Programming Languages

G22.2110-001

Program # 1: Mastermind in Scheme

Due date: July 8th

For this assignment, you will write a standard Scheme program that plays the game of Mastermind. You should implement the algorithm as outlined below (or one of its variations).

Background

The game of Master Mind was introduced in the 1970s and quickly became one of the most popular games of that decade. Your task is to implement a specific variant of this game in Scheme.

The game has two players: the codemaker, and the codebreaker. The codemaker selects a "word", four letters long, from the range **A**..**F** which we will call the *code*. Duplicates are allowed. The code is not visible to the codebreaker, who has a limited number of attempts to guess it. After each guess, the code maker provides feedback by specifying how many letters were correct and in the right position, and how many were correct but not in the right position. For example, Figure 1 has a sample game.

The codemaker picks code **AEBC**. The codebreaker makes the guesses, and gets the feedback, in the table below.

Guess	Feedback	
ABCD	1:2	(**A** in the right position, **C** and **D** in wrong positions)
BCDE	0:3	(**B**, **C**, and **E** in wrong positions)
CDEA	0:3	(ditto)
DEAF	1:1	(**E** in the right position, **A** in an wrong position)
ABDE	1:2	(**A** in the right position, **B** and **E** in wrong positions)
AEDC	3:0	(**A**, **E**, and **C** in right positions)
AEBC	4:0	(all in right positions)

Figure 1: Sample game

The program will play the role of the codebreaker. The program will implement a variant of this game: the range of letters and the length of the sequence should be read in by the program at the start of a game[1]. The human will then choose a code (not revealing it to the computer). The computer will make successive guesses at the code. After each guess, the human player will provide *feedback* by inputting two numbers, as shown in the table above. The game ends when one of several events occurs:

1. The computer makes a guess which has all the letters in the correct position.

2. 10 guesses have been made.

3. The computer realizes that the human's feedback is inconsistent.

1 Algorithm

There are many practical algorithms that determine the code in a small number of guesses. For this assignment, implement the algorithm described in Algorithm 1 in standard Scheme.

First, some terminology. We will call a sequence of letters (from the letters in the range) of the correct length a *code*. We will also refer to a code

[1] The standard game (as shown above) uses 6 different letters (**A** through **F**), and the length of each code is 4.

as a *guess*, depending on context. The *codeset* is the set of all codes which may be the answer, based on all feedback so far. For example, if the guess is **AAAA**, and the feedback is 1:0, then **ABCD** and **FFFA** are possible codes, but **AABC** is not.

Initially, all codes are in the codeset. For the standard game, this has $6^4 = 1296$ members. The algorithm keeps track of the codeset, and determines at each step which guess will be guaranteed to eliminate the maximum number of members from it.

Algorithm 1 Codebreaker

```
$codeset \leftarrow$ all possible codes
Print an arbitrary code.
loop
    Read the feedback (in the form of two numbers).
    Update codeset, keeping only those members which are possible answers based on the feedback.
    Exit the loop if $|codeset| \leq 1$.
    for each possible guess, not just the members of codeset, do
        $score(guess) \leftarrow \underset{f \in \text{possible feedbacks}}{\texttt{minimum}} \; numRemovals(codeset, guess, f)$
    end for
    Print the guess which has the highest score.
end loop
if $|codeset| = 1$ then
    Print the remaining member of codeset.
    Read the feedback.
    If the feedback does not show that the guess is correct, there is an inconsistency in the feedback. Report this and terminate the game,
else
    The codeset is empty. This means there was an inconsistency in the feedback. Report this and terminate the game,
end if
```

numRemovals(*codeset*, *guess*, *feedback*) is the number of members of *codeset* for which *guess* would not result in *feedback*. For example, if *codeset* consists of:

$$\{\mathbf{AAAA}, \mathbf{AABC}, \mathbf{ADEF}, \mathbf{ACCF}\}$$

then

$$\begin{aligned}
\mathit{numRemovals}(\mathit{codeset}, \mathbf{ABCA}, 1:1) &= 4 \\
\mathit{numRemovals}(\mathit{codeset}, \mathbf{AAAA}, 4:0) &= 3 \\
\mathit{numRemovals}(\mathit{codeset}, \mathbf{FFFF}, 0:1) &= 2 \\
\mathit{numRemovals}(\mathit{codeset}, \mathbf{BFAB}, 0:2) &= 1
\end{aligned}$$

Your program should compile and run under `drscheme` using the "standard RSR5" language variant. `drscheme` is available on the Linux lab machines.

Important: Your program should not use any imperative features of Scheme. These are the functions which end with an exclamation point, e.g., `SET!`, `SET-CAR!`, etc.

Part of your grade will depend on program style, including proper indentation, comments, design, etc. Submission details will be posted on the class mailing list.

References

- `http://en.wikipedia.org/wiki/Mastermind_(board_game)`