CSE216
Foundations of Computer Science

Instructor: Zhoulai Fu

State University of New York, Korea

Lab Exercises on function types

Warm-up Lab Exercise

- Write a power function of type **power: int -> int -> int** using only recursion, without using "$**$" or other build-in functions in Ocaml

Exercise 1 (done last time)

- Write down the types of the defined functions in OCaml:

  - a)   let double x = 2*x;;

  - b)   let square x = x*x;;

  - c)   let twice f x = f (f x);;

  - d)   let quad = twice double;;

  - e)   let fourth = twice square;;

Exercises 2

- Write down the types of the defined functions in OCaml:

  - a)   let tripleFloat x = 3.0*.x;;

  - b)   let thrice f x = f(f(f(x)));;

  - c)   let composition f g x = f(g(x));;

  - d)   let div x y = x/y;;

  - e)   let triple3 = thrice tripleFloat;;

Exercise 3

- Define twice such that takes f and x as an input, applies f to x a total of 2 times.

Exercise 4

- Generalize twice to a function repeat, such that repeat f n x applies f to x a total of n times. That is,

  - repeat f 0 x yields x

  - repeat f 1 x yields f x

  - repeat f 2 x yields f (f x) (which is the same as twice f x)

  - repeat f 3 x yields f (f (f x))

Exercise 5: Any Difference Here?

```
let abs = fun x -> if x<0 then -x else x

let abs (x:int) = fun x -> if x<0 then -x else x

let abs x = fun x -> if x<0 then -x else x

let abs (x:int) : int = if x<0 then -x else x

let abs:int -> int = fun x -> if x<0 then -x
else x
```

Exercise 6:
Determine function types

```
let rec p x y  =
  if y=0 then 1 else x * (p x (y-1)) ;;


let rec p (x, y) =
    if y=0 then 1 else x * p (x, y-1) ;;
```

Exercise 7: Where is the bug?

```
(*buggy*)
# let rec p x y  =
   if y=0 then 1 else x * (p x y-1) ;;
val p : int -> int -> int = <fun>

# p 3 2 ;;
Stack overflow during evaluation (looping
recursion?)
```