```
/*
   File List.cpp

   Class List stores Nodes (class Node) in a linked list.

   This file has the implementation for class List

   Class list is definition for a linked list, having the following operations:

   1. Initialize the list to an empty list.
   2. Free the nodes of a list.
   3. Determine whether a list is empty.
   4. Add a node with a given value into the list following
      the first node with another given value.
   5. Add a node with a given value to the front of the list.
   6. Add a node with a given value to the end of the list.
   7. Delete the first node from the list.

   Eduardo Augusto Bezerra <eduardo.bezerra@ufsc.br>
   Departamento de Engenharia Eletrica

   Data da criacao: Abril de 2006.
   Data da ultima alteracao: 8 de outubro de 2015.

*/

#include "List.h"

List::List() {
   head = 0;
}

List::~List() {
  Node* cursor = head;
  while(head) {
    cursor = cursor->getNext();
    delete head;
    head = cursor;
  }
  head = 0; // Officially empty
}

void List::insertBeforeFirst(int dat) {
  head = new Node(dat, head);
}

void List::insertAfterLast(int dat) {
  Node* p = head;
  Node* q = head;

  if (head == 0)
     head = new Node(dat, head);
  else {
     while (q != 0) {
        p = q;
        q = p->getNext();
     }
     p->setNext(new Node(dat,0));
  }
}

int List::readFirst() {
   return head->getVal();
}

int List::removeFirst() {
  int retval = 0;
  if (head != 0){
```

```
      cout << "Removendo: " << head << endl;
      cout << "e fica:" << head->getVal() << endl;
      retval = head->getVal();
      Node* oldHead = head;
      head = head->getNext();
      delete oldHead;
   }
   return retval;
 }


 void List::insertionSort(int value) {
   Node* p = head;
   Node* q = head;

   if (head == 0)
   {
      head = new Node(value,head);
          int i;
          i = head->getVal();
   }
   else
   {
           int pint;
           int auxint;
           pint = q->getVal();
           auxint = pint;
           while ((q != 0) && (auxint < value))
           {
                   p = q;
                   q = p->getNext();
                   if (q != 0)
                   {
                         pint = q->getVal();
                         auxint = pint;
                   }
           }
           if (p == q)
                   head = new Node(value,head);
           else
                         p->setNext(new Node(value,q));
   }
 }


 int List::removeNode(int dat) {
   Node* p = head;
   Node* q = head;
   int result;

   if (head == 0)
      result = 0;
   else {
           while ((q != 0) && (q->getVal() != dat)){  // Error!! the addresses will always
 be different!
                   p = q;
                   q = p->getNext();
           }
           if (q != 0) {
                   if (q == head){       // it is the first node
                           result = q->getVal();
                           head = q->getNext();
                           delete q;
                   }
                   else{                           // the node is in the middle
                           result = q->getVal();
                           p->setNext(q->getNext());
                           delete q;
                   }
```

```
            }
            else
                    result = 0;           // node not found!
    }

    return result;
  }

  void List::listAll() {
    Node* aux = head;
    while (aux != 0){
       cout << aux->getVal() << endl;
       aux = aux->getNext();
    }
  }
```