## Append:

# Append.txt

```
// Append function p 133
local Append L1 L2 L3 Out Reverse Out1 in
  Append = fun {$ Ls Ms}
              case Ls
                 of nil then Ms
                 [] '|'(1:X 2:Lr) then Y in
                    Y = {Append Lr Ms}
                    // skip Full
                    (X|Y)
               end
           end
   Reverse = fun {$ Ls}
              case Ls
                 of nil then nil
                 [] '|'(1:X 2:Xr) then X1 in
                 X1 = (X|ni1)
                 {Append {Reverse Xr} X1}
              end
            end
  L1 = (1|(2|(3|ni1)))
  L2 = (4|(5|(6|ni1)))
  L3 = (7|(8|(9|(10|ni1))))
  Out = {Append L1 L2}
  Out1 = {Reverse L3}
  skip Browse Out
  skip Browse Out1
  skip Full
end
// From the information of the store, when we pass L3: (7|(8|(9|(10|ni1)))) into Reverse,
// it is actually doing :
// 1. {Append {Reverse [8,9,10]} [7]} -----> {Append [10,9,8] [7]}----->[10,9,8,7] (final result)
// 2. {Reverse [8,9,10]} ----> {Append {Reverse [9,10]} [8]} -----> {Append [10,9] [8]}
// 3. {Reverse [9,10]} ----> {Append {Reverse [10]} [9]} -----> {Append [10] [9]} ----->[10,9]
// 4. {Reverse [10]} ----> {Append {Reverse []} [10]} ---->[10]
```

#### **Store and Environment:**

```
Store: ((77, 63), '|'(1:74 2:75)),
((76, 61, 51, 43, 21), 10),
((75, 66), '|'(1:72 2:73)),
((74, 59, 37, 19), 9),
((73, 69, 71, 68, 65, 28, 24), '|'(1:25 2:26)),
((72, 31, 17), 8),
((70, 32), nil()),
((67, 60, 56, 58, 55, 34, 30), '|'(1:31 2:32)),
((64, 62, 53), '|'(1:59 2:60)),
((57, 38), nil()),
((54, 52, 48, 50, 40, 36), '|'(1:37 2:38)),
((49, 44), nil()),
((39, 46, 42), '|'(1:43 2:44)),
((47, 22), nil()),
((45), ni1()),

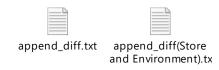
((41, 20), '|'(1:21 2:22)),

((35, 18), '|'(1:19 2:20)),

((33), '|'(1:51 2:52)),
((29, 16), '|'(1:17 2:18)),
((27), '|'(1:61 2:62)),
((25, 15), 7),
((26), nil()),
((23, 11), '|'(1:15 2:16)),
((8), proc(["Ls","Ms","EXU1"],[case Ls of ni1() then [EXU1 = Ms] else [case Ls of '|'(1:X 2:Lr) then [local ["Y"] [local ["EXU2","EXU3"] [E
 else [skip]]],[("Append",8)])),
((9), Unbound),
((10), Unbound),
((12), Unbound),
((13), proc(["Ls","EXU1"],[case Ls of nil() then [EXU1 = nil()] else [case Ls of '|'(1:X 2:Xr) then [local ["X1"] [local ["EXU2","EXU3"] [E
2"],EXU3 = X1,"Append" "EXU2" "EXU3" "EXU1"]]] else [skip]]],[("Reverse",13),("Append",8)])),
((14), '|'(1:76 2:77)),
((1), Primitive Operation),
((2), Primitive Operation),
((3), Primitive Operation),
((4), Primitive Operation),
((5), Primitive Operation),
((6), Primitive Operation),
((7), Primitive Operation)
Mutable Store: Empty
Current Environment : ("Append" -> 8, "L1" -> 9, "L2" -> 10, "L3" -> 11, "Out" -> 12, "Reverse" -> 13, "Out1" -> 14, "IntPlus" -> 1, "IntMi
```

```
// From the information of the store, when we pass L3: (7|(8|(9|(10|nil)))) into Reverse,
// it is doing:
// 1. {Append {Reverse [8,9,10]} [7]} -----> {Append [10,9,8] [7]} -----> [10,9,8,7] (final result)
// 2. {Reverse [8,9,10]} ----> {Append {Reverse [9,10]} [8]} -----> {Append [10,9] [8]}
// 3. {Reverse [9,10]} ----> {Append {Reverse [10]} [9]} -----> [10,9]
// 4. {Reverse [10]} ----> {Append {Reverse []} [10]} ----> [10]
```

### Append\_diff:



```
// From the information of the store, when we pass [4,3,2,1] into Reverse
// {Reverse [4,3,2,1]} -----> {ReverseD [4,3,2,1] Out []} -----> Return [1,2,3,4]

// {ReverseD [4,3,2,1] Out []} -----> {ReverseD [3,2,1] Out [4]}

// {ReverseD [3,2,1] Out [4]} -----> {ReverseD [2,1] Out [3,4]}

// {ReverseD [2,1] Out [3,4]} -----> {ReverseD [1] Out [2,3,4]}

// {ReverseD [1] Out [2,3,4]} -----> for the [] case, ReverseD bound [1,2,3,4] to Out, Out = [1,2,3,4]

// Return [1,2,3,4]
```

### Reverse with size of 6: