**CSE225L – Data Structures and Algorithms Lab**

**Lab 04**

**Unsorted List (array based)**

In today’s lab we will design and implement the List ADT where the items in the list are unsorted.

|  |  |
| --- | --- |
| **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 info[MAX\_ITEMS];  int currentPos;  };  #endif // UNSORTEDTYPE\_H\_INCLUDED  **unsortedtype.cpp**  #include "UnsortedType.h"  template <class ItemType>  UnsortedType<ItemType>::UnsortedType()  {  length = 0;  currentPos = -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>::ResetList()  {  currentPos = -1;  }  template <class ItemType>  void UnsortedType<ItemType>::GetNextItem(ItemType& item)  {  currentPos++;  item = info [currentPos] ;  } | 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 == info[location])  {  found = true;  item = info[location];  }  else  {  location++;  moreToSearch = (location < length);  }  }  }  template <class ItemType>  void UnsortedType<ItemType>::InsertItem(ItemType item)  {  info[length] = item;  length++;  }  template <class ItemType>  void UnsortedType<ItemType>::DeleteItem(ItemType item)  {  int location = 0;  while (item != info[location])  location++;  info[location] = info[length - 1];  length--;  } |

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 |  |  |
| * Insert four items | 5 7 6 9 |  |
| * Print the list |  | 5 7 6 9 |
| * Print the length of the list |  | 4 |
| * Insert one item | 1 |  |
| * Print the list |  | 5 7 6 9 1 |
| * Retrieve 4 and print whether found or not |  | Item is not found |
| * Retrieve 5 and print whether found or not |  | Item is found |
| * Retrieve 9 and print whether found or not |  | Item is found |
| * Retrieve 10 and print whether found or not |  | Item is not found |
| * Print if the list is full or not |  | List is full |
| * Delete 5 |  |  |
| * Print if the list is full or not |  | List is not full |
| * Delete 1 |  |  |
| * Print the list |  | 7 6 9 |
| * Delete 6 |  |  |
| * Print the list |  | 7 9 |
| * Write a class studentInfo that represents a student record. It must have variables to store the student ID, student’s name and student’s CGPA. 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 studentInfo. |  |  |
| * Insert 5 student records | 15234 Jon 2.6  13732 Tyrion 3.9  13569 Sandor 1.2  15467 Ramsey 2 3.1  16285 Arya 3.1 |  |
| * Delete the record with ID 15467 |  |  |
| * Retrieve the record with ID 13569 and print whether found or not along with the entire record |  | Item is found  13569, Sandor, 1.2 |
| * Print the list |  | 15234, Jon, 2..6  13732, Tyrion, 3.9  13569, Sandor, 1.2  16285, Arya, 3.1 |