CSCI 1300 Style Guide

Consider the following short paragraph in English:

> *Marcela is a cat with brown fur and long fur and white spots on brown fur and enjoys drinking water. Then, Marcela drank water. The water was from the sewer. Any water from the sewer is filthy. The fact that any water from the sewer is filthy is a fact that everyone knows. Then, Marcela hallucinated. Then, Marcela danced a jig. Then, everyone had a nice time.*

That sure was a neat story, wasn't it? But, it reads quite poorly. If we use some rules for styling our sentences in English, we can clean it up to read a bit more nicely:

> *Marcela is a cat with long, brown fur and white spots. Marcela drank some water, which she enjoys. The water, however, was from the sewer, and as everyone knows, any water from the sewer is filthy. As a result, Marcela hallucinated and danced a jig, and everyone had a nice time.*

Similarly, your code also should be well-styled. This will help your code to be easy to understand by other people. The coding style includes white space, indentation and naming variables. In a most professional work environment, you're expected to obey the company's coding style guide to keep consistency within the company. For example, if a painting company was hired to paint the exterior of a building but different employees used different styles, the building would look rather silly. So too a set of codes that a group of programmers collaboratively develop will look silly if each one uses a different style.

Whitespace and indentation

- **Indenting:** increase your indentation by one increment of each brace `{`, and decrease it once on each closing brace `}`. Place a line break after every `{`. Use Tab to increase indent and Shit+Tab to decrease indentation.

```
// bad
int x = 3, y = 7; double z = 4.25;
x++;
if (a == b) { foo(); }
```

```
// good
int x = 3;
int y = 7;
double z = 4.25;

x++;
if (a == b) {
    foo();
}
```

- **Expressions:** Place a space between operators and their operands

```
int x = (a + b) * c / d + foo();
```

- **Blank lines:** use them to separate different blocks of code, and show clearly where each one begins and ends.

```
void foo() {
    ...
}
                              // this blank line here
void bar() {
    ...
}
```

Naming and variables

- **Names:** Give variables descriptive names, such as `firstName` or `homeworkScore`. Avoid one-letter names like `a` or `x`, except for loop counter variables such as `i`.

- **Capitalization:** Name variables and functions with camel-casing `likeThis`, and name classes with Pascal casing `LikeThis`.
- **Avoid global variables:** Never declare a modifiable global variable. The only global named values in your code should be `const` constants. Instead of making a value global, pass it as a parameter and/or return it as needed. Modifiable global variables greatly increase the risk of accidentally changing the value in one routine, but using it incorrectly in another.

```
// bad
int count; // global variable; egads!!

void func1() {
    count = 42;
}

void func2() {
    count++;
}

int main() {
    func1();
    func2();
}
```

```
// better
int func1() {
    return 42;
}

void func2(int& count) {
    count++;
}

int main() {
    int count = func1();
    func2(count);
}
```

Acknowledgment

This style sheet is adapted from [CS106B Style Guide](#).