Homework 2 — Simple ML programs — assigned Monday 19 February — due Sunday 4 March

General instructions

Carefully state all preconditions for all function arguments. Make sure that your tests cover the space of permissible argument values well.

You may find yourself writing numerous recursive functions over lists. For each such function, carefully consider and document (in code comments) its traversal pattern. For each such function, determine if it can be expressed as an application of some built-in (typically higher-order) function from the `List` structure in the Standard ML Basis Library. If so, comment out the original recursive definition and provide a new definition in terms of library functions.

2.1 Puzzle (50pts) [A.3; K.2.2]

Devise an efficient algorithm for solving the puzzle discussed in class and implement it as an ML program. Report all solutions to the puzzle as well as the running time of the program (include the version of SML/NJ used, the hardware platform, and how the timing was done).

The puzzle, as given on NPR, is as follows.

> The NPR Weekend Edition Sunday puzzle by Will Shortz, given 7 January 2007. Challenge from January 7 and January 14: The object is to arrange 16 different letters of the alphabet into a 4X4 square, so four common, uncapitalized words read across and four common uncapitalized words read down. To start, the second word across is ruly. This is the only slightly unusual word in the square, every other word in it is one any school child would know. Can you do it?

Our version of the puzzle removes the stipulation that the second row should read RULY.

The simple ML program that implements an inefficient algorithm for this problem was discussed in class and is being separately provided on the course mailing list. Feel free to reuse any portions of that program that you find useful. The dictionary file is also being provided on the course mailing list.

2.2 Matrices (50pts) [A.1; K.2.2]

In this exercise, we adopt the type `real list` as our representation of column vectors and the type `real list list` as our representation of matrices,[1] and we develop functions for matrix arithmetic.

Preface your code with the declaration:

```
type realvector = real list
type realmatrix = real list list
exception Singular
```

[1] Note that there are better representations available in the Standard ML Basis Library.

1. (10pts) Write a function `rmvp: realmatrix * realvector -> realvector` that computes a matrix-vector product.
2. (10pts) Write a function `rmmp: realmatrix * realmatrix -> realmatrix` that computes a matrix-matrix product.
3. (10pts) Write a function `rmt: realmatrix -> realmatrix` to compute the matrix transpose.
4. (20pts) Write a function `rminv: realmatrix -> realmatrix` to compute the matrix inverse. Raise the exception as appropriate.

How to turn in

Submission instructions: see course mailing list.

Include the following statement with your submission, signed and dated:
I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual.