```
#include <iostream>
#include <string>
using namespace std;

struct Node
{
   int value;
   Node* next;
};

class LinkedList
{
public:
   LinkedList();
   void printItems();
   void addToFront(int v);
   void addToRear(int v);
   void addInOrder(int v);
   void deleteItem(int v);
   bool findItem(int v);
   ~LinkedList();

private:
   Node* head;

};

LinkedList::LinkedList()
{
   head = nullptr;
}

void LinkedList::printItems()
{
   Node* p = head;

   while(p!=nullptr)
   {
      cout<< p->value << endl;
      p=p->next;
   }
}

void LinkedList::addToFront(int v)
{
   Node* p = head;
   Node* q = new Node;
   q->value=v;
   q->next=p;
   head=q;
}

void LinkedList::addToRear(int v)
{
   Node* p = head;
   Node* q = new Node;

   q->value=v;

   if(p==nullptr)
   {
      addToFront(v);
      return;
   }

   while(p->next!=nullptr)
   {
      p=p->next;
   }

   p->next=q;
   p->next->next=nullptr;
   return;
}

void LinkedList::addInOrder(int v)
{
   Node* p = head;
   Node* q = new Node;
   Node* temp;

   q->value=v;

   if(p==nullptr)
   {
      addToFront(v);
      return;
   }

   while(p->next!=nullptr && p->next->value<=v)
   {
      p=p->next;
   }

   temp=p->next;
   p->next=q;
   p->next->next=temp;
   return;
}

void LinkedList::deleteItem(int v)
{
   Node* p = head;
   Node* q = head;

   if(p==nullptr)
   {
      return;
   }

   while(p->next!=nullptr && p->next->value!=v)
   {
      p=p->next;
   }

   if(p->next==nullptr)
      return;

   q=p->next->next;
   cout << "Destroyed Node with value :: " << p->next->value<<endl;
   delete p->next;
   p->next=q;

   return;
}

bool LinkedList::findItem(int v)
{
   Node* p = head;

   if(p==nullptr)
   {
      return false;
   }

   while(p->next!=nullptr && p->next->value!=v)
   {
      p=p->next;
   }

   if(p->next==nullptr)
      return false;

   return true;
}

LinkedList::~LinkedList()
{
   Node* p = head;

   while(p!=nullptr)
   {
      Node* temp=p->next;
      cout << "Destroyed Node with value :: " << p->value<<endl;
      delete p;
      p=temp;
   }

}

int main()
{
   LinkedList test;

   /*
   test.addToFront(1);
   test.addToFront(2);
   test.addToFront(3);
   test.addToFront(4);
   test.addToFront(5);
   test.printItems();
   */

   /*
   test.addToRear(1);
   test.addToRear(2);
   test.addToRear(3);
   test.addToRear(4);
   test.addToRear(5);
   test.printItems();
   */

   /*
   test.addToRear(1);
   test.addToRear(2);
   test.addToRear(3);
   test.addToRear(4);
   test.addToRear(5);
   test.printItems();

cout<<"========================"<<endl;

   test.addInOrder(15);
   test.printItems();
   */
   test.addToFront(1);
   test.addToFront(2);
   test.addToFront(3);
   test.addToFront(4);
   test.addToFront(5);

   test.printItems();

   if(test.findItem(3))
      cout<< "Found 3"<<endl;
   else
      cout<< "Did not find 3"<<endl;

   test.deleteItem(3);

   if(test.findItem(3))
      cout<< "Found 3"<<endl;
   else
      cout<< "Did not find 3"<<endl;
}
```

```
#include <iostream>
#include <string>
using namespace std;

class Car
{
public:
   Car(int numModels);
   Car(const Car &other);
   //Copy
   Car& operator= (const Car& src);
   //Assignment Operator
   ~Car();

   void addModel(string model);
   void printModels();


private:
   int m_numModels;
   string modelType;
   string* contents;
};

Car::Car(int numModels)
:m_numModels(numModels)
{

   contents = new string[m_numModels];
}

Car::Car(const Car &other)
{
   m_numModels=other.m_numModels;
   contents=new
string[other.m_numModels];
   for (int i=0; i<other.m_numModels; i++)
   {
      contents[i]=other.contents[i];
   }
}

Car& Car::operator=(const Car &src)
{
   if (&src == this)
      return(*this); // do nothing

   m_numModels=src.m_numModels;
   delete [] contents;
   contents=new string[src.m_numModels];
   for (int i=0; i<src.m_numModels; i++)
   {
      contents[i]=src.contents[i];
   }
   return*this;
}

Car::~Car()
{
   delete [] contents;
}

void Car::addModel(string model)
{
   for(int i=0; i<m_numModels; i++)
   {
      if(contents[i]=="")
      {
         contents[i]=model;
         return;
      }

   }
}

void Car::printModels()
{
   for(int i=0; i<m_numModels; i++)
   {
         cout<<contents[i]<<endl;
   }
}

int main()
{
   string modelType;
   Car porsche(5);

   cout<<"Enter model type:: ";
   cin>>modelType;
   porsche.addModel(modelType);

   cout<<"Enter model type:: ";
   cin>>modelType;
   porsche.addModel(modelType);

   porsche.printModels();
   //Assignment
   Car toyota(5);
   Car honda(10);
   toyota=honda;

   //Copy Constructor
   Car ferrari(111);
   Car lamborghini(333);
   Car maserati=ferrari;
   Car jaguar(lamborghini);

   return 0;
}
```