**North South University - Spring 2023**

Course: CSE225L Assessment: Lab 1 / HW 1 / PS 1

Section: 6 NSU ID: 2211424642 Name: Joy Kumar Ghosh

// Task-1 Codes

|  |
| --- |
| #ifndef SORTEDTYPE\_H\_INCLUDED  #define SORTEDTYPE\_H\_INCLUDED  #include <iostream>  using namespace std;  const int MAX\_ITEMS = 5;  template <class ItemType>  class SortedType{  public :  SortedType();  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 // SORTEDTYPE\_H\_INCLUDED |
| #include "sortedtype.h"  template <class ItemType>  SortedType<ItemType>::SortedType() {  length = 0;  currentPos = - 1;  }  template <class ItemType>  void SortedType<ItemType>::MakeEmpty() {  length = 0;  }  template <class ItemType>  bool SortedType<ItemType>::IsFull() {  return (length == MAX\_ITEMS);  }  template <class ItemType>  int SortedType<ItemType>::LengthIs() {  return length;  }  template <class ItemType>  void SortedType<ItemType>::ResetList() {  currentPos = - 1;  }  template <class ItemType>  void  SortedType<ItemType>::GetNextItem(ItemType &item){  currentPos++;  item = info [currentPos];  }  template <class ItemType>  void SortedType<ItemType>::InsertItem(ItemType item){  int location = 0;  bool moreToSearch = (location < length);  while (moreToSearch) {  if(item > info[location]) {  location++;  moreToSearch = (location < length);  } else if(item < info[location])  moreToSearch = false;  }  for (int index = length; index > location;  index--)  info[index] = info[index - 1];  info[location] = item;  length++;  }  template <class ItemType>  void SortedType<ItemType>::DeleteItem(ItemType item){  int location = 0;  while (item != info[location])  location++;  for (int index = location + 1; index < length;  index++)  info[index - 1] = info[index];  length--;  }  template <class ItemType>  void SortedType<ItemType>::RetrieveItem(ItemType &item, bool &found){  int midPoint, first = 0, last = length - 1;  bool moreToSearch = (first <= last);  found = false;  while (moreToSearch && !found) {  midPoint = (first + last) / 2;  if(item < info[midPoint]) {  last = midPoint - 1;  moreToSearch = (first <= last);  } else if(item > info[midPoint]) {  first = midPoint + 1;  moreToSearch = (first <= last);  } else {  found = true;  item = info[midPoint];  }  }  } |
| #include "sortedtype.cpp"  int main()  {  //declaring integer list  SortedType<int> integerList;  //Helping Variable  int i, item;  bool isFound;  //printing length  cout << integerList.LengthIs() << endl;  //inserting item from user  for(i = 0; i < 5; i++){  cin >> item;  integerList.InsertItem(item);  }  //Printing item  for(i = 0, integerList.ResetList(); i < integerList.LengthIs(); i++){  integerList.GetNextItem(item);  cout << item << " ";  }  cout << endl;  //retrieving 6  item = 6;  integerList.RetrieveItem(item, isFound);  if(isFound)  cout << "Item is found" << endl;  else  cout << "Item is not found" << endl;  //retrieving 5  item = 5;  integerList.RetrieveItem(item, isFound);  if(isFound)  cout << "Item is found" << endl;  else  cout << "Item is not found" << endl;  //printing length full or not  if(integerList.IsFull())  cout << "List is full" << endl;  else  cout << "List is not full" << endl;  //deleting item 1  integerList.DeleteItem(1);  //printing item list  for(i = 0, integerList.ResetList(); i < integerList.LengthIs(); i++){  integerList.GetNextItem(item);  cout << item << " ";  }  cout << endl;  //printing list is full or not  if(integerList.IsFull())  cout << "List is full" << endl;  else  cout << "List is not full" << endl;  return 0;  } |
|  |

// Task-2 Codes

|  |
| --- |
|  |
|  |
|  |

// Task-3 Codes

|  |
| --- |
|  |
|  |
|  |

// Task-4 Codes

|  |
| --- |
|  |
|  |
|  |

// Task-5 Codes

|  |
| --- |
|  |
|  |
|  |