CSE216
Foundations of Computer Science

Instructor: Zhoulai Fu

State University of New York, Korea

Agenda

- Homework 08

1. (Points = 60) Define the function `power` of type `float->int->float`. The function takes a float `x` and an integer `y` as inputs, and return x to the power of y. You can use `failwith` to raise exceptions if needed.

```
Test cases:
- `power 3. 5` should equal to 243.
- `power 0. 0` should raise an exception
- `power 0. (-2)` should also raise an exception
- `power 2. (-5)` should equal 0.03125
- `power (-8.9) 0`  should equal to 1.
```

2. (Points = 20) What is your result of `print_float (power (-27.6) (-3))`?

3. (Points = 20) Is the result above exactly the same as `(-27.6)**(-3.)` in Ocaml?

1.

```
# let rec power x n =
    if x = 0. then if n > 0 then 0. else failwith "undefined"
    else
    if n < 0 then (power x (n+1))/.x
    else if n = 0 then 1.
    else x *. power x (n-1) ;;
val power : float -> int -> float = <fun>
# (power(-27.6) (-3)) ;;
- : float = -4.75633848692121e-05
```

2.

```
# print_float(power(-27.6) (-3)) ;;
-4.75633848692e-05
- : unit = ()
```

3.

```
# (-27.6) ** (-3.) ;;
- : float = -4.75633848692121e-05
```

Entirely difference in type. In terms of the calculation results, they have a different precision