G51PRG Exercise Four: Matching Pairs

Steven R. Bagley

Introduction

In this coursework, you are going to develop a simple 'Matching Pairs' game that can be played on the computer by two players. However, instead of the writing the program from scratch, an initial, single-player implemnetation of the game is given to you to start from. Unfortunately, this implementation has bugs (it doesn't work correctly) so for the first part of the coursework you are to 'debug' the provided solution to produce a working single-player version of the program.

For the second half of the coursework, you are to modify your working solution to implement a two-player version of the game according to the rules outlined below.

Again, there is no single correct solution for this exercise; any program which correctly implements the problem will be given a pass mark. To achieve higher grades, your program should make efficient and appropriate use of variables and functions in your solution.

Game Description

This is a description of the *single-player* version of the game, the two-player version is described later. This game is played on a six by six grid which is filled in with pairs of letters (from '`A`' to '`R`'). Each letter is distributed randomly on the board, and then covered up by a '`*`'. As the game is played, the user selects two squares to be temporarily uncovered (by entering co-ordinates for the squares via the keyboard). If the uncovered squares contain identical letters, then the letters are removed from the board and the squares are shown empty (no letter or '`*`' is displayed), otherwise after a brief pause to allow the user to memorize what was behind the squares, the letters are covered again and the user is asked to select another pair of squares to reveal. Play ends when all the letters have been revealed and removed, at which point the program reports back how many turns it took the player.

The following diagram shows the initial board display, and the grid reference system can be seen. The rows are identified by letters (case-insensitive) and the columns by numbers, allowing the user to enter a co-ordinate as '`e1`' or '`A6`'. The system should not accept incorrect co-ordinates, and ask the user to re-enter valid coordinates.

```
      1    2    3    4    5    6

A     *    *    *    *    *    *

B     *    *    *    *    *    *

C     *    *    *    *    *    *

D     *    *    *    *    *    *

E     *    *    *    *    *    *

F     *    *    *    *    *    *

      Enter square to reveal:
```

Revealed squares are shown thus:

```
      1    2    3    4    5    6

A     I    *    *    R    *    *

B     *    *    *    *    *    *

C     *    *    *    *    *    *

D     *    *    *    *    *    *

E     *    *    *    *    *    *

F     *    *    *    *    *    *

      No Match
```

With removed squares being represented as a blank space:

```
      1    2    3    4    5    6

A          *         *    *    *

B     *    *    *    *    *    *

C     *    *    *    *    *    *

D     *    *    *    *    *    *

E     *    *    *    *    *    *

F     *    *    *    *    *    *

      Enter square to reveal:
```

A working version (compiled for *school's Linux machines*) can be downloaded from the G51PRG website, which you can use to test the functionality.

1. Debugging

The source code for the single-player version can be download from the G51PRG website (along with a working compiled version) in this `.ZIP` file:

```
http://g51prg.cs.nott.ac.uk/pairs.zip
```

This program has a series of bugs in it which stop it from working correctly (and also, in some cases, from compiling). For the first part of the coursework, you are to study this program and correct the bugs in the program. In total, there are **five** bugs to find of varying degrees of obscurity.

As you discover, and fix, each bug you should add a comment to your code like this to identify the change you have made:

```
/* BUG FIX: 26/11/2014 srb -- missing semicolon inserted */
```

Where you should specify the current date, and your username (instead of `srb`) along with a brief, one-line description of the bug and your fix, so that we can identify what you have done. You are required to submit your modified, fixed version for marking.

Note: *Completing part one of this coursework alone will obtain a grade equivalent to a 40% pass.*

2. Two-player version

The second part of the coursework requires you to modified your corrected solution for part one to become a two-player version of the game. The modifications should run like this:

- The game starts with Player One's turn.
- Play alternates between the two players, but if a player correctly reveals two matching letters then they get to continue their go.
- Above the board, the current score is displayed, like this:

```
Player One:*   3   Player Two:    5

      1    2    3    4    5    6

A     I    *    *    R    *    *

B     *    *    *    *    *    *

. . .
```

- The asterisk ('`*`') is used to identify whose turn it is, and should alternate between `Player One:` and `Player Two:`.
- At the end of the game, the program should say which player won and report how many turns it took them.

Submission

You should submit two *separate* C files: your corrected, single-player version from part one and the two-player version from part two.