CSCI 334:
Principles of Programming Languages

Lecture 15 addendum:
Mapping and folding examples

Instructor: Dan Barowy

Williams

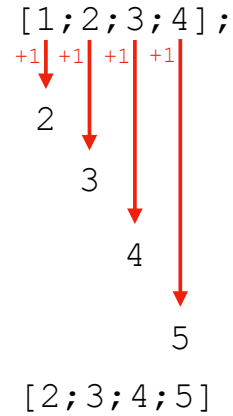
Mapping and Folding

map

Intuition:

map

```
List.map (fun x -> x + 1) [1;2;3;4];
```



map

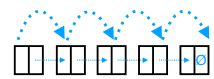
```
[2;8;22;4]
|> List.map (fun x -> x + 1)
|> List.map float
|> List.map (fun x -> x / 3.3)
|> List.sort
```

```
[0.9090909091; 1.515151515; 2.727272727;
6.96969697]
```

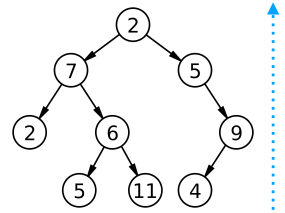
fold

structural recursion → fold it!

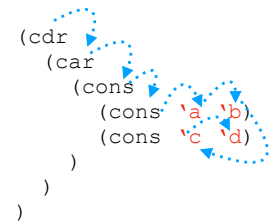
(in a nutshell: any problem that recurses on a subset of input)



list length



tree height



evaluation

fold

Intuition:

fold left

```
List.fold (fun acc x -> acc+x) 0 [1;2;3;4]
```

```
acc = 0, [1;2;3;4]
acc = 0+1, [2;3;4]
acc = 1+2, [3;4]
acc = 3+3, [4]
acc 6+4, []
returns acc = 10
```

fold right

```
List.foldBack
  (fun x acc -> acc+x) [1;2;3;4] 0
```

```
[1;2;3;4], acc = 0
[1;2;3], acc = 0+4
[1;2], acc = 4+3
[1] acc = 7+2
[], acc = 9+1
returns acc = 10
```

Try this at home!

```
let number_in_month(ds: Date list)(month: int) : int =
```

- Write a function `number_in_month` that takes a list of dates (where a date is `int*int*int` representing year, month, and day) and an `int month` and returns how many dates are in `month`
- Use `List.fold`

fold

```
let number_in_month(ds: Date list)(month: int) : int =
  ds
  |> List.fold (fun acc (_,mm,_) ->
      if month = mm then
        acc + 1
      else
        acc
    ) 0
```