

OCaml: Practice

Concepts of Programming Languages
Lecture 3

Announcements

- » Assignment 1 is due on Thursday, 8pm (can submit until 11:59pm)
- » Please sign the course manual
- » Office Hours have started! Look at calendar on course webpage
- » Update standard library:
 - `cd path/to/stdlib320`
 - `git pull`
 - `eval $(opam env)`
 - `opam upgrade stdlib320`

Outline

» OCaml practice today!

» We will cover the following:

- Converting loops to recursion
- How to get around mutating variables
- Nested Loops
- Lists

» Leetcode problems!

demo

(loops and recursion)

Lists

» Very much like the linked list, **not vectors**

» Syntax:

- `[]` for an empty list
- `x::l` for “cons”-ing an element `x` to the front of list `l`
- `[x1; x2; ...; xn]` for a fixed list

Some Examples

'a list

```
let l1 = []
```

int list

```
let l2 = 1::l1
```

int list

```
let l3 = 2::3::l2
```

int list

```
let l4 = [1; 2; 3]
```

(* Are l3 and l4 equal? *)

Some Examples

'a list

```
let l1 = []
```

int list

```
let l2 = 1::l1
```

int list

```
let l3 = 2::3::l2
```

int list

```
let l4 = [1; 2; 3]
```

(* Are l3 and l4 equal? *)

:: is right associative
So, this would be
equivalent to
2::(3::l2)

Generating a List

» Generate a list of n natural numbers in increasing/decreasing order

```
int -> int list
let rec generate n =
  if n = 0
  then []
  else n::(generate (n-1))

(* generate 5 = [5; 4; 3; 2; 1] *)
```

```
int -> int list
let generate n =
  let rec gen_helper n k =
    if n = 0
    then []
    else k::(gen_helper (n-1) (k+1))
  in
  gen_helper n 1

(* generate 5 = [1; 2; 3; 4; 5] *)
```


Using Lists

» We will use pattern-match (like a switch statement but MUCH BETTER)

```
match <list-expr> with  
| [] -> <body-for-null-expr>  
| h::t -> <body-for-non-null-expr>
```

Examples

```
'a list -> int
let rec length lst =
  match lst with
  | [] -> 0
  | h::t -> 1 + (length t)
```

```
int list -> int
let rec sum lst =
  match lst with
  | [] -> 0
  | h::t -> h + sum t
```

demo

(basic lists)

Leetcode Medium

2. Add Two Numbers

Medium

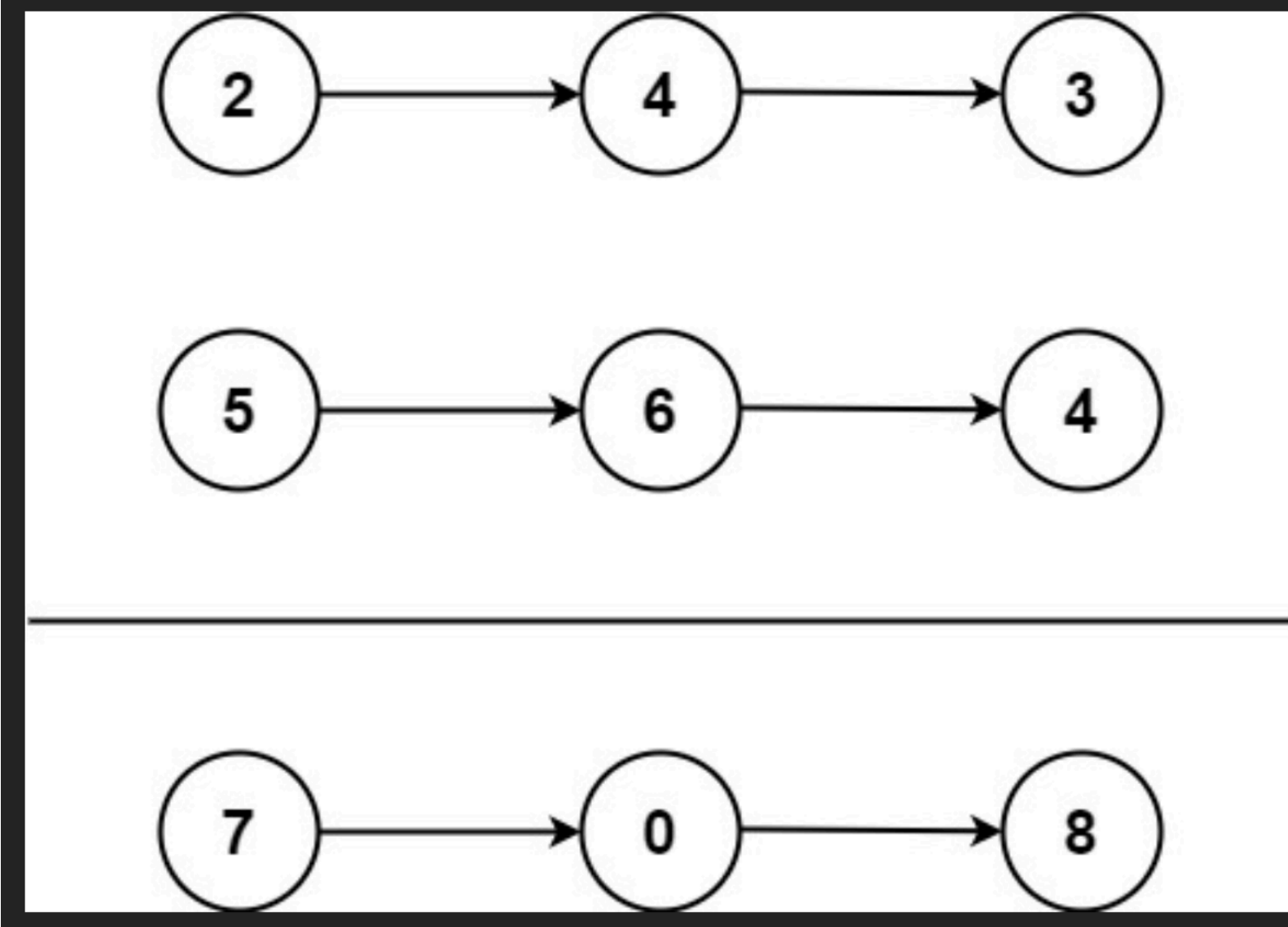
Topics

Companies

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:



Leetcode Hard

25. Reverse Nodes in k-Group

Hard

Topics

Companies

Given the `head` of a linked list, reverse the nodes of the list `k` at a time, and return *the modified list*.

`k` is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of `k` then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

Example 1:

