

# COMP 1010 PROGRAMMING STANDARDS

This document lists the programming standards that you must follow for the programming questions in your assignments. You will lose marks on assignments if you do not follow these standards.

## COMMENTING FILES AND FUNCTIONS

1. [Assignments 1-5] Each program file must begin with a comment block similar to the following. Phrases in brackets should be replaced with the designated information. You don't have to follow this example exactly, but all of the information must be there.

```
/*
 * COMP 1010      SECTION [Axx]
 * INSTRUCTOR:    [Name of your instructor]
 * NAME:          [Your name, but NOT your student number]
 * ASSIGNMENT:    [Assignment #]
 * QUESTION:      [question #]
 *
 * PURPOSE:       [What exactly is the program intended to do?]
 */
```

2. [Assignments 1-5] Every function *except* `setup()` and `draw()` must be preceded by a complete and correct comment, giving the following information:

```
/*
 * [What specifically does this function do?]
 * [Where does it get information (mouse, keyboard, parameters, global variables)?]
 * [What does it produce? (Graphics, console output, results returned, globals set?)]
 * [What processing or calculations or decisions or actions are done?]
 */
```

## WRITING READABLE CODE

3. [Assignments 1-5] Use spacing within and between lines of code to separate parts of complex expressions or blocks of code.
4. [Assignments 1-5] Comment non-obvious blocks of code. Describe what a block of code is intended to accomplish, not what each individual line does. Eg:  
`// This section checks to see if we have a valid coordinate.`
5. [Assignments 1-5] Use meaningful but reasonable identifiers, following the naming standards.
  - a. Names of constants use all capitals and underscores, such as `PST_RATE` or `MAXIMUM_VELOCITY` (and should be declared *final*).
  - b. All other identifiers should use initial lower case letters and mixed case, such as `finalGrade` or `calculateTotals`.

Examples:

a	// Bad: too short, not meaningful.
averageMark	// Good
averageOfAllTheMarksInTheList	// Bad: too long and wordy.

6. [Assignments 1-5] A short description should appear in a comment following a variable declaration, including physical units (like kg) where appropriate. Indicate the meaning of initial values where appropriate. For example:

```
int centreX, centreY; // position of the ball's centre in the window
int bounces = 0; // # of times the ball has bounced, initially 0
```

7. [Assignments 2-5] Use indentation to clarify control structures (e.g. loops and **if** constructs).
- Align **else** with the corresponding **if** for readability.
  - Avoid long lines that could be word-wrapped in a text editor; if a statement is too long for one line (more than roughly 80 characters), break the line at an appropriate point, and indent the continuation more than the starting line.
  - Place braces **{ }** in predictable and consistent positions. Any readable and consistent style is acceptable. The essential feature is that all statements that are nested within braces must be indented. Common styles include:

<pre>if-while-else-etc {     stuff-inside ;     stuff-inside ; }</pre>	<pre>if-while-else-etc {     stuff-inside ;     stuff-inside ; }</pre>
<pre>if-while-else-etc {     stuff-inside ;     stuff-inside ; }</pre>	<pre>if-while-else-etc {     stuff-inside ;     stuff-inside ; }</pre>

## WRITING MAINTAINABLE AND EXTENDABLE CODE

When you write code, anticipate that you or someone else (e.g. a marker) will examine it later to see how you wrote it, or modify it. Strive for clarity. Avoid common mistakes that experienced programmers will see right away.

8. [Assignments 1-5] Do not use non-trivial numbers (or other literals) in your code (“magic numbers”). If the number’s value is not immediately obvious, document it. If that number is used only once, place a comment nearby. If it may be used more than once, declare a named constant and define it. That way its value can be changed later without finding every place it is used.

```
sum = 0 ; // literal constant 0 is often OK
count = count + 1 ; // literal constant 1 is OK for increments
width = width - 2 ; // bad unless you explain -2; e.g.,
// -2 since we add a margin on both sides
size = size * 1.08 ; // bad; use a constant like GROW_RATE = 1.08
if (keyPressed == 'F') // bad; declare a constant like GO_RIGHT = 'F'.
```

9. [Assignments 3-5] Never change the value of a **for** loop variable inside the loop. All experienced programmers follow this rule, and if you violate it, you will reveal your lack of experience. Worse, you will increase the chance that modifications to your code will be incorrect.
10. [Assignments 3-5] Use the best possible loop construct:
- If the number of loop iterations is known, or can be calculated, before you enter the loop, use a **for** loop (e.g., when examining the characters in a String, or the positions in an array).
  - If there is no limit on the number of loop iterations (e.g., continue as long as the player stays alive), or the number of iterations cannot be predicted or calculated in advance, use a **while** or **do/while** loop.
  - Use a **do/while** loop if the loop must always be done at least once. Use a **while** otherwise.