What are the types of these functions?

\[
f1 \ 'a' \ _ = []
\]
\[
f1 \ x \ y = x:y
\]

\[
f2 \ (x:y:z:w:l) = w
\]
\[
f2 \ _ = 0
\]

\[
equal :: (Eq \ a) \Rightarrow \ [a] \rightarrow \ [a] \rightarrow \ Boolean
\]
\[
equal \ [] \ [] = True
\]
\[
equal \ _ \ [] = False
\]
\[
equal \ [] \ _ = False
\]
\[
equal \ (x:xs) \ (y:ys) = 
  \begin{align*}
  &\text{if } x == y \\
  &\text{then equal } xs \ ys \\
  &\text{else False}
  \end{align*}
\]
Use pattern matching to write a function `everyOther` that takes a list and returns a new list consisting of every other element in the original list starting with the first element. As an example, `everyOther [1,5,2,4,-1]` should return `[1,2,-1]`.

What does the following function do? What are examples of function calls that would evaluate to each of “group 1”, “group 2”, “group 3”, and “group 4”?

```haskell
import Data.Char

mystery x y
  | aL > 'm' && bL > 'm' = "group 4"
  | aL > 'm' && bL <= 'm' = "group 3"
  | aL <= 'm' && bL > 'm' = "group 2"
  | otherwise = "group 1"
where (a:_ ) = x
      (b:_ ) = y
      aL = toLower a
      bL = toLower b
```