Very minimal manual of formatting style

Programming style deals with the appearance of your program. If you were to write an entire program on one line, it would compile and run properly, but doing this would be bad programming style because the program would be hard to read. Documentation consists of explanatory remarks and comments for the program. Programming style and documentation are as important as coding. Good programming style and appropriate documentation reduce the chance of errors and make programs easy to read. The following set of rules is very minimal in comparison with typical requirements imposed by companies that deal with software development and/or software project managers. It must be followed in all homework code.

1. Indentation of code should be carried out in some consistent matter through your program.
2. The program should have brief information that allows for its easy identification encoded at the beginning of file in the form of comment. For example:

```c
/**
 * @course: ECE103-01, Spring 2013
 * @homework: 1.1 (Simple Resistor Network Calculator)
 * @author: John F. Smith
 */
```
3. Naming convention: Use ALL CAPITAL characters in names of constants, otherwise, be consistent through the program. Note that this is very relaxed naming convention and in real life you may be asked to follow more strict rules. Variable names are typically lower case, except arrays, structures and classes. If the name consist of multiple words it can be used like: twoWords or two_words or TwoWords.
4. Spacing – in formulas use space to make them more readable. Please be consistent through your program.
   a. In expressions: do use `c = a++ + ++b;` instead of `c=a+++++b;` (not a typical line, dramatization)
   b. In blocks of code – separate steps of an algorithm with one blank line, separate function implementations with at least one, preferably two blank lines in between them.
5. Blocks of code surrounded by `{}` may be handled in three ways with the first two strongly preferred.
   a. General rules and functions

```c
void main() { // your instructor’s style
    // code goes here
    // indented always the same increment per each nested block
}

void main() { // commonly encouraged
    // code goes here
    // indented always the same increment per each nested block
}

void main() { /* a very short block that fits one line */ } // last resort
```
   b. Nested blocks – indent using the same increment

```c
void main() {
    //
    if (a>10) {
        if (b<20) {
            // ...
        } else {
            // ...
        }
    // ...
}
```
   c. if-else can be compressed like this below

```c
if (a==1) {
    // instructions
} else if (a==2) {
    // instructions
} else {
    // instructions
}
```
   d. for loops – use space after ;

```c
for (int a=1; a<=N; a++) {
    // instructions
}
```