

Section Solution

Discussion Problem 1's Solution: <http://tinyurl.com>

```
static const int kTinyURLBase = 36;
string TinyURLConvertNumber(int num) {
    string code; // default constructed to the empty string
    while (num > 0) {
        int digit = num % kTinyURLBase;
        code = ConvertDigit(digit) + code;
        num /= kTinyURLBase;
    }

    return code;
}

char ConvertDigit(int digit) {
    if (digit < 10) return '0' + digit;
    return 'a' + digit - 10;
}

int TinyURLRecoverNumber(string code) {
    int num = 0;
    for (int i = 0; i < code.size(); i++) {
        num = kTinyURLBase * num + ConvertChar(code[i]);
    }

    return num;
}

int ConvertChar(char ch) {
    if (isdigit(ch)) return ch - '0';
    return ch - 'a' + 10;
}
```

Discussion Problem 2's Solution: Maximizing Game Score

```
int ComputeMaxScore(Grid<int>& board) {
    int prev = max(board[0][0], board[1][0]);
    if (board.numCols() == 1) return prev;
    int curr = max(prev, max(board[0][1], board[1][1]));
    if (board.numCols() == 2) return curr;
    for (int col = 2; col < board.numCols(); col++) {
        int next = max(curr, prev + max(board[0][col], board[1][col]));
        prev = curr;
        curr = next;
    }

    return curr;
}
```

Lab Problem Solution 1: Look And Say

Obviously, the full working program includes some user interactivity that the code I provide below doesn't include, but what is presented here is unquestionably the core of the solution and what's of interest to you:

```

string LookAndSay(string before) {
    string after;
    int len = before.size();
    int pos = 0;
    while (pos < len) {
        char ch = before[pos];
        int count = 1;
        while ((pos + count < len) &&
            (before[pos + count] == ch)) {
            count++;
        }
        after += count + '0';
        after += ch;
        pos += count;
    }

    return after;
}

int main() {
    string current = "1";
    while (true) {
        cout << current << endl;
        Pause(1.0);
        current = LookAndSay(current);
    }
}

```

Lab Problem Solution 2: Scheme Expressions

Kudos to Aubrey Gress for typing up a working solution. Again, the core of what needs to happen is captured by this sample implementation below. There are, of course, multiple approaches to solving the problem. This is just one of them. (There was a discrepancy in return type between section handout lab starter code. Either a **double**-returning or **int**-returning version of **Evaluate** should be fine.)

```

double ApplyOperator(double x1, double x2, char op) {
    switch(op) {
        case '+': return x1 + x2;
        case '-': return x1 - x2;
        case '*': return x1 * x2;
        case '/': return x1 / x2;
        default: Error("Error: Unknown operator");
    };

    // won't get here, but compilers can't always tell that
    return 0;
}

struct state {
    char op;
    bool haveFirstValue;
    double currValue;
};

double Evaluate(string expression) {
    Scanner s;
    Stack<state> stateStack;
}

```

```

s.setInput(expression);
s.setSpaceOption(Scanner::IgnoreSpaces);
s.setNumberOption(Scanner::ScanNumbersAsReals);
s.nextToken();

state currState = { s.nextToken()[0], false };
while (s.hasMoreTokens()) {
    string str = s.nextToken();
    if (str == "(") {
        stateStack.push(currState);
        currState.op = s.nextToken()[0];
        currState.haveFirstValue = false;
    } else if (str == ")") {
        if (stateStack.isEmpty()) break;
        double computedValue = currState.currVal;
        currState = stateStack.pop();
        if (currState.haveFirstValue) {
            currState.currValue = ApplyOperator(currState.currVal,
                                                computedValue,
                                                currState.op);
        } else {
            currState.currValue = computedValue;
            currState.haveFirstValue = true;
        }
    } else {
        double nextValue = StringToReal(str);
        if (currState.haveFirstValue) {
            currState.currValue = ApplyOperator(currState.currVal,
                                                nextValue,
                                                currState.op);
        } else {
            currState.currValue = nextValue;
            currState.haveFirstValue = true;
        }
    }
}

return currState.currVal;
}

```