

Homework # 5

due Tuesday, March 1st, 3:30 PM

1 Simple Programming

Solve the following **one line** programming problems: Exercises 10, 19, 21 of Chapter 9. Your answers should not use recursion, but should instead use the `foldr` function to do the work. Put your solutions in file `oneline.sml` in your AFS `homework5` directory.

2 Vectors

We will need a new higher-order function `map2` that allows us to map over two lists in parallel. The problem is that one of the lists may end early. In that case, we need extra functions to complete the map.

```
map2 : ('a * 'b -> 'c) -> ('a -> 'c) -> ('b -> 'c) ->
      'a list -> 'b list -> 'c list
```

You should use pattern matching and recursion rather than `if` to distinguish the cases. You are permitted (and encouraged) to use `map` in your implementation (but it will only handle special cases for you).

You should be able to write:

```
- fun id x = x;
val id = fn : 'a -> 'a
- map2 (op ~) id (op ~) [100,99,98] [1,2,3,4];
val it = [99,97,95,~4] : int list
- map2 id (fn x => (x,"oops!")) (fn x => (42,x)) [1,2,3] ["hello","world"];
val it = [(1,"hello"),(2,"world"),(3,"oops!")] : (int * string) list
```

Next, re-write all the functions (except `projectAll`) from the previous homework to use currying and to use `foldl`, `foldr`, `map` and/or `map2` (your choice) instead of recursion. (Hint: `dot` will need two applications of higher-order functions!) You will need to write anonymous functions `fn x => ...` for some cases. Don't write any other helper functions, except that you are allowed to use `id` (the identity function).

Even though they don't use recursion, the functions `magn` and `project` need to be rewritten since they call functions that are now curried. The function `inSpan` must be rewritten to use `residue` as described below.

The definition of `projectAll` given in the previous homework only worked for orthogonal bases (when for all two different $\mathbf{w}, \mathbf{w}' \in S$ we have $\mathbf{w} \cdot \mathbf{w}' = 0$). So, for this Homework, we will define a new recursive function `residue` defined as follows:

$$R_{\{\}} \mathbf{v} = \mathbf{v}$$

$$R_{\{\mathbf{w}_1, \dots, \mathbf{w}_p\}} \mathbf{v} = R_{\{\mathbf{w}_2, \dots, \mathbf{w}_p\}} (\mathbf{v} - P_{\mathbf{w}_1} \mathbf{v}) \text{ where } \mathbf{w}'_1 = R_{\{\mathbf{w}_2, \dots, \mathbf{w}_p\}} \mathbf{w}_1$$

The residue represents the portion of a vector that is *not* in the span of the basis set S :

```
- residue [ [1.0] , [1.0, 0.0, ~1.0] ] [1.0, 2.0, 3.0];
val it = [0.0,2.0,0.0] : vector
```

Then, we redefine `inSpan` to be curried and to accept vectors whose residues are vanishingly small:

$$\mathbf{v} \in \text{span}(S) \text{ iff } |R_S \mathbf{v}| < \epsilon$$

```
- inSpan [[1.0], [1.0, 0.0, ~1.0]] [1.0, 2.0, 3.0];
val it = false : bool
- inSpan [[1.0], [1.0, 0.0, ~1.0]] [1.0, 0.0, 3.0];
val it = true : bool
```

The new types of the functions should be:

```
scale : real -> vector -> vector
dot : vector -> vector -> vector
magn : vector -> real
add : vector -> vector -> vector
project : vector -> vector -> vector
residue : vector list -> vector -> vector
inSpan : vector list -> vector -> bool
```

These functions (including `map2`) should be placed in `vector3.sml`, also in your AFS `homework5` directory.

3 Paper

Do Exercises 2 and 4 on page 163. For question 2, use the following code fragment to explain your answer about `let`:

```
(let ((x ...A...))
     (y ...B...))
...C...
```

Is `x` in scope in the area marked A ? B ? C? Then answer the same question for `let*` and then for `letrec`. Turn your answers for this part on paper.