FUNCTIONAL PROGRAMMING 2018 MOISE GABRIEL

QUESTION 3

(a)

```
> (++) :: [a] -> [a] -> [a]
> [] ++ ys = ys
> (x:xs) ++ ys = x: (xs ++ ys)
```

The time-complexity of (++) is $O(N)$, where $N =$ length xs

```
> reverse :: [a] -> [a]
> reverse [] = []
> reverse (x:xs) = reverse xs ++ [x]
```

(b) Let $T(N)$ be the number of steps needed for reverse xs, where length xs $= N$

Then, we have $T(N) \leq T(N-1) + (N-1) \Rightarrow T(N) \leq 1+2+\dots+(N-1) = \frac{(N-1)N}{2} \Rightarrow T(N) = O(N^2)$

(c)

```
> data Catlist a = Nil | One a | Cat (Catlist a) (Catlist a)
```

- [] can be represented as Nil, Cat Nil Nil, Cat (Cat Nil Nil) Nil
- [x] can be represented as One x, Cat (One x) Nil, Cat Nil (One x)

(d)

```
> rep :: [a] -> Catlist a
> rep []=Nil
> rep (x:xs) = Cat (One x) (rep xs)
```

(e)

```
> abs' :: Catlist a -> [a]
> abs' = flatCat []

> flatCat :: [a] -> Catlist a -> [a]
> flatCat ys Nil = ys
> flatCat ys (One x) = x : ys
> flatCat ys (Cat l r) = flatCat (flatCat ys r) l
```

The time complexity needed is $O(N)$ as we only made use of (:), and not (++), where N is the size of the list of type Catlist.

(f) The "abs''" function is not in general linear in the length of the returned list, as for example each element of the returned list could be made of N time Nil and the element, where N is [the] size of the returned list, so it makes it run quadratically in N.

(g)

```
> rev :: CatList a -> CatList a
> rev Nil = Nil
> rev (One x) = One x
> rev (Cat l r) = Cat (rev r) (rev l)
```

This runs in O(N), where N is the size of the input.

(h)

```
> reverse' :: [a] -> [a]
> reverse' = abs'. rev. rep
```

We first transform the list into a CatList using "rep" in O(N) time, then reverse it using "rev" in O(N) time and then transform it back to a normal list using "abs'", using O(N) time.

Therefore, the new definition of "reverse" is linear in the length of the argument.