# CSC120 Week 5 Lab: Mastermind

Instructor: Mitsu Ogihara

The goal of this lab is to write a program for playing Mastermind with the user. Mastermind is a game for guessing a secret which is a four-digit number. The master of game and the player conduct multiple rounds of question-and-answering, where the player provides a guess and the master of the game provides feedback about the guess. Both the secret and any guess must be a four-digit number, where each digit is one of 1 through 9 and no two digits are identical to each other. Given a secret and a guess, the feedback provides consists of the number of "hits" and the number of "misses". A hit is a digit in the guess that appears at the same position in the secret, and a miss is a digit in the guess that appears at a different position in the secret. For example, if the secret is 4321 and the guess is 4519, then 4 appears at the same position and 1 appears at a different position, and so the guess has one hit and one miss.

Here is one execution of the program to show how the program works.

```
############################################################
# Let's play the game of MasterMind.
# I have a four-digit secret number with pairwise distinct
# digits from 1 to 9 and with no appearance of 0.
# Your goal is to guess the secret.
# You have 20 rounds to find out the secret.
# For each valid guess, I will inform you how many hits
# and how many misses you have.
# To terminate, enter a negative number.
# To review your guess history, enter 0.
############################################################
.... The secret is 5261....
==== Round 1
==== Enter your guess (0 for your history ): 1234
1:1234; 1 Hits, 1 Misses
==== Round 2
==== Enter your guess (0 for your history ): 5678
2:5678; 1 Hits, 1 Misses
==== Round 3
==== Enter your guess (0 for your history ): 1256
3:1256; 1 Hits, 3 Misses
==== Round 4
==== Enter your guess (0 for your history ): 0
==== This is your history:
1:1234; 1 Hits, 1 Misses
```

```
2:5678; 1 Hits, 1 Misses
3:1256; 1 Hits, 3 Misses
==== Round 4
==== Enter your guess (0 for your history ): 5261
==== Congratulations! You've found it.
```

## Goal of the Program in a Bit More Detail

Our program will operate as follows:

1. The program prints the rule of the game using the method `intro()`. The method is available in the template.

2. The program generates a random secret as an integer and records its four digits using the method `generateSecret()`. The program reports the secret.

3. The program enters a for-loop that iterates over the variable `i` from 1 to 20. The update operation is `i += advance` where `advance` is an `int` variable. The program also uses a `String` variable `history` whose initial value is `""`. The action in the loop is as follows:

   (a) The program initializes `advance` with the value of 0.

   (b) The program prompts the user to enter a guess and receives one.

   - If the guess is 0, the program reports the history.
   - Otherwise, if the guess is `-1`, the program terminates the program.
   - Otherwise, the program calls `validGuess()` to check the validity of the guess. The method obtains the four digits in the process, and returns a `boolean` informing whether or not the guess is valid.
   - If the guess is not valid, the program reports so; otherwise, the program calls `compare()` to obtain the number of hits and the number of misses, adds the result to the history, prints the result, and sets the value of `advance` to 1 (one round has been expended).
   - If the guess is not valid, the program reports that the guess is not valid.

## Technical Detail

In this program, we use the following static variables (those that are defined outside methods and so accessible from all the methods, and thus, should not be declared elsewhere). All of them are `int`.

- `secret, s1, s2, s3, s4`: the secret and its four digits;

- `guess, g1, g2, g3, g4`: the guess and its four digits;

- `hits, misses`: the number of hits and the number misses.

For each guess that the user generates, we reuse the second and third categories of variables.

How do we go from `guess` to `g1`, `g2`, `g3`, and `g4`? These quantities can be:

- the quotient of guess divided by 1000;
- the remainder of guess / 100 by 10;
- the remainder of guess / 10 by 10;
- the remainder of guess by 10.

After obtaining the four digits, the validity of guess can be tested using if they satisfy the following conditions:

- guess is between 1 and 9999.
- None of the four digits is 0 (which is equivalent to say that the product of the four is not 0).
- No two of the four digits are equal to each other (there are six such comparisons).

Use these pieces of information to write the code for validGuess. Once you have the code for it, you can recycle it to write the code for validSecret.

To write the code for generateGuess, you can think of the following infinite loop:

```
for ( ; ; ) {
  // generate a random integer between 1 and 9999
  //    and store it in the static variable secret;
  // call validSecret() and if the method returns true,
  //    break from the loop
}
```

The method validSecret() computes s1, s2, s3, and s4 for each secret candidate the program generates. Since these variables are static, you do not have to recalculate them.

To compute hits and misses, you can use a series of if-else statements that compares g1 to s1, s2, s3, and s4, g2 to s1, s2, s3, and s4, etc.