

**North South University**  
**CSE-225L(Data Structures & Algorithm)**  
**Summer - 2018**

**Lab-04 (Unsorted List – Array Based, ItemType data)**

**Class 'ItemType':**

**itemtype.h**

```
#ifndef ITEMTYPE_H_INCLUDED
#define ITEMTYPE_H_INCLUDED
#include <iostream>
#include <string>
#include <stdio.h>
using namespace std;

const int MAX_ITEMS = 6;

enum RelationType {LESS, GREATER, EQUAL};

class ItemType
{
public:
    ItemType();
    RelationType ComparedTo(ItemType);
    void Initialize(int,string);
    int getValue();
    const string getName();
private:
    int value;
    string name;
};
#endif
```

**itemtype.cpp**

```
#include "itemtype.h"

ItemType::ItemType()
{
    value = 0;
}

RelationType ItemType::ComparedTo(ItemType otherItem)
{
    if (value < otherItem.value)
        return LESS;
    else if (value > otherItem.value)
        return GREATER;
    else
        return EQUAL;
}
```

```

void ItemType::Initialize(int v,string n)
{
    value = v;
    name = n;
}

int ItemType::getValue()
{
    return value;
}

const string ItemType::getName()
{
    return name;
}

```

## **Class 'UnsortedType':**

### **unsortedtype.h**

```

#ifndef UNSORTEDTYPE_H_INCLUDED
#define UNSORTEDTYPE_H_INCLUDED
#include "itemtype.h"

class UnsortedType
{
    public :
        UnsortedType();
        void InsertItem(ItemType);
        bool SearchItem(ItemType);
        void DeleteItem(ItemType);
        ItemType GetNextItem();
        int LengthIs();
        bool IsFull();
        bool IsEmpty();
        void ResetList();
        void MakeEmpty();

    private:
        int length;
        ItemType info[MAX_ITEMS];
        int currentPos;
};

#endif // UNSORTEDTYPE_H_INCLUDED

```

## unsortedtype.cpp

```
#include "unsortedtype.h"

UnsortedType::UnsortedType()
{
    length = 0;
    currentPos = -1; // Because, initially the list is empty
}

void UnsortedType::InsertItem(ItemType item)
{
    info[length] = item;
    length++;
}

bool UnsortedType::SearchItem(ItemType item)
{
    bool found = false;

    for(int index = 0; index < length; index++)
    {
        if(info[index].ComparedTo(item) == EQUAL)
        {
            found = true;
            break;
        }
    }

    return found;
}

void UnsortedType::DeleteItem(ItemType item)
{
    if(SearchItem(item) == true)
    {
        int location = 0;

        while (item.ComparedTo(info[location]) != EQUAL)
        {
            location++;
        }
        info[location] = info[length - 1];
        length--;
    }
    else
    {
        cout << "Item not in the list" << endl;
    }
}
```

```

}

ItemType UnsortedType::GetNextItem()
{
    currentPos++;
    return info[currentPos];
}

int UnsortedType::LengthIs()
{
    return length;
}

bool UnsortedType::IsFull()
{
    return (length==MAX_ITEMS);
}

bool UnsortedType::IsEmpty()
{
    return (length==0);
}

void UnsortedType::ResetList()
{
    currentPos = -1;
}

void UnsortedType::MakeEmpty()
{
    length = 0;
}

```

### **main.cpp**

```

#include "itemtype.h"
#include "unsortedtype.h"

int main()
{

    UnsortedType u1,u2;
    ItemType t1,t2;

    int num1,num2;
    string name1,name2;

    cout<<"\nName: ";

```

```
getline(cin,name1);
cout<<"\nAmount: ";
cin>>num1;
```

```
getchar();
```

```
/*
```

You have to waste the newline in the buffer which comes from executing the line `cin>>num1`. Because, as 'num1' is int type, any input after entering the integer value for num1 (including the '\n' generated due to pressing of the ENTER key on the keyboard) will remain in the input buffer. As the next reading of user input in the `getline(cin,name2)` is expecting any kind of string, it'll consider the '\n' as input, instead of taking actual input from the user.

```
*/
```

```
cout<<"\nName: ";
getline(cin,name2);
cout<<"\nAmount: ";
cin>>num2;
```

```
//Initializing the ItemType objects with the data read from
//console
```

```
t1.Initialize(num1,name1);
t2.Initialize(num2,name2);
```

```
// Inserting the ItemType objects into the list objects
u1.InsertItem(t1);
u2.InsertItem(t2);
```

```
// Utilize the hints above for writing the codes of your solution.
```

```
return 0;
}
```