CPSC 449 — Principles of Programming Languages
Theory Components of Assignment 4

Name: Haohu Shen
Student ID: 30063099
Tutorial: 03

Problem 2

Use the following definition of the function *concat*:

```
concat = foldr (++) []
```

You may also use the axiom (map++) on page 261 (under Exercise 11.31).

```
map f (ys ++ zs) = map f ys ++ map f zs
```

Prove that for all finite lists xs, and functions f,

```
concat (map (map f) xs) = map f (concat xs)
```

Solution. Before giving any proof, we list the definitions of the functions *map*, $foldr$ and all extended properties we are allowed to use, notice that all contents are referred from the textbook:

```
concat = foldr (++) [] -- (concat.3)

-- map.1 and map.2 are in Page 217 of the textbook
map f [] = [] -- (map.1)
map f (x:xs) = f x : map f xs -- (map.2)
map f (ys ++ zs) = map f ys ++ map f zs -- (map++)

-- foldr.1 and foldr.2 are in Page 222 of the textbook
foldr f s [] = s -- (foldr.1)
foldr f s (x:xs) = f x (foldr f s xs) -- (foldr.2)
```

Now we prove the statement in the problem holds for all finite lists xs and functions f by structural induction:

Proof Goals

- **Base Case**

  `concat (map (map f) []) = map f (concat [])` (base)

- **Induction Step**
  - **Assume:**

    `concat (map (map f) xs) = map f (concat xs)` (hyp)
  - **Prove:**

    `concat (map (map f) (x:xs)) = map f (concat (x:xs))` (ind)

Base Case

- **Want:**

```
concat (map (map f) []) = map f (concat [])
```

- **Left-hand side:**

```
concat (map (map f) [])
= concat []                                        by (map.1)
= foldr (++) [] []                                 by (concat.3)
= []                                               by (foldr.1)
```

- **Right-hand side:**

```
map f (concat [])
= map f (foldr (++) [] [])                         by (concat.3)
= map f []                                         by (foldr.1)
= []                                               by (map.1)
= L.H.S.
```

Thus it shows that the two sides are the same, which completes the proof of the base case.

Induction Step

- **Assume:**

```
concat (map (map f) xs) = map f (concat xs)            (hyp)
```

- **Want:**

```
concat (map (map f) (x:xs)) = map f (concat (x:xs))
```

- **Left-hand side:**

```
concat (map (map f) (x:xs))
= concat ((map f x) : map (map f) xs)              by (map.2)
= foldr (++) [] ((map f x) : map (map f) xs)       by (concat.3)
= map f x ++ (foldr (++) [] (map (map f) xs))      by (foldr.2)
```

Let

```
p = map (map f) xs
```

Then we have

```
L.H.S
= map f x ++ (foldr (++) [] p)                   by replacement
= map f x ++ concat p                            by (concat.3)
= map f x ++ concat (map (map f) xs)             by replacement
= map f x ++ map f (concat xs)                   by (hyp)
```

- **Right-hand side:**

```
map f (concat (x:xs))
= map f (foldr (++) [] (x:xs))                          by (concat.3)
= map f (x ++ (foldr (++) [] xs))                       by (foldr.2)
= map f (x ++ concat xs)                                by (concat.3)
= map f x ++ map f (concat xs)                          by (map++)
= L.H.S.
```

Since the final step makes the left- and right-hand sides equal, on the assumption that the induction hypothesis holds, This completes the induction step, and therefore the proof itself. □