LAB 7

CSE225L

Unsorted List (Linked List—Based)

In this lab, we will:

- Design and implement the List ADT, where the items are unsorted.
- Implement the List ADT using a linked list—based structure.
- Create the **UnsortedType** class with methods for insertion, searching, deletion, and resetting the list.
- Test the functionality of the **UnsortedType** class by performing various operations.
- Create two lists using the **UnsortedType** class, merge them, remove duplicates, and sort them.

Notes

UNSORTED LIST (LINKED LIST–BASED)

unsortedtype.h

```
#ifndef UNSORTEDTYPE_H
#define UNSORTEDTYPE_H

template <class T>
class UnsortedType
{
private:
    struct Node
    {
        T data;
        Node* next;
    };
    Node* head;
    Node* pointTo;
    int size;
public:
    UnsortedType();
    ~UnsortedType();
    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 // UNSORTEDTYPE_H
```

unsortedtype.cpp

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

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

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

```cpp
template <class T>
void UnsortedType<T>::Insert(T value)
{
    Node* temp = new Node;
    temp->data = value;
    temp->next = head;
    head = temp;
    size++;
}

template <class T>
void UnsortedType<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 UnsortedType<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>
UnsortedType<T>::~UnsortedType()
{
    MakeEmpty();
}
```

```
template <class T>
void UnsortedType<T>::Delete(T value)
{
    bool found = false;

    Node* i = head;
    Node* prev = NULL;

    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--;
    }
    else
    {
        cout << "Error: Item could not be found in the list" << endl;
    }
}

template <class T>
void UnsortedType<T>::GetNext(T &value)
{
    if (pointTo == NULL)
    {
        pointTo = head;
        value = pointTo->data;
    }
    else
    {
        value = pointTo->data;
    }
    pointTo = pointTo->next;
}

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

UNSORTED LIST (LINKED LIST–BASED)

TASKS:

Instructions:

- Create the driver file (main.cpp) and perform the following tasks.
- You cannot make any changes to the header (**.h**) or source (**.cpp**) files of the **UnsortedType** class.

Operation	Input Values	Expected Output
Create a list of integers		
Insert four items	5 7 6 9	
Print the list		9 6 7 5
Insert one item	1	
Print the list		1 9 6 7 5
Delete 5		
Print the list		1 9 6 7
Delete 1		
Print the list		9 6 7
Delete 6		
Print the list		9 7
Delete 16		Error: Item could not be found in the list

Tasks	Input Values
Create two lists of numbers using the **UnsortedType** class, and then perform the following tasks: 1. Merge the two lists into a single list. *Ensure that the time complexity for combining the sequences is **O(n)***. 2. Sort the list in ascending order. Ensure that there are no duplicate numbers in the list.	First List: 10 1 5 6 10 14 20 25 31 38 40 Second List: 12 2 4 7 9 16 19 23 24 32 35 36 42
	Expected Output
	Merged List (Unsorted): 12 2 4 7 9 16 19 23 24 32 35 36 42 40 38 31 25 20 14 10 6 5 1 10
	Sorted List (Ascending): 1 2 4 5 6 7 9 10 12 14 16 19 20 23 24 25 31 32 35 36 38 40 42

Hint:

- **First List (Linked List)**: In this list, the last inserted element is added first. For example, since 40 is the last element added, it appears at the beginning of the list. The list looks like this:

  40 → 38 → 31 → 25 → 20 → 14 → 10 → 6 → 5 → 1 → 10

- **Second List (Linked List)**: Similarly, in this list, the last inserted element is also added first. Since 42 is the last element added, it appears at the beginning of the list. The list looks like this:

  42 → 36 → 35 → 32 → 24 → 23 → 19 → 16 → 9 → 7 → 4 → 2 → 12

- **First + Second List**: By merging the First and Second Lists, you get a combined linked list that contains elements from both lists. After merging, you can proceed to remove duplicates and sort the list in **O(n)** time.

  12 → 2 → 4 → 7 → 9 → 16 → 19 → 23 → 24 → 32 → 35 → 36 → 42 → 40 → 38 → 31 → 25 → 20 → 14 → 10 → 6 → 5 → 1 → 10