:02:20 15:34:23