There is no 'best' way to format program text, comments, etc. Nevertheless, this document is meant to give you a guideline of how to format your C++ programs. I consider it more important to format the C++ source consistently than to meticulously follow all the rules mentioned here. At the same time, if you adhere to the rules described below your program will be easy to read and to understand. You and others will have a stress-free way to check for block-structuring errors without having to switch attention back-and-forth between different sections of the program.

The rules are as follows, and we believe that all of our example code has been formatted accordingly:

1. At the beginning of the program 3 sections of comments are mandatory: the Identification Header, the Problem Specification, and the Problem Analysis. See the appendix to the homework assignment 1 (Submitting homework) as an example.
2. A consistent level of indentation is used throughout all code, either 4 spaces or 1 tab stop for each level of indentation.
3. Declarations and statements normally appear one per line. Variables and data members of structures and classes will also appear one per line.
4. Only preprocessor directives and items declared at the outermost (global) level commence in column 1. These will often be namespace declarations, using declarations/directives and function declarations/definitions not contained within classes or namespaces.
5. All declarations and statements that start a nested syntax level, e.g. namespace, class, functions, if, while etc., are written with the symbol that introduces the nested level (often { ) at the end of a line or at the beginning of the next line indented at the same level as the preceding expression.
6. All the declarations and statements within a nested level are indented one level from that of the declaration or statement that introduced that nested level.
7. The exception to (5) is that the keywords case and default appear directly under the switch keyword and the keywords public, private and protected when used to designate the access control of class members appear directly under the keyword struct or class.
8. The exception to (4) and (5) is where the contents of a nested level can conveniently appear on the same line as the declaration or statement introducing that nested level, the entire declaration or statement can appear on a single line. This is typically restricted to the control part of statement, function argument lists, "empty" structures or classes, empty or single statement member functions and simple if statements.
9. Conversely, where a single statement will not fit onto a single line, the "extension" lines will be indented accordingly. In the case of stream I/O operations the cascaded operators shall be aligned vertically.
10. At the end of the nested syntax level mentioned in (4), the symbol that terminates the nested level is placed on a new line, directly underneath the first symbol of the original declaration or statement that introduced the level.
11. The symbol starting the nested level is never omitted, whether this is permitted by the language or not, with the single exception of an if that follows an else.

12. Comments should either be to the right of the statements they explain, or in-line, indented the same amount as the normal statements (to avoid obscuring the indentation).

13. Every struct, class or function shall be preceded by a comment briefly indicating its purpose.

14. Where the method used to program a function (or any other piece of code) is not obvious, explanatory comments are introduced. Comments should never merely translate C++ into equally opaque English, and they should be omitted unless they tell us something that is not obvious. The briefer, while still being coherent, informative, and grammatically correct the better.

15. Operators are surrounded by spaces, except for unary operators and ->, ., [ ], (, and ). Comma and semicolon are followed by a space, but not preceded by one.