1 Many people write sloppy code.

Despite dim memories of lectures on coding style, many people—some of them professionals with years of experience—have bad programming habits. Bad habits often develop at the beginner level, when some coding disciplines seem excessive. A function with only five lines of code doesn’t seem to need an eight-line comment block listing the pre- and post-conditions.

- Confident their code is easily understood, programmers do not adequately document their programs.
- In a rush to get their ideas down ‘on phosphor’, they don’t give adequate time to planning out variable and function names.
- Having discovered the need for an extra test, they put a condition on a section of code, but don’t go back to re-format the code so it reflects visually the new structure.
- Sometimes they plan to clean up their code, but almost no one, student or otherwise, ever does. If it’s truly impenetrable, they’re likely to give up even if they do come back to it.

Students have a natural desire to optimize, doing only what must be done. The same thought process which leads them to ask Is this going to be on the test? causes them not to give style much attention, since it’s rarely a heavily weighted aspect of grading.

2 Sloppy code is not just annoying.

It’s bad enough to read (and grade) poorly-written code. But sloppy student code causes more significant problems:

- Programs are assigned to exercise high-level concepts. Getting bogged down in the misunderstandings common to sloppy code makes it difficult to focus on learning the concepts actually being taught.
- If understanding a program is difficult, reusing the code is essentially impossible. However, rewriting everything from scratch means students get little practical experience with maintenance.

3 Why this assignment helps.

This classroom assignment requires students to make changes on a program which exhibits some of the poor coding styles which make maintenance difficult. Being forced to deal with such code gives them a new perspective on the value of good coding style.

4 The ugly program.

The program used for the exercise scans all input characters and prints out the character which appears most frequently. Students are told what the program does and assured that it functions properly.

The source code is printed on the back of this sheet. Machine-readable versions of the program and other relevant files are available for download at: http://elvis.rowan.edu/~nlt/sigcse/ugly/.

5 The assignment.

The students were divided into three groups. Each group was assigned one of these maintenance tasks:

Group 1: Modify the program so that it prints the most frequently occurring letter and the most frequently occurring digit. In both cases, print the character and the number of times it occurs in the input.

Group 2: Modify the program so that it prints the most and least frequently occurring characters in the input. In both cases, print the character and the number of times it occurs in the input. Do not count the characters that occur 0 times.

Group 3: Print every character that occurs 5 or more times in the input and the number of times the character occurs.

Students are told to think about several things:

- What makes this code so hard to understand and modify? Jot down as many specific complaints as you can.
- What should the programmer have done to make the code more readable? What specific changes would you make to the program if you were really going to have to use and maintain this program for a long time?

After all the groups have finished making their code modifications, the class is led through a discussion of these questions. Invariably, a student will note that it would be easier to rewrite the program from scratch than to decipher this one, and asks if that would be okay. The answer is ‘No.’