There are five categories considered when marking the source code: formatting, commenting, naming, maintainability (embedded constants), and structure.

## 1 Formatting

- Is the program indented by the same amount for each block?
- Is whitespace used consistently? For example, is there a space around binary operators in some places but not others.
- Are braces used consistently?

1. Abysmal.
2. Minimal care taken.
3. A bit sloppy.
4. A tidy effort with only a few mistakes.
5. Meticulous (apart from 1 or 2 whitespace inconsistencies).

## 2 Commenting

- Is there a banner at the top of the file listing the authors' names and what it does?
- Does each function have a comment explaining its purpose?
- Are the comments well formatted and consistently formatted?
- Are the comments relevant and meaningful?
- Is the program over commented? For example, do most lines have a comment?
- Are there any inappropriate comments? For example, 'add one to i'?

1. No comments.
2. Only a few comments or many inappropriate comments.
3. Good attempt at comments but with poor format.
4. Good, well formatted comments.
5. Excellent, well formatted comments.

## 3 Naming

- Are the variables named consistently?
- Do the variables have meaningful names?
- Are the functions named consistently?
- Do the functions have meaningful names?
- Are the constants named consistently?
- Do the constants have meaningful names?

1. Random or meaningless names.
2. Some variables, functions, and constants have consistent meaningful names.
3. Most, functions, and constants have consistent meaningful names.
4. Almost all variables, functions, and constants have consistent meaningful names.
5. All variables, functions, and constants have consistent meaningful names.

## 4 Embedded Constants

- Does the program use unnamed constants (magic numbers)? This does not include trivial numbers like 1 for incrementing a loop.
- Are dependent constants defined in terms of an independent constant?

1. No use of named constants.
2. Minimal use of named constants.
3. Good use of named constants.
4. Very good use of named constants.
5. Excellent use of named constants. Dependent constants related to independent constants.

TOTAL: _____/25 Oct. 2017

## 5 Structure

- Can you quickly figure out how to use the module?
- Does each module do one thing well, or is it a mishmash of different things?
- Are things (functions, constants) that should be private but are public?

1. No attempt at using a module.
2. An attempt at using a module but of no use to anyone.
3. The module may be useful but is either trivial or hard to use.
4. The module is not trivial and is easy to use.
5. There are multiple modules that are easy to use.