C Labs, Assignment marking scheme

Each assignment is generally split into 3 "parts" as described in the spec sheet and presentation for the assignment. Each "part" is given a mark out of 100. Thus the maximum mark possible for a complete assignment is 300.

Within each part of an assignment, the breakdown of the marks is: 60 for code functionality, and 40 for code quality. Code functionality is examined by multiple automated tests. **Make sure you follow the spec sheet carefully!** Do not print any extraneous messages which are not required in the spec, and print all required messages in exactly the format specified. Deviations from the spec will break the automated testing, which causes more work for the human marker and slows down the delivery of feedback to the rest of the students. Thus any such deviations from the spec will result in a loss of 10 marks for each test affected. This penalty will only be applied to the first part of the assignment within which it occurs. Also note that in order to automate the testing, stdin and stdout will be redirected to files (as in the first lab) so make sure that any input your program requires works both from keyboard and from file redirection. Where the assignments are provided with example inputs and output files, you should use these to ensure that your program meets all the criteria before submission.

Code quality is further broken down into 5 categories, each worth 8 marks (for that "part" of the assignment). The categories of code quality are:

- **Use of functions** – Make sure that any significant pieces of code which are duplicated throughout your program are written as functions to reduce clutter and ease debugging.
- **Comments** – Your code should be well commented, generally having one comment per function, indicating the purpose of the function and what it's inputs and outputs are. The comment for the "main" function should generally explain what the program does.
- **Formatting** – Following standard formatting conventions such as indentation of loops makes your code easier to read. Don't try to write your entire program on one line.
- **Variable naming** – Using clear and meaningful names for variables helps to make your code self documenting and easier for you or someone else to understand later.
- **Conciseness** – The code should be a reasonable and efficient solution to the problem in question.

Category	Marks
Code functionality	60
Use of functions	8
Comments	8
Formatting	8
Variable naming	8
Conciseness	8

Table 1: Mark breakdown for a single "part" of an assignment