MASTERMIND IMPLEMENTED

David Kauchak
CS52 – Spring 2017

Admin

Assignment 7

CS lunch today in Frank Blue Room

Miscellaneous SML

Assignment 3 revisited

2. [2 point] Write a function `exactMatches` that takes a secret code and a guess and returns the number of exact matches. (We read the lists from left to right. If one list is longer than the other, the trailing elements of the longer list are ignored.)

```
exactMatches : ''a list -> ''a list -> int
```

new type equivalence: code -> code -> int

```
fun exactMatches secret guess =
  length (List.filter
              (fn (x,y) => x=y)
              (ListPair.zip (secret, guess)));
```

Assignment 3 revisited

3. [3 points] a. Write a function `countColors` that takes a code and returns a list with the number of pegs of each color. It is convenient to order the elements of the list according to the enumeration of colors in `allColors`. For example, `countColors [Red,Blue,Yellow,Yellow]` returns `[1,0,2,0,1,0]`.

```
countColors : Peg list -> int list
```

new type equivalence: code -> int list

```
fun countColors theList =
  map (fn c => length (List.filter
                         (fn x => x=c)
                         theList))
      allColors;
```

Assignment 3 revisited

b. Write a function `totalMatches` that takes a secret code and a guess and returns the number of matches—exact or inexact.

```
totalMatches : Peg list -> Peg list -> int
```

new type equivalence: code -> code -> int

Suggestion: Use `countColors`. The number of Red matches between two lists is the minimum of the Red-counts of the two lists. If you know the number of matches for each color, you can simply add them to arrive at the total number of matches.

Assignment 3 revisited

b. Write a function `totalMatches` that takes a secret code and a guess and returns the number of matches—exact or inexact.

```
totalMatches : Peg list -> Peg list -> int
```

new type equivalence: code -> code -> int

```
fun sum []      = 0
  | sum (x::xs) = x + sum xs;

fun totalMatches  secret guess =
  sum (map Int.min
           (ListPair.zip (countColors secret, countColors guess)));
```

Assignment 3 revisited

4. [1 point] Use results from previous problems to write a function `matches` that takes the role of the codemaker. Given a secret code and a guess, the function returns an ordered pair of integers—first the number of exact matches and then the number of inexact matches.

```
matches : Peg list -> Peg list -> int * int
```

new type equivalence: code -> code -> response

```
fun matches secret guess =
      let
          val totalM = totalMatches secret guess;
          val exactM = exactMatches secret guess;
      in
          (exactM, totalM - exactM)
      end;
```

Assignment 3 revisited

Write a function `isConsistent` that takes a guess, a response, and a candidate code and returns a boolean value telling whether the candidate is consistent with the guess and response.

```
isConsistent : Peg list -> int * int -> Peg list -> bool
```

new type equivalence: code -> response -> code -> bool

```
fun isConsistent g r c = (matches g c) = r;
```

Assignment 3 revisited

6. [2 point] The next step in our strategy is to "thin out" a list of potential candidates for the secret code. Write a function `filterCodes` that takes a guess; a codemaker function, as described in Problem 4; and a prior list of candidates. It returns a list of those candidates that are consistent with the given guess and the response to it.

```
filterCodes : Peg list -> (Peg list -> int * int) ->
                  Peg list list -> Peg list list
```

new type equivalence: code -> codemaker -> code list -> code list

```
fun filterCodes guess f candidates =
  List.filter (isConsistent guess (f guess)) candidates;
```