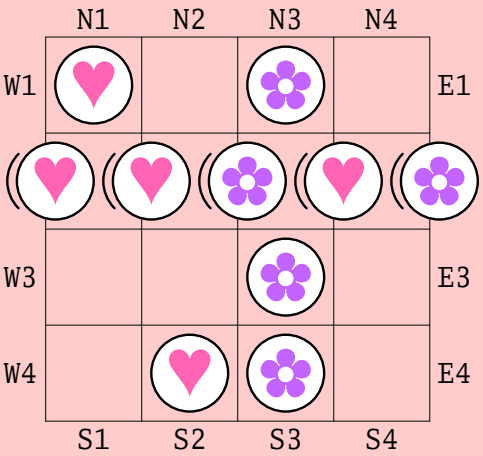
Hearts and Flowers – a programming competition

Mike Spivey, December 2002

This competition is open to all current Oxford undergraduates, with cash prizes of £150 funded by Data Connection, and another prize of books from O'Reilly to the value of £100. The competition is for a program (written in any reasonable language) that plays the game of *Hearts and Flowers* using input and output formats that are defined below. Entries must be handed to the receptionist at the Computing Laboratory by noon on St. Valentine's day, 14th February, 2003. The entries will be judged by Mike Spivey and Irina Voiculescu.

Hearts and Flowers is a game for two players, and is played on a board of $n \times n$ squares. One player has pieces marked with a heart, and the other has pieces marked with a flower. The aim of each player is to make a row of n of their pieces, horizontally, vertically, or along one of the two main diagonals of the board. The players take turns to slide one of their pieces onto the board from one of the four sides. If any existing pieces are adjacent to the place where the new piece is played, then they move along to make room as the new piece slides onto the board; if one of them slides off the board then it is removed and plays no further part in the game. The winner is the first player to complete a line of n pieces, whether that line is created during their move or during the opponent's move. If a move creates lines for both players simultaneously, then the game is a draw.



In the picture, Hearts is sliding a piece onto the board from the left, and a whole row of four pieces is sliding along to make room. A piece with a flower is about to fall off the right-hand side of the board, and the move ends by completing a vertical row for Flowers, so that Flowers wins the game.

The judges will run a tournament among all the programs that have been entered in order to decide the winner of the competition. The number of rounds in the tournament will be decided by the judges. Contestants may submit more than one program, but only the highest-scoring program from each contestant will be allowed to proceed beyond the first round of the tournament.

Specification: Your program should read from the standard input a line containing the dimension n of the board, followed by n lines that show the current state of the board. The program should choose the next move for Hearts and send to the standard output a single line that describes the move. Each move is encoded as a side of the board (`N`, `S`, `E`, or `W`), followed by a number from 1 to n. Your program should send nothing to the standard output apart from this one line. For example, the state before the move shown above would be encoded as follows.

```
4
H.F.
HFHF
..F.
.HF.
```

and the move itself is represented as `W2`.

In the tournament, a master program will be used to simulate games between your program and one of the other entries. It will run the two programs alternately, each time providing the current state of the board as input to the program. The master program computes the new board layout after each move, and takes care of swapping the pieces so that it appears to each program that it is playing as Hearts.

The tournament will be run on a 500 MHz Pentium-III with 256 MB of memory under Slackware Linux. Your program will have 30 seconds to decide on each move. If your program runs for longer than this, or fails to output a legal move, then it will be judged to have lost the game. If a game is not finished after each player has made $2n^2$ moves, then it will be regarded as drawn. Programs that are caught cheating in any way will be disqualified, and their authors will never again receive a Valentine's card.

How to enter: Each entry to the competition must be a computer program that can be run under Linux. Entries should be in the form of a floppy disk, labelled with your name, college and e-mail address, and containing a single gzip compressed tar file called entry.tgz or a ZIP archive called entry.zip with the following files:

README
: Description of the entry.

runme
: Executable program.

support
: Subdirectory containing supporting files (optional).

src
: Subdirectory containing program source.

src/Makefile
: Makefile for re-creating runme.

src/...
: Source file(s) for the program.

Your program runme will be run in the same directory into which we unpack the archive. If you wish, your program may read and write files you have put in the support subdirectory, perhaps to record its strategy between moves.

We will try to cope with most programming languages, but you can help us in a number of ways. If possible, include in your entry a statically-linked Linux executable containing all necessary libraries: that way, we can run your program even if different libraries are installed. This is the best option for entries written in C or C++, for example. For other programming languages, we make the following suggestions:

- For Haskell, package your entry as a standalone program that uses runhugs. We will install the latest version of HUGS.
- For Java, provide class files for the program in the support subdirectory, and make runme a shell script that invokes the Java virtual machine.
- For Oberon, make runme an Oxford Oberon bytecode file. We will take care of installing the latest runtime system.
- For Perl, TCL and similar languages, make runme a script that uses the #! convention.

If the supplied executable does not work, we will attempt to rebuild your program from the sources you provide. We will try our best to fix trivial problems, but we cannot promise to succeed.

If you develop your entry under an operating system different from Linux, we suggest sticking to the most basic libraries to maximize the chance that they will work the same way under Linux. If you cannot build your program on a Linux machine before submitting it, please note this fact in your README file, and we will do our best to help.

mike@comlab.ox.ac.uk