

# PEP Scala (3)

- Email: christian.urban at kcl.ac.uk  
Office: N7.07 (North Wing, Bush House)
- Slides & Code: KEATS
- Office Hours: Thursdays 12:00 – 14:00  
Additionally: (for Scala) Tuesdays 10:45 – 11:45

# Preliminary 6

Raw marks (261 submissions):

- 3%: 219
- 2%: 19
- 1%: 0
- 0%: 23 (4 no submission)

```
def collatz(n: Long) : Long =  
  {  
    val toReturn = collatzHelper(n, 0)  
    toReturn  
  }
```

```
def collatz(n: Long) : Long =  
  {  
    val toReturn = collatzHelper(n, 0)  
    toReturn  
  }
```

---

```
def collatz(n: Long) : Long =  
  collatzHelper(n, 0)
```

# Default Arguments



```
def collatzHelper(n: Int, a: Int = 0) : Int = ...
```

```
collatzHelper(n, 3)
```

```
collatzHelper(n, 0)
```

```
collatzHelper(n) // a = 0
```

# Last Week: Options & HO Funs.

```
List(7,2,3,4,5,6).find(_ < 4)  
res: Option[Int] = Some(2)
```

```
List(5,6,7,8,9).find(_ < 4)  
res: Option[Int] = None
```

```
List(1,2,3,4,5).map(x => x * x)  
res: List[Int] = List(1, 4, 9, 16, 25)
```

# Web-Crawler (1)

```
def get_page(url: String) : String = {  
  Try(fromURL(url)("ISO-8859-1").take(10000).mkString)  
    .getOrElse { println(s" Problem with: $url"); "" }  
}
```

## Web-Crawler (2)

```
val http_pattern = """https?://[^\"]*""".r
```

```
val email_pattern =
```

```
"""([a-z\d\.-]+)@([\da-z\.-]+\.[a-z\.]{2,6})"""
```

```
def unquote(s: String) = s.drop(1).dropRight(1)
```

```
def get_all_URLs(page: String): Set[String] =  
  http_pattern.findAllIn(page).map(unquote).toSet
```

```
// returns all URLs in a page
```



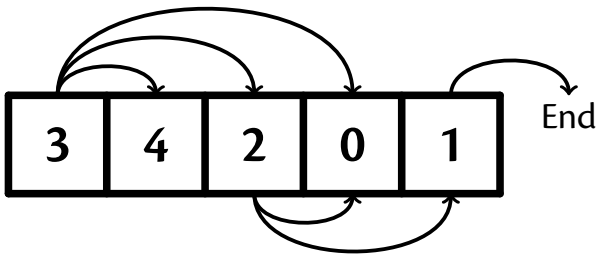
# Web-Crawler (3)

```
def crawl(url: String, n: Int) : Unit = {  
  if (n == 0) ()  
  else {  
    println(s" Visiting: $n $url")  
    val page = get_page(url)  
    for (u <- get_all_URLs(page))  
      crawl(u, n - 1)  
  }  
}
```

# Email Harvester

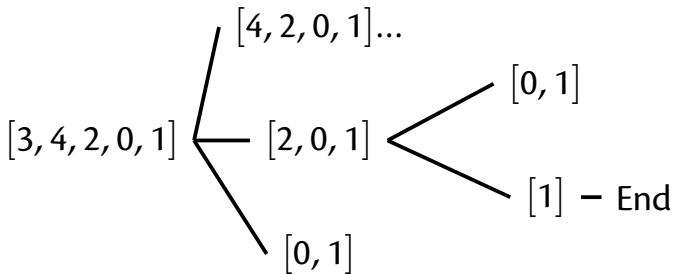
```
def emails(url: String, n: Int) : Set[String] = {  
  if (n == 0) Set()  
  else {  
    println(s" Visiting: $n $url")  
    val page = get_page(url)  
    val new_emails =  
      email_pattern.findAllIn(page).toSet  
    new_emails ++  
      (for (u <- get_all_URLs(page))  
        yield emails(u, n - 1)).flatten  
  }  
}
```

# Jumping Towers



shortest: 3 → 4 → End

# “Children” / moves



# Reverse Polish Notation

$$(3 + 1) * (2 + 9)$$

$\Rightarrow$

$$3 \ 1 \ + \ 2 \ 9 \ + \ *$$

# Sudoku

A very simple-minded version on 110 problems:

- 1 core: 800 secs
- 2 cores: 400 secs
- 8 cores: 290 secs
- 18 cores: 142 secs