

ENCM 339 Fall 2005: Handout for T02 (Dr. Norman's tutorial) Tues., Nov. 15

Author: Dr. S. A. Norman. Electronic copies of handouts for L02 and T02 can be found at <http://www.enel.ucalgary.ca/People/Norman/encm339fall2005/>

Exercise 1. Here are definitions for a struct type and a class type ...

```
struct Node {
    int item;
    Node *next;
};

class Nov15List {
public:
    Nov15List(); // PROMISES: Creates empty list.
    Nov15List(const Nov15List& src);
    Nov15List& operator=(const Nov15List& rhs);
    ~Nov15List();
    void insert_first(int item_arg);
    // PROMISES: Node with item == item_arg is inserted at head of list.

    void remove(int item_arg);
    // PROMISES: If item_arg matches at least one item in the list, the node
    // with that item closest to the head is removed. If there is no match,
    // the list is unchanged.

    bool search(int item_arg) const;
    // PROMISES: Returns true if at least one node in the list has an item
    // that matches item_arg, otherwise returns false.
private:
    Node *headM;
};
```

Here are some of the member definitions ...

```
Nov15List::Nov15List() : headM(0) { }

void Nov15List::insert_first(int item_arg)
{
    Node *new_node = new Node;
    new_node->item = item_arg;
    new_node->next = headM;
    headM = new_node;
}

void Nov15List::remove(int item_arg)
{
    // WARNING: Defective function definition.

    if (headM == 0)
        return;
    if (headM->item == item_arg) {
        Node *delete_me = headM;
        headM = headM->next;
        delete delete_me;
        return;
    }
    Node *before = headM;
    while (before != 0 && before->next->item != item_arg)
        before = before->next;
    if (before != 0) {
        Node *delete_me = before->next;
        before->next = before->next->next;

        // point two

        delete delete_me;
    }
}
```

Here is a main function that uses the class ...

```
int main()
{
    Nov15List n;
    n.insert_first(123); n.insert_first(189); n.insert_first(145); n.insert_first(167);

    // point one

    n.remove(189);
    n.remove(123);
    return 0;
}
```

Questions are on the next page ...

Questions for Exercise 1:

- (A) Draw a diagram for point one.
- (B) Write a definition for `Nov15List::search`.
- (C) Draw diagrams for both times the program gets to point two.
- (D) Identify the defect in `Nov15List::remove` and correct it.

Exercise 2. A program with lots of nodes and pointers ...

```
#include <iostream>
struct Node {
    int item;
    Node *foo;
    Node *bar;
};
class Quux {
public:
    Quux();
    ~Quux();
    void f1(int arg);
    void f2(int arg);
    void print() const;
private:
    Node dummyM;
};

Quux::Quux()
{
    dummyM.foo = dummyM.bar = &dummyM;
}

Quux::~Quux() { /* Need code here. */ }

void Quux::f1(int arg)
{
    Node *p = new Node;
    p->item = arg;
    p->foo = dummyM.foo;
    p->bar = &dummyM;
    dummyM.foo->bar = p;
    dummyM.foo = p;
}

void Quux::f2(int arg)
{
    Node *p = new Node;
    p->item = arg;
    p->foo = &dummyM;
    p->bar = dummyM.bar;
    dummyM.bar->foo = p;
    dummyM.bar = p;
}

void Quux::print() const
{
    std::cout << '[';
    for (const Node *p = dummyM.foo; p != &dummyM; p = p->foo)
        std::cout << ' ' << p->item;
    std::cout << " ]\n";
}

int main()
{
    Quux q;
    q.f1(111);

    // point one

    q.f1(222);
    q.f2(333);

    // point two

    q.print();
    return 0;
}
```

Questions for Exercise 2:

- (A) Draw a diagram for point one.
- (B) Draw a diagram for point two.
- (C) Determine the program output.
- (D) Suggest better names for `f1`, `f2`, `foo` and `bar`.