```
compose_flips_append  THM

⊢  ∀k:ℕ, L1,L2:ℕ(k-1) List.
|  compose_flips(L1) o compose_flips(L2) = compose_flips(L1 @ L2)
| by Auto THEN Unfold `compose_flips` 0
|
1. k : ℕ
2. L1 : ℕ(k-1) List
3. L2 : ℕ(k-1) List
⊢  compose_list(map(λi.(i, i+1);L1)) o compose_list(map(λi.(i, i
|  +1);L2))
|  =
|  compose_list(map(λi.(i, i+1);L1 @ L2))
| by Subst map(λi.(i, i+1);L1 @ L2) ~ (map(λi.(i, i+1);L1)
|    @ map(λi.(i, i+1);L2)) 0
|
|\
| .....equality .....
| 1. k : ℕ
| 2. L1 : ℕ(k-1) List
| 3. L2 : ℕ(k-1) List
| ⊢  map(λi.(i, i+1);L1 @ L2) ~ (map(λi.(i, i+1);L1)
|    @ map(λi.(i, i+1);L2))
|   by BackThruLemma `map_append_sq` THEN Auto
|
.\
. 1. k : ℕ
. 2. L1 : ℕ(k-1) List
. 3. L2 : ℕ(k-1) List
. ⊢  compose_list(map(λi.(i, i+1);L1)) o compose_list(map(λi.(i, i
. |  +1);L2))
. |  =
. |  compose_list(map(λi.(i, i+1);L1) @ map(λi.(i, i+1);L2))
. | by Fold `compose_flips` 0 THEN RWO "compose_append<" 0 THEN Auto'
. |
. 1. k : ℕ
. 2. L1 : ℕ(k-1) List
. 3. L2 : ℕ(k-1) List
. ⊢  compose_flips(L1) o compose_flips(L2)
.    =
.    compose_list(map(λi.(i, i+1);L1)) o compose_list(map(λi.(i, i
.    +1);L2))  by Fold `compose_flips` 0 THEN Auto
END
```