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

struct nodeType
{
        int info;
        nodeType *link;
};
```

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


struct nodeType
{
        int info;
        nodeType *link;
};


class linkedListType
{
public:
     void initializeList();
     bool isEmptyList() const;
     void print() const;
     int length() const;
     void destroyList();
     int front() const;
     int back() const;
     void insertFirst(const int& newItem) ;
     void deleteNode(const int& deleteItem);
     linkedListType();
     ~linkedListType();

private:
   int count;
   nodeType *first;
   nodeType *last;

};
```

```
bool linkedListType::isEmptyList() const
{
   return(first == NULL);
}

linkedListType::linkedListType()
{
   first = NULL;
   last = NULL;
   count = 0;
}

void linkedListType::destroyList()
{
   nodeType *temp;

   while (first != NULL)
   {
      temp = first;
      first = first->link;
      delete temp;
   }
   last = NULL;

   count = 0;
}
```

```
void linkedListType::initializeList()
{
	destroyList();
}

//complete this function definition
void linkedListType::insertFirst(const int& newItem)
{
	nodeType *newNode; //_____________________________
	newNode = new nodeType; //_______________
	assert(newNode != NULL);//_____________

	newNode->info = newItem;	//store the new item in the node
	newNode->link = first;	//insert newNode before first
	first = newNode;	//make first point to the
		//actual first node
	count++;			//increment count

	if(last == NULL)
		last = newNode;
}

void linkedListType::deleteNode(const int& deleteItem)
{
	nodeType *current; //_____________________
	nodeType *trailCurrent; //_____________________
	bool found;

	if(first == NULL)
		cerr<<"Can not delete from an empty list.\n";
	else
	{
		if(first->info == deleteItem)
		{
			current = first;
			first = first->link;
			count--;
			if(first == NULL)
				last = NULL;
			delete current;
		}
		else
		{
			found = false;
			trailCurrent = first;

			current = first->link;
```

```
                        while(current != NULL && !found)
                        {
                                if(current->info != deleteItem)
                                {
                                        trailCurrent = current;
                                        current = current->link;
                                }
                                else
                                        found = true;
                        }

                        if(found)
                        {
                                trailCurrent->link = current->link;
                                count--;

                                if(last == current)
                                        last = trailCurrent;

                                delete current;
                        }
                        else
                                cout<<"Item to be deleted is not in the list."<<endl;
                } //end else
        } //end else
} //end deleteNode
```

```
void linkedListType::print() const
{
   nodeType *current;

   current = first;
   while (current != NULL)
   {
      cout << current->info << " ";
      current = current->link;
   }
}

linkedListType::~linkedListType()
{
  destroyList();
}
```

```
int linkedListType::length() const
{
   return count;
}


int linkedListType::front() const
{
   assert(first != NULL);
   return first->info;
}

int linkedListType::back() const
{
   assert(last != NULL);

   return last->info;
}
```

```
int main()
{
  int start=99;
  int data;
  linkedListType ListObj;
  ListObj.initializeList();
  ListObj.print();

  while (start !=0)
          {
           cout << endl<<endl;
           cout << "Please select an option :.... " << endl<<endl;
           cout << "0. Exit the program." << endl;
           cout << "1. Insert First." << endl;
           cout << "2. Display Front." << endl;
           cout << "3. Display Last." << endl;
           cout << "4. Display All Nodes in the List." << endl;
           cout << "5. Delete Node." << endl;
    cout << "6. Delete All Nodes in the List." << "    ";
    cout << " Choice: ";
           cin >> start;
           switch (start)
            {
             case 1 :   cout << endl;
              cout << "Enter Int number: ";
              cin >> data;
              ListObj.insertFirst(data);
              break;
      case 2 :   cout << "----------------------" << endl;
              cout << "First Data: ";
              cout << ListObj.front();
              break;
             case 3 :   cout << "----------------------" << endl;
              cout << "Last Data: ";
              cout << ListObj.back();
              break;
             case 4 :   cout << "----------------------" << endl;
              ListObj.print();
              break;
      case 5 :   cout << "----------------------" << endl;
              cout << "Enter number to be deleted: ";
              cin >> data;
              ListObj.deleteNode(data);
              break;
      case 6:   ListObj.destroyList();
      default:  cout << "Wrong....Please Reenter: ";
           } //swtich
         } //while

} //main()
```