Haskell

Let’s review some of the Haskell concepts you have been learning on your own.

The answers are included, but try it yourself first.

Exercises – don’t use built in functions for these as we want the practice

• Write a recursive function to add up all the numbers in a list

• "flatten" a list of lists into a list –
  concat [[1,2], [], [3], [4,5]]
  = [1,2,3,4,5]
sum' [] = 0
sum' (x:xs) = x + sum' xs

flatten' [] = []
flatten' (x:xs) = x++ flatten' xs

More Exercises – don’t use built in functions

• return the first element in a list, giving an error if the list is empty
e.g., head' "string" = 's'

• Rewrite zip: take two lists and make a list of pairs -- e.g., zip' [1,2] [3,4,5] = [(1,3),(2,4)]
head' [] = error "no head"
head' (x:xs) = x

zip' [] x = []
zip' x [] = []
zip' (x:xs) (y:ys) = [(x,y)] ++ (zip' xs ys)

or
zip' (x:xs) (y:ys) = (x,y) : (zip1 xs ys)

Use list comprehensions to do these

• For a list of tuples, add each pair in a list
  addPairs :: [(Integer,Integer)] -> [Integer]
  [(2,3), (4,0), (5,5)] yields [5,4,10]

• Write a function to select only the items in a
  list which are bigger than 10

• List all pairs of numbers from [1,3,5] X [2,4,6]
  in which the second component is bigger
• addPairs list = \[x+y \mid (x,y)\in\text{list}\]

• select x = [a \mid a < x, a > 10]
  or filter (>10) list  (using the built-in function)

• bigger x y = [(a,b) \mid a < x, b < y, a < b]

• Write a function to remove an element from a list
  delete 4 [1,3,4,6] yields [1,3,6]
  delete 5 [1,3,4,6] yields [1,3,4,6]

• Write a function to remove all items from the first list that are also in the second list
  diff [1,3,4,5,7] [3,9] yields [1,4,5,7]
remove y [] = []
remove y (x:xs) = if y==x then remove y xs else x:(remove y xs)

or remove x list = [y|y <- list, y/=x]

setDif z [] = z
setDif xs (y:ys) = setDif (remove y xs) ys

Using list comprehensions

• Create a list which multiplies all combinations of elements of the two lists together

mult [1,2,3,4] [3,5,7,9] yields
[3,5,7,9,6,10,14,18,9,15,21,27,12,20,28,36]

Create a list of booleans which tells whether each item in a list is odd or even

oddEle [1..9]
yields [True,False,True,False,True,False,True,False,True]
\[
mult\;xs\;ys\;=\;[a\ast b\mid a<-\;xs,\;b<-\;ys]\]

\[
oddEle\;xs\;=\;[\text{odd}\;x\mid x<-\;xs]\]

or

\[
oddEle\;ys\;=\;[\;y``\mod`\;2==1\mid y<-\;ys]\]

---

**Factors**

- Find the factors of a number less than itself.

factors 24 yields

\[1,2,3,4,6,8,12\]
factors x = [n | n<-[1..(x `div` 2)], x `mod` n ==0]

• Generate all possible permutations of a list
  perms [1,2,3] yields
  [[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2],[3,2,1]]

Note: If you have two generators it treats them as nested:
  [(i,j) | i<- [1,2], j<- [10..14]]
yields the result
  [[(1,10), (1,11), (1,12),(1,13),(1,14),(1,14),(2,10),(2,11),
    (2,12),(2,13),(2,14)]]
remove y [] = []
remove y (x:xs) = if y==x then remove y xs else x:(remove y xs)
perms [] = [[]]
perms xs = [(x:y) | x <- xs, y <- perms (remove x xs)]

Quicksort

• Pick an element, called a *pivot*, from the list.
• Reorder the list so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the *partition* operation.
• *Recursively* sort the sub-list of lesser elements and the sub-list of greater elements.
qsort :: [Int] → [Int]
qsort [] = []
qsort (x:xs) =
    qsort smaller ++ [x] ++ qsort larger
where
    smaller = [a | a <- xs, a ≤ x]
    larger  = [b | b <- xs, b > x]

or
qsort [] = []
qsort (x:xs) = qsort [y|y<- xs, x≥ y] ++ x:(qsort [y|y<- xs, x< y])