Today’s music: *Blank Space* by Taylor Swift

I could show you incredible things // Magic, madness, heaven, sin
So it’s gonna be forever // Or it’s gonna go down in flames //
You can tell me when it’s over // If the high was worth the pain
The AI Quiz is due tonight. Have you submitted?

A. Yes
B. No
C. No, but only because I plan to drop the class in the next 12 hours

You will be unable to submit A0 if you do not submit the AI quiz by the deadline.
Queue Me In
Office Hours. Simplified.
Textbook vs. lecture

CS 3110

PRIMARY

summary
Search

- Website
- Textbook
- Piazza
Section & lecture attendance

about 1% each: incentive not penalty
Review

Previously in 3110:
• Expressions, definitions
• Scope
• Functions

Today:
• Data types for collections: lists, records, tuples
• Pattern matching
# Collections

<table>
<thead>
<tr>
<th></th>
<th>Homogenous elements</th>
<th>Heterogenous elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed number of elements</td>
<td>Lists</td>
<td>Tuples and records</td>
</tr>
<tr>
<td>Unbounded number of elements</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LISTS
List implementation

• **Immutable:** can't change elements
• **Singly-linked:**
  – Good for sequential access of short-to-medium length lists (say, up to 10k)
  – Data structures are tools: none is perfect
• **Terminology:** nil and cons (from Lisp)
• Next lecture: we'll see the implementation
Question

What is the type of `31 : : [10]`?
A. int
B. int list
C. int int list
D. int list list
E. Not well-typed
Question

What is the type of $31::[10]$?

A. int

B. int list

C. int int list

D. int list list

E. Not well-typed
RECORDS
TUPLES
Records by name vs. Tuples by position
Records and tuples

- New kind of definition: type definition
- New kinds of types: record types, tuple types
# Collections

<table>
<thead>
<tr>
<th></th>
<th>Homogenous elements</th>
<th>Heterogenous elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed number of elements</td>
<td></td>
<td>Tuples and records</td>
</tr>
<tr>
<td>Unbounded number of elements</td>
<td>Lists</td>
<td></td>
</tr>
</tbody>
</table>

Syntax and semantics: all in the textbook
Question

You want to represent a camel, which has some number of humps and some number of riders. What do you use?

A. Tuple
B. Record
C. List
PATTERN MATCHING
Deconstructing data

Data:

Patterns:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Andrew&quot;</td>
<td>&quot;CS&quot;</td>
</tr>
<tr>
<td>3.0</td>
<td>false</td>
</tr>
</tbody>
</table>
Deconstructing data

Pattern does not match data
Deconstructing data

"Andrew"   "CS"

3.0       false

Pattern does match data
Extracts gpa
PATTERN MATCHING ON LISTS
Pattern matching

- Match shape of data
- Extract part(s) of data

Syntax:

\[
\text{match } e \text{ with} \\
| \ p1 \rightarrow e1 \\
| \ p2 \rightarrow e2 \\
| \ ... \\
| \ pn \rightarrow en
\]

\(p1..pn:\) pattern expressions
Semantics of pattern matching

• [ ] matches [ ] and nothing else

• h :: t
  – matches 2 :: [ ], binding h to 2 and t to [ ]
  – matches 1 :: 3 :: [ ], binding h to 1 and t to 3 :: [ ]

• _ matches everything
  underscore character, called wildcard
  (it’s like a blank space)

Full details in textbook
Question

```plaintext
match ["taylor"; "swift"] with
| []  -> "1989"
| h :: t  -> h
```

To what value does the above expression evaluate?
A. “taylor”
B. “swift”
C. “1989”
D. []
E. h
match ["taylor"; "swift"] with
| []  ->  "1989"
| h :: t  ->  h

To what value does the above expression evaluate?
A. “taylor”
B. “swift”
C. “1989”
D. []
E. h
Why pattern matching is INCREDIBLE

1. You can’t forget a case
   (inexhaustive pattern-match warning)

2. You can’t duplicate a case
   (unused match case warning)

3. You can’t get an exception
   (e.g., \texttt{hd \[ \]})

4. Pattern matching leads to elegant, concise, beautiful code
Upcoming events

• [Wed] A0 due
  Advice: you’ll be very sad later in the semester if you use any late days on A0

• [Thu] A1 out

This is incredible.

THIS IS 3110