main.cpp

```
#include "unsortedtype.h"
int main()
{
   UnsortedType u;
   /*
    Your Code Here to Solve the Given Problem
   */
    return 0;
}
```

unsortedtype.h

```
#ifndef UNSORTEDTYPE_H_INCLUDED
#define UNSORTEDTYPE_H_INCLUDED
#include <iostream>
using namespace std;
const int MAX_ITEMS = 5;

class UnsortedType
{
    public :
        UnsortedType();
        void InsertItem(int);
        bool SearchItem(int);
        void DeleteItem(int);
        void GetNextItem(int&);
        int LengthIs();
        bool IsFull();
        bool IsEmpty();
```

```
        void ResetList();
        void MakeEmpty();

    private:
        int length;
        int info[MAX_ITEMS];
        int currentPos;
};
#endif // UNSORTEDTYPE_H_INCLUDED
```

unsortedtype.cpp

```
#include "unsortedtype.h"

UnsortedType::UnsortedType()
{
  length = 0;
  currentPos = -1;
}
void UnsortedType::InsertItem(int item)
{
    info[length] = item;
    length++;
}

bool UnsortedType::SearchItem(int item)
{

    bool found = false;
```

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

    return found;
}
void UnsortedType::DeleteItem(int item)
{

    if(SearchItem(item)==true)
    {
        int location = 0;

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

```
void UnsortedType::GetNextItem(int& item)
{
    currentPos++;
    item = 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;
}
```