Pseudocode for “Guess a Number” game

Use this pseudocode as a guide for constructing your flowchart for the “Guess a Number” game. In doing so, there are some important points to keep in mind:

- There’s not necessarily a one-to-one correspondence between lines of pseudocode and flowchart elements, nor between lines of pseudocode and lines of implementation code.
- Indentation is generally used to indicate structure in pseudocode. For example, when a line expressing a conditional or iteration statement is immediately followed by one or more lines that are indented further to the right than that line, those indented lines are most likely those that make up the body of the conditional or iteration statement.
- Pseudocode may not necessarily follow the syntax of any one programming language closely, but generally incorporates elements (e.g. assignment, conditional execution, iterative execution) common to most programming languages. On the other hand, the key words used in pseudocode might not be the same as those in any given programming language, but might be synonymous—or at least similar in meaning.
- In some cases, the pseudocode may include constructs that are *not* available in the target programming language or flowcharting tool. However, these constructs can almost always be replicated by nested or chained constructs in the target language or tool. (In fact, the C/C++/Java/JavaScript/C# `if … else if … else` statement actually consists of 2 or more nested `if … else` statements.)
- While a fundamental aim for articulating the logic of an algorithm (or an entire program) in pseudocode is to eliminate as much ambiguity as possible, that doesn’t necessarily mean that all of its operations will be spelled out in detail.

Guess a Number

The following are the basic steps for a game in which one player (in this case, the computer) picks a random number, and the other side (the user) must guess it, while being guided by the first player’s prompts.

```
1. secret (integer) ← random number between 1 and 100 (inclusive)
2. While user hasn’t guessed secret and has not given up:
    1. guess (integer) ← user input (use appropriate prompt and/or input filtering)
    2. If guess = 0,
        ▪ user gives up;
        otherwise, if guess = secret,
            ▪ user has guessed secret;
        otherwise, if guess < secret,
            ▪ inform user that guess is too low;
        otherwise,
            ▪ inform user that guess is too high.
3. Display outcome (value of secret, user success/failure).
4. play again (Boolean) ← user input (use appropriate prompt and/or input filtering)
5. If play again, return to step 1.
6. Done!
```