**CSE225L – Data Structures and Algorithms Lab**

**Lab 13**

**Sorted and Unsorted lists using Linked List**

In today’s lab we will design and implement the UnsortedList and SortedList ADTs using linked list.

|  |
| --- |
| **list2.h**  #include "list.h"  template<class T>  class UnsortedList: protected SinglyLinkedList<T>{  public:  UnsortedList(){}  ~UnsortedList(){}  //make the following members of ancestor classes  //publicly accessible which are now protected members  //of UnsortedList class (because of protected inheritance)  using SinglyLinkedList<T>::isEmpty;  using SinglyLinkedList<T>::searchItem;  using SinglyLinkedList<T>::reset;  using SinglyLinkedList<T>::nextItem;  using SinglyLinkedList<T>::hasNext;  //functions specific to UnsortedList class  virtual void insertItem(T value);  virtual void deleteItem(T value);  };  /\* Linked List based implementation of SortedList \*/  template<class T>  class SortedList: public UnsortedList<T>{  public:  SortedList(){}  ~SortedList(){}  //functions specific to SortedList class  virtual void insertItem(T value);  }; |
| **list2.cpp**  #include "list2.h"  template<class T>  void UnsortedList<T>::insertItem(T value)//O(1)  {  this->insertAtStart(value);  }  template<class T>  void UnsortedList<T>::deleteItem(T value)//O(N)  {  if(isEmpty())throw ListEmpty();  //if head->data == value  if(this->head->data==value){  this->deleteStart();  return;  }  node<T> \*pre=NULL, \*cur=(this->head);  while(cur != NULL && cur->data != value)  {  pre = cur;  cur = cur->next;  }  if(cur != NULL){ //if value is found in the list  pre->next = cur->next;  delete cur;  (this->length)--;  }  }  template<class T>  void SortedList<T>::insertItem(T value)//O(N)  {  if(this->isEmpty() || value < (this->head)->data)  {  this->insertAtStart(value);  }  else  {  node<T> \*temp=new node<T>;  node<T> \*pre=NULL, \*cur=(this->head);  while(cur != NULL && (cur->data) < value)  {  pre = cur;  cur = cur->next;  }  temp->data = value;  temp->next = cur;  pre->next = temp;  (this->length)++;  }  } |

**Task1:**

In main function:

\* Instantiate an object of UnsortedList<string> class; name this object *ls*

\* Read n strings (n is another user input) from user and insert those strings into *ls* object

\* Read another string from user and then try to delete it from *ls*

**Task2:**

In main function:

\* Instantiate another object of SortedList<string> class; name this object *lt*

\* Take the strings in *ls* sequentially and insert those strings into *lt* object

[Hint: use reset(), hasNext(), and nextItem() functions to get/access the items in *ls*]

\* print the contents of *lt*

[Hint: use reset(), hasNext(), and nextItem() functions]