CS 558 — 29 August 2024

Note Title 2024-08-29

```
ones = 1 : ones

take 5 ones
   ↓
take 5 (1 : ones)
   ↓
1 : take 4 ones
     ↓
1 : take 4 (1 : ones)
     ↓
1 : (1 : take 3 ones)
   ...
1 : (1 : (1 : (1 : (1 : take 0 ones))))
     ↓
1 : (1 : (1 : (1 : (1 : []))))
                    = [1,1,1,1,1]
```

```
take n (x:xs) =
            x : take (n-1) xs
take 0 _ = []
```

```
[f x | x <- [1,2 ..]]
```

---

xs is some list you have computed

```
xs :: [String]   e.g.   ["Alice", "Bob", ..., "Z"]
          we want
            [(1, "Alice"), (2, "Bob"), ..., (563, "Z")]
f :: [String] -> [(Int, String)]
f xs =   zip [1 .. length xs] xs
 OR
f xs = zip [1..] xs
```

(:) adds single element at the front of the list

(++) appends two lists

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

Example

```
[1,2] ++ [3,4]
   ↓
1 : ([2] ++ [3,4])
   ↓
1 : (2 : ([] ++ [3,4]))
   ↓
1 : (2 :  [3,4])
              = [1,2,3,4]
```