Homework 5

February 21, 2014

1 High order functions

Use high-order functions `map`, `foldl`, or `foldr` for this question. Your solution should have the correct type as indicated.

1. Use foldr or foldl to implement a curried function

   `is_member: ''a list -> ''a -> bool`

   so that `is_member L x` returns true iff x is a member of L.

   Note that it is OK to get a warning "calling polyequal".

2. Write a function `splitter: string -> string list` that takes a string sentence and return a list of words in the sentence. The words in a sentence are separated by any character in the string `" ,.;?:!\t\n"`.

   Hint: you should use a library function

   `String.tokens: (char -> bool) -> string -> string list`.

   For example, `String.tokens (fn c => c = #" ") "hello world"` returns `["hello", "world"]`. That is, the `tokens` function can split a string sentence into word list using the a boolean function to determine the splitter character. In this example, the splitter character is the white space. Your implementation should allow any character in the string `" ,.;?:!\t\n"` to be a splitter.

   The function call `splitter "hello there, are you lost?"` should return `["hello", "there", "are", "you", "lost"]`.

   Note that this question does not require high order functions.

3. Given the global variable:

```
val stop_words =
"a,able,about,across,after,all,almost,also,am,among,an,and,any,are,as,at,be,
because,been,but,by,can,cannot,could,dear,did,do,does,either,else,ever,every,
for,from,get,got,had,has,have,he,her,hers,him,his,how,however,i,if,in,into,is,
it,its,just,least,let,like,likely,may,me,might,most,must,my,neither,no,nor,not,
of,off,often,on,only,or,other,our,own,rather,said,say,says,she,should,since,so,
some,than,that,the,their,them,then,there,these,they,this,tis,to,too,twas,us,
wants,was,we,were,what,when,where,which,while,who,whom,why,will,with,would,yet,
you,your";
```

implement a function `is_stop_word: string -> bool list` so that

`is_stop_word sentence`

returns a list of booleans with each true corresponding to a stop word and each false corresponding to a non-stop word.

For example, `is_stop_word "hello there, are you lost?"` should return `[false, true, true, true, false]`.

When you declare the variable `stop_words`, do not leave line breaks within the string quotes.

4. Using foldl or foldr, write a function `get_stop_words: string -> string list` that takes a sentence and return the list of unique stop words in the sentence.

   For example, `get_stop_words "hello there, how are you doing there?";`

   should return `["how", "are", "you", "there"]`. It is OK if you get the same list of words but they appear in different order.

   Notice that the word `there` appears twice in the sentence but only once in the returned list of stop words.

5. Using foldl or foldr, write a function

   `remove_stop_words: string -> string list`

   that takes a string input and return a list of words with stop words removed.

   For example, `remove_stop_words "hello there, are you lost?"` should return `["hello", "lost"]`.

   It is OK if you get the same list of words but they appear in different order.