Class 1: Haskell Basics

January 23

introductions

class:
on Tuesdays at 3:30–5 p.m.
graded on attendance and effort

homework:
due Mondays at 10 p.m.
graded on correctness and style

components

prerequisite:
CIS 1200 or equivalent

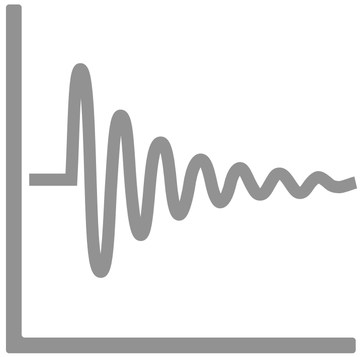
odds and ends

Website: class policies and schedule
Ed: announcements and Q&A
Canvas: attendance grades
Gradescope: homework submissions and grades

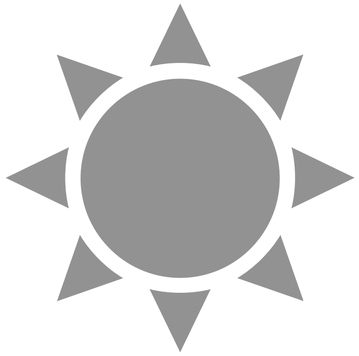
odds and ends

(tour: cis1904-spring24)

why Haskell?



functional



pure

nicely typed

```
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

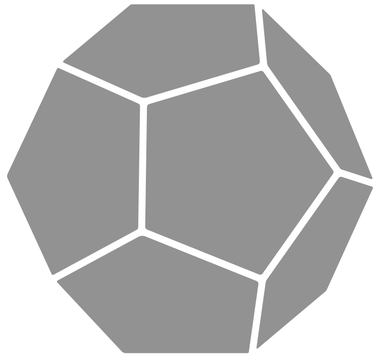
```
var a = getData();
for (var a_i in a) {
  var b = getMoreData(a_i);
  for (var b_j in b) {
    var c = getMoreData(b_j);
    for (var c_k in c) {
      var d = getEvenMoreData(c_k);
      for (var d_l in d) {
        output d_l;
      }
    }
  }
}
```

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

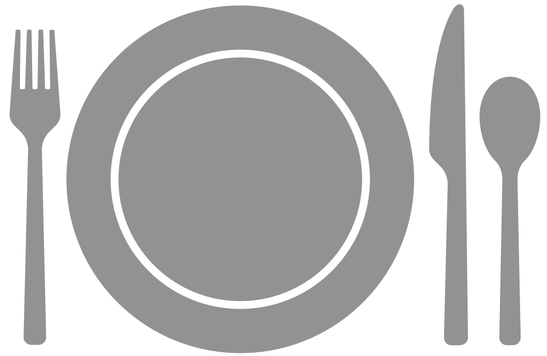
https://philipnilsson.github.io/Badness10k/posts/2017-05-07-escaping-hell-with-monads.html

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

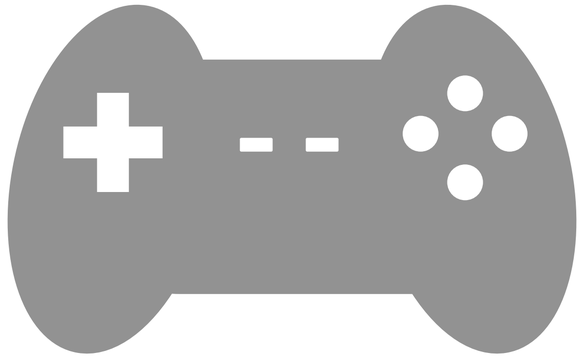
```
sum (map (3 *) lst)
```



abstraction



wholemeal programming



most importantly, it’s fun!

the programming process

let’s learn some Haskell!

```
x :: Int
```
“x has type Int”

```
x = 3
```
“x is defined to be 3”

declarations

(example: GHCi)

you should always have type signatures!

```
i :: Int
i = 3

d :: Double
d = 3.14
```

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

```
c :: Char
c = ‘x’

s :: String
s = “Hello!”
```

basic types

(example: arithmetic)

Join by Web **PollEv.com/jessicashi159** Join by Text Send **jessicashi159** and your message to **37607**

Rewrite the definition of ex3 using infix instead of prefix notation.

Correct answer: `j `div` i`

Join by Web **PollEv.com/jessicashi159** Join by Text Send **jessicashi159** and your message to **37607**

What does ex3 evaluate to? Use GHCi to see!

Correct answer: 1

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

functions

```haskell
combine :: Int -> Int -> Int -> Int
combine x y z = x + y + z + 17

result :: Int
result = combine 1 2 3
```

functions

please follow along!

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

pattern matching

Join by Web **PollEv.com/jessicashi159** Join by Text Send **jessicashi159** and your message to **37607**

Write the type signature of sumtorial.

Correct answer: `sumtorial :: Int -> Int`

(exercise: sumtorial)

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

lists

```
[]
[3]
[2, 3]
[1, 2, 3]
```

```
[]
3 : []
2 : 3 : []
1 : 2 : 3 : []
```

lists

please follow along!

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

(exercise: double and swap)

questions?

looking ahead:
Homework 0 (installation) due Friday but preferably earlier
Homework 1 (this class) due next Monday

office hour schedule posted on Ed

if you are not registered
(but would like to be),
please stay a minute