Inf1-FP Revision Tutorial 5

Characters and Strings

Week of 9–13 Nov 2015

All of the following questions are taken directly from past exam papers. To check your solutions write tests that they behave according to the examples given, plus QuickCheck tests to test correctness.

1. (a) Write a function `f1 :: [String] -> String` to concatenate every word in a list that begins with an upper case letter. For example,

```
f1 ["This","Is","not","A","non","Test"] = "ThisIsATest"
f1 ["noThing","beGins","uPPER"] = ""
f1 ["Non-words","like","42","get","Dropped"] = "Non-wordsDropped"
f1 ["An","Empty","Word","","gets","dropped"] = "AnEmptyWord"
```

   Use *basic functions*, *list comprehension*, and *library functions*, but not recursion.

   (b) Write a second function `g1 :: [String] -> String` that behaves like `f1`, this time using *basic functions*, *recursion*, and the *library function to append two lists*, but not list comprehension or other library functions.

2. (a) Write a function `f2 :: String -> Bool` to verify that every vowel in a string is uppercase. In English, the following characters are the vowels: `'a'`, `'A'`, `'e'`, `'E'`, `'i'`, `'I'`, `'o'`, `'O'`, `'u'`, and `'U'`. The function should return `True` for strings that have no vowels at all. For example,

```
f2 "ALL CAPS" == True
f2 "r3cURsI0n" == True
f2 [] == True
f2 "normal text" == False
```

   Your definition may use *basic functions*, *list comprehension*, and *library functions*, but not recursion.

   (b) Write a second function `g2 :: String -> Bool` that behaves like `f2`, this time using *basic functions* and *recursion*, but not list comprehension or other library functions.

   (c) Write a third function `h2 :: String -> Bool` that also behaves like `f2`, this time using one or more of the following higher-order library functions:

```
map :: (a -> b) -> [a] -> [b]
filter :: (a -> Bool) -> [a] -> [a]
foldr :: (a -> b -> b) -> b -> [a] -> b
```

You may also use *basic functions*, but not list comprehension, other library functions, or recursion.

3. (a) Write a function `f3 :: String -> Bool` to verify that all the digits in a string are equal to or greater than 5. For example,

```
f3 "normal text" == True
f3 "number 75" == True
f3 "" == True
f3 "17 is a prime" == False
```

Your definition may use *basic functions*, *list comprehension*, and *library functions*, but not recursion.

(b) Write a second function `g3 :: String -> Bool` that behaves like `f3`, this time using *basic functions* and *recursion*, but not list comprehension or other library functions.

(c) Write a third function `h3 :: String -> Bool` that also behaves like `f3`, this time using one or more of the following higher-order library functions:

```
map :: (a -> b) -> [a] -> [b]
filter :: (a -> Bool) -> [a] -> [a]
foldr :: (a -> b -> b) -> b -> [a] -> b
```

You may also use *basic functions*, but not list comprehension, other library functions, or recursion.

4. (a) Write a function `f4 :: String -> Bool` to verify that every punctuation character in a string is a space. A character is punctuation if it is not a letter or a digit. The function should return `True` for strings that have no punctuation characters at all. For example,

```
f4 "Just two spaces" == True
f4 "No other punctuation, period." == False
f4 "No exclamations!" == False
f4 "What the @#$!?" == False
f4 "l3tt3rs and d1g1ts 0k" == True
f4 "NoSpacesAtAllOK" == True
f4 "" == True
```

Your definition may use *basic functions*, *list comprehension*, and *library functions*, but not recursion.

(b) Write a second function `g4 :: String -> Bool` that behaves like `f4`, this time using *basic functions* and *recursion*, but not list comprehension or other library functions.

(c) Write a third function `h4 :: String -> Bool` that also behaves like `f4`, this time using one or more of the following higher-order library functions:

```
map :: (a -> b) -> [a] -> [b]
filter :: (a -> Bool) -> [a] -> [a]
foldr :: (a -> b -> b) -> b -> [a] -> b
```

You may also use *basic functions*, but not list comprehension, other library functions, or recursion.

5. (a) Write a function `f5 :: String -> Bool` that takes a string, and returns `True` if every digit in the string is even. Any characters in the string that are not digits should be ignored. For example,

```
f5 "246" == True
f5 "2467" == False
f5 "x4y2z" == True
f5 "abc12" == False
```

Your definition may use *basic functions*, *list comprehension*, and *library functions*, but not recursion.

(b) Write a second function `g5 :: String -> Bool` that behaves like `f5`, this time using *basic functions* and *recursion*, but not list comprehension or other library functions.

(c) Write a third function `h5 :: String -> Bool` that also behaves like `f5`, this time using one or more of the following higher-order library functions:

```
map :: (a -> b) -> [a] -> [b]
filter :: (a -> Bool) -> [a] -> [a]
foldr :: (a -> b -> b) -> b -> [a] -> b
```

You may also use *basic functions*, but not list comprehension, other library functions, or recursion.