Class 1: Haskell Basics

January 17

introductions

components

- **classwork (10%):**
  - on Tuesdays at 3:30–5 p.m.
  - combination of lecture and in-class exercises
- **homework (90%):**
  - due Mondays at 10 p.m.
  - designed to be relatively short and mostly autograded

odds and ends

- prerequisite: CIS 1200 or equivalent

- *Website*: class policies and schedule
  Ed: announcements (did you get the reminder?) and Q&A
  Canvas: attendance grades only
  Gradescope: homework submissions and grades

- masks are required for all class-related activities
  for illness or otherwise, extensions and excused absences are available!

- cis1940-spring23 repo walkthrough

why Haskell?

functional **pure** **nicely typed**

```
int acc = 0;
for (int i = 0; i < lst.length; i++) {
    acc = acc + 3 * lst[i];
}
```

$\rightarrow$

```
sum (map (3 *) lst)
```

```
var a = getData();
if (a != null) {
  var b = getMoreData(a);
  if (b != null) {
     var c = getMoreData(b);
     if (c != null) {
        var d = getEvenMoreData(c);
        if (d != null) {
          output d;
        }
     }
  }
}
```

```
do
  a <- getData
  b <- getMoreData a
  c <- getMoreData b
  d <- getEvenMoreData c
  output d
```

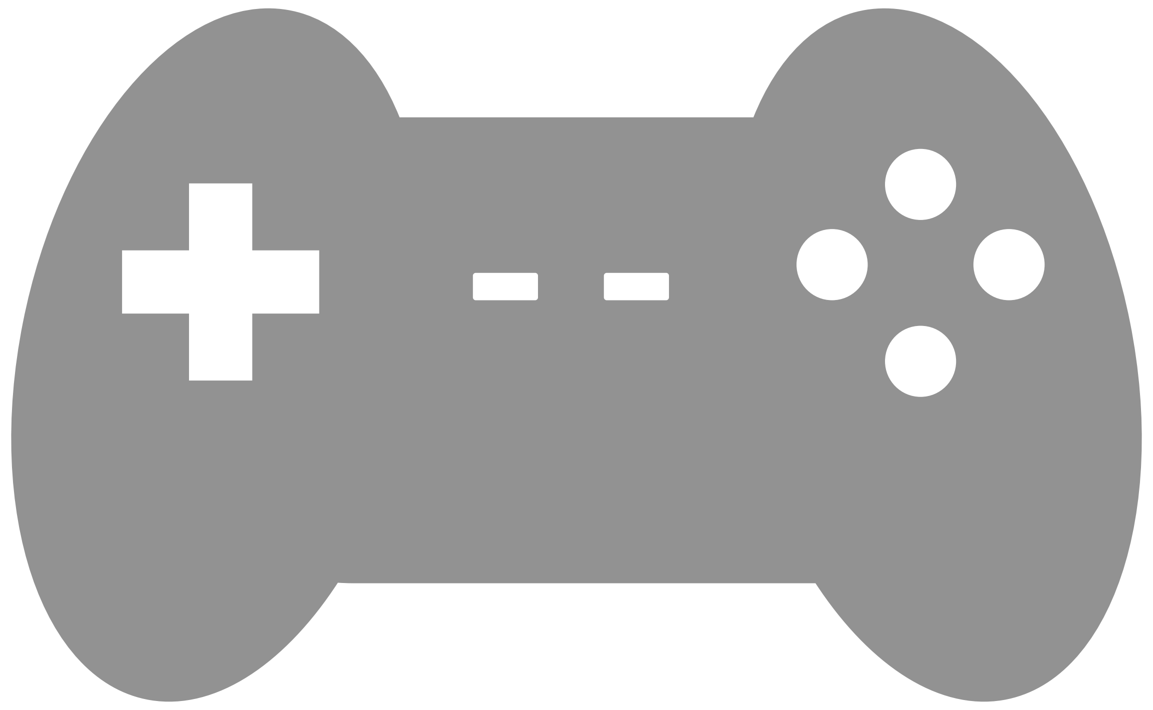
https://philipnilsson.github.io/Badness10k/posts/2017-05-07-escaping-hell-with-monads.html

```
var a = getData();
for (var a_i in a) {
  var b = getMoreData(a_i);
  for (var b_j in b) {
    var c = getMoreData(b_j);
    for (var c_k in c) {
      var d = getEvenMoreData(c_k);
      for (var d_l in d) {
        output d_l;
      }
    }
  }
}
```

$\rightarrow$

```
do
  a <- getData
  b <- getMoreData a
  c <- getMoreData b
  d <- getEvenMoreData c
  output d
```

https://philipnilsson.github.io/Badness10k/posts/2017-05-07-escaping-hell-with-monads.html



most importantly, it’s fun!

let’s learn some Haskell!

`x :: Int`	"x has type Int"
`x = 3`	"x is defined to be 3"

Declarations

(GHCi example)

```haskell
i :: Int
i = 3

d :: Double
d = 3.14
```

```haskell
b :: Bool
b = False
```

```haskell
c :: Char
c = ‘x’

s :: String
s = “Hello!”
```

Basic Types

(arithmetic example)

```haskell
add3 :: Int -> Int
add3 n = n + 3
```

Functions

```haskell
factorial :: Int -> Int
factorial 0 = 1
factorial n = n * factorial (n - 1)
```

Pattern Matching

(sumtorial exercise)

```haskell
nums :: [Int]
nums = [1, 5, 19]
```

Lists

```
[]                []
[3]               3 : []
[2, 3]            2 : (3 : [])
[1, 2, 3]         1 : (2 : (3 : []))
```

Lists

```haskell
add3List :: [Int] -> [Int]
add3List [] = []
add3List (x : xs) = x + 3 : add3List xs
```

(double exercise)

(swap example)

questions?

looking ahead

- Homework 0 (installation) due Friday but preferably earlier
- Homework 1 (this class) due next Monday

- Ernest's office hours: Thursdays 4–5 p.m.
  Jessica's office hours: Mondays 10–11 a.m. and by appointment

- we will be in person next class!

if you are not registered for the class, please stay a minute