CSE225L – Data Structures and Algorithms Lab
Lab 07
Sorted List (Linked-list 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
#define SORTEDTYPE_H

template <class T>
class SortedType
{
private:
    struct Node
    {
        T data;
        Node* next;
    };
    Node* head;
    Node* pointTo;
    int size;
public:
    SortedType();
    ~SortedType();
    int Length();
    void Insert(T value);
    void Search(T value, bool &found);
    void Delete(T value);
    void MakeEmpty();
    void GetNext(T &value);
    void Reset();
};
#endif // SORTEDTYPE_H
```

unsortedtype.cpp

```
#include "sortedtype.h"
#include <iostream>
using namespace std;

template <class T>
SortedType<T>::SortedType()
{
    head = NULL;
    pointTo = NULL;
    size = 0;
}

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

```cpp
template <class T>
void SortedType<T>::Insert(T value)
{
    Node* temp = new Node;       // Create a new node
    temp->data = value;          // Set the data of the new node
    temp->next = NULL;           // Initialize the next pointer

    // Case 1: Empty list
    if (head == NULL)
    {
        head = temp;             // Insert the new node as the head
    }
    else
    {
        // Case 2: Insert at the beginning
        if (value < head->data)
        {
            temp->next = head;  // New node points to the old head
            head = temp;        // New node becomes the new head
        }
        else
        {
            // Case 3: Traverse the list to find the correct position
            Node* i = head;
            Node* prev = NULL;

            while (i != NULL && value > i->data)
            {
                prev = i;       // Move prev to 'i'
                i = i->next;    // Move to the next node
            }
            // Insert between prev and 'i'
            temp->next = i;     // New node points to 'i'
            prev->next = temp;  // Previous node points to new node
        }
    }
    size++;  // Increment the size of the list
}

template <class T>
void SortedType<T>::Search(T value, bool &found)
{
    found = false;

    Node* i = head;

    while(i != NULL)
    {
        if (value == i->data)
        {
            found  = true;
            break;
        }
        else
        {
            i = i->next;
        }
    }
}
```

```
template <class T>
void SortedType<T>::Delete(T value)
{
    Node* i = head;
    Node* prev = NULL;
    bool found = false;

    while(i != NULL)
    {
        if (value == i->data)
        {
            found = true;
            break;
        }
        else
        {
            prev = i;
            i = i->next;
        }
    }

    if (found)
    {
        if (prev == NULL) // first node / no previous nodes
            head = i->next;
        else
            prev->next = i->next;
        delete i;
        size--;
    }
}

template <class T>
void SortedType<T>::MakeEmpty()
{
    Node* i = head;
    Node* nextNode;

    while (i != NULL)
    {
        nextNode = i->next; // Store the next node
        delete i;           // Delete the current node
        i = nextNode;       // Move to the next node
    }

    head = NULL;
    size = 0;
}

template <class T>
SortedType<T>::~SortedType()
{
    MakeEmpty();
}

template <class T>
void SortedType<T>::GetNext(T &value)
{
    if (pointTo == NULL)
    {
```

```
        pointTo = head;
        value = pointTo->data;
    }
    else
    {
        value = pointTo->data;
    }
    pointTo = pointTo->next;
}

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

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
Delete 1		
Print the list		2 4 5 7
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 *seconds*, *minutes* and *hours* 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