Midterm Practice Solutions

TA: Brian Choi

1. (a)

```
SortedLinkedList::SortedLinkedList()
{
    m_head = m_tail = nullptr;
    m_size = 0;
}
```

(b)

```
bool SortedLinkedList::insert(const ItemType& value)
{
    Node* p = m_head;
    Node* q = nullptr;

    while (p != nullptr)           // Find the first element with a greater value
    {                              // than the input value.
        if (value == p->m_value)
            return false;

        if (value < p->m_value)
            break;

        q = p;
        p = p->m_next;
    }

    Node* newNode = new Node;      // The new node must sit between q and p.
    newNode->m_value = value;
    newNode->m_next = p;
    newNode->m_prev = q;

    if (p != nullptr)              // Is there a following node?
        p->m_prev = newNode;
    else
        m_tail = newNode;

    if (q != nullptr)              // Is there a preceding node?
        q->m_next = newNode;
    else
        m_head = newNode;

    m_size++;
}
```


(c)

```
Node* SortedLinkedList::search(const ItemType& value) const
{
    for (Node* p = m_head; p != nullptr; p = p->m_next)
    {
        if (p->m_value == value)
            return p;
    }
        return nullptr;
}
```

(d)

```
void SortedLinkedList::remove(Node* node)
{
    if (node == nullptr)
        return;

    if (node != m_head)
        node->m_prev->m_next = node->m_next;
    else
        m_head = m_head->m_next;

    if (node != m_tail)
        node->m_next->m_prev = node->m_prev;
    else
        m_tail = m_tail->m_prev;

    delete node;
    m_size--;
}
```

(e)

```
void SortedLinkedList::printIncreasingOrder() const
{
    for (Node* p = m_head; p != nullptr; p = p->m_next)
        cout << p->m_value << endl;
}
```


(f)

See the following example.

```
SortedLinkedList linkedList;
linkedList.insert(20);
linkedList.insert(30);
linkedList.insert(40);

Node* p = linkedlist.search(30);
p->m_value = 100;
```

What will the list look like? What will you see if you call `printIncreasingOrder()`?

You can add a protection by making the returned Node pointer constant.

```
const Node* search(const ItemType& value) const;
```

