Chapter 76: Data Structures in C++

Section 76.1: Linked List implementation in C++

Creating a List Node

```
class listNode
{
    public:
    int data;
    listNode *next;
    listNode(int val):data(val),next(NULL){}
};
```

Creating List class

```
class List
{
    public:
    listNode *head;
    List():head(NULL){}
    void insertAtBegin(int val);
    void insertAtEnd(int val);
    void insertAtPos(int val);
    void remove(int val);
    void print();
    ~List();
};
```

Insert a new node at the beginning of the list

```
void List::insertAtBegin(int val)//inserting at front of list
{
    listNode *newnode = new listNode(val);
    newnode->next=this->head;
    this->head=newnode;
}
```

Insert a new node at the end of the list

```
void List::insertAtEnd(int val) //inserting at end of list
{
    if(head==NULL)
    {
        insertAtBegin(val);
        return;
    }
    listNode *newnode = new listNode(val);
    listNode *ptr=this->head;
    while(ptr->next!=NULL)
    {
        ptr=ptr->next;
    }
    ptr->next=newnode;
}
```

Insert at a particular position in list

```
void List::insertAtPos(int pos,int val)
{
    listNode *newnode=new listNode(val);
    if(pos==1)
    {
        //as head
        newnode->next=this->head;
        this->head=newnode;
        return;
    }
    pos--;
    listNode *ptr=this->head;
    while(ptr!=NULL && --pos)
    {
        ptr=ptr->next;
    }
    if(ptr==NULL)
    return;//not enough elements
    newnode->next=ptr->next;
    ptr->next=newnode;
}
```

Removing a node from the list

```
void List::remove(int toBeRemoved)//removing an element
{
    if(this->head==NULL)
    return; //empty
    if(this->head->data==toBeRemoved)
    {
        //first node to be removed
        listNode *temp=this->head;
        this->head=this->head->next;
        delete(temp);
        return;
    }
    listNode *ptr=this->head;
    while(ptr->next!=NULL && ptr->next->data!=toBeRemoved)
    ptr=ptr->next;
    if(ptr->next==NULL)
    return;//not found
    listNode *temp=ptr->next;
    ptr->next=ptr->next->next;
    delete(temp);
}
```

Print the list

```
void List::print()//printing the list
{
    listNode *ptr=this->head;
    while(ptr!=NULL)
    {
        cout<<ptr->data<<" " ;
        ptr=ptr->next;
    }
    cout<<endl;
}
```

Destructor for the list

```
 List::~List()
{
    listNode *ptr=this->head,*next=NULL;
    while(ptr!=NULL)
    {
        next=ptr->next;
        delete(ptr);
        ptr=next;
    }
}
```