Section Handout 4 Solutions

Problem 1: Saving Memory with the Stack

```
int OurStack::pop() {
    int result = top();
    logicalLength --;

    // See if we can save space
    if (logicalLength <= allocatedLength/4) {
        int newLength = allocatedLength/2;
        newElems = new int[newLength];

        // copy over the old elements
        for (int i=0; i < size(); i++) {
            newElems[i] = elems[i];
        }
        delete[] elems;
        elems = newElems;
    }

    return result;
}
```

Problem 2: The Chat and Cut

```
struct Person {
    string name;
    Person *next;
};


bool chatAndCut(Person *firstInLine, string cutter, Set<string> & friends) {

    Person *cur = firstInLine;

    while (cur != NULL) {
        if (friends.contains(cur->name)) {
            // We found a friend! Create a Person struct for the cutter
            Person *theCutter = new Person;
            theCutter->name = cutter;

            // Re-link up the linked list
            theCutter->next = cur->next;
            cur->next = theCutter;

            return true;
        }
        cur = cur->next;
    }
    return false;
}
```

Problem 3: Reversing a Linked List

```
Node *reverse(Node *root) {
    // Create the root of a new linked list
    Node *newRoot = NULL;

    while (root != NULL) {
        // Hold on to the next element for safe keeping
        Node *next = root->next;

        // re-wire the current root to point toward
        // the front of the new list
        root->next = newRoot;
        newRoot = root;

        // remove the root from the original list
        // and set the root to the next item
        root = next;
    }
    // Return the new root, which was previously the last item
    return newRoot;
}
```