## csci54 - discrete math \& functional programming tuples and lists

## Recap

- Write a function cap' that not only caps the upper limit at 100, but additionally evaluates to 0 if $n$ is less then or equal to 0 .
- Write a function pow that takes two parameters n and k and returns n to the kth power. (assume that $k$ is guaranteed to be a nonnegative integer. do not use

```
cap n =
    if n > 100
    then 100
    else n
```

```
cap' n
```

cap' n
if n < 0
if n < 0
then 0
then 0
else (cap n)

```
    else (cap n)
```

        then 100
        else
        if \(n<0\)
        then 0
        else n
    ```
pow n k =
    if k == 0
    then 1
    else n * (pow n (k-1))
```

the ** operator)

## maxInt

- write a function maxInt that takes a list of integers and returns the value of the largest element. you may assume the list is not empty.

$$
\begin{aligned}
& \operatorname{maxInt}[x]=x \\
& \operatorname{maxInt}(x: x s)=\max x \text { (maxInt xs) }
\end{aligned}
$$

## Lists in Haskell

- Homogenous (all same type)
- square brackets with element separated by commas
- building lists
- square brackets with values separated by commas

$$
\text { ghci> aList }=[1,10,-3,5]
$$

- cons
ghci> aList2 $=2$ : [1, 10, $-3,5]$
- concatenation
ghci> aList3 = aList ++ aList2


## Lists in Haskell continued

- functions on lists
- head, tail
- init, last

$$
\text { aList }=[2,1,10,-3,5]
$$

- take, drop
- length, null
- reverse
- ...
-     - $l e m ` ~ v s ~ e l e m ~$
- infix vs. prefix

$$
\begin{aligned}
& \text { elem } 1[2,1,10,-3,5] \\
& 1 \text { 'elem } \quad[2,1,10,-3,5]
\end{aligned}
$$

- same with arithmetic functions: div, mod
- div: round down
- mod: integer mod (goes with div)
(Haskell also has quot, rem, which behave differently than div/mod with negative numbers)


## Practice problems

- what does this function do?

$$
\begin{aligned}
& \text { numList } \mathrm{n}= \\
& \text { if } \mathrm{n}<=0 \\
& \text { then }[] \\
& \text { else } \\
& \mathrm{n}: \text { (numList }(\mathrm{n}-1) \text { ) }
\end{aligned}
$$

## Practice problems

- (on week01-ps) numList $n$ evaluates to a list of integers from $n$ down to 1

```
numList n =
    if n <= 0
    then []
    else
        n : (numList (n-1))
```

- Write a function oddList where oddList n evaluates to a list of odd integers from $n$ down to 1 . If $n<1$ the function should return an empty list.
- Write a function oddList' where oddList' evaluates to a list of odd integers from 1 up to, but possibly not including, n . If $\mathrm{n}<$ 1 the function should return an empty list. Do not use the reverse function.
- what does the following do?

$$
\begin{aligned}
& \text { aList }=[2,1,10,-3,5] \\
& \text { aList }=2: \text { aList }
\end{aligned}
$$

## List comprehensions (and ranges)

- A way to build up lists:

$$
\begin{array}{|l|l|}
\hline\left[x^{*} 2\right. & x<-[1 . .3] \\
\hline
\end{array}
$$

- Note use of ranges in Haskell

$$
\left.\begin{array}{|llll|}
{[ } & 1,3 . . & 10 & ] \\
{[ } & 10,9 . . & 1
\end{array}\right]
$$

$$
\begin{array}{llll}
{\left[\begin{array}{lll}
{[ } & 1, & 4
\end{array}\right]} \\
{[ } & 47 & . & ]
\end{array}
$$

- Can add more to list comprehensions:

$$
\left[x^{*} y \mid x<-[1 . .3], y<-[6,4,2]\right]
$$

$$
\left[x^{*} y \mid y<-[6,4,2], x<-[1 . .3]\right]
$$

## More on list comprehensions

- Can add predicates:

$$
\left[x^{*} y \mid x<-[1 . .3], y<-[1 . .3], x>y\right]
$$

- Can use any expression:
[ if $\mathrm{x} * \mathrm{y}>3$ then "BIG" else "SMALL" | $\mathrm{x}<-$ [1..3], y <- [1..3]]
[ (x,y) | x <- ['a'..'c'], y <- ["rat","ox","tiger"]]
* a tuple does not need to be homogenous. cannot append or concatenate, so must know number of elements from start


## Practice problems

- Write a function oddList where oddList n evaluates to a list of odd integers from n down to 1 . If $\mathrm{n}<1$ the function should return an empty list.
- Write a function oddList' where oddList' evaluates to a list of odd integers from 1 up to, but possibly not including, n. If $n<$ 1 the function should return an empty list
- Rewrite oddList and oddList' using list comprehensions
- What do these evaluate to?

$$
\begin{aligned}
& {\left[\text { if } x^{*} y>3 \text { then [1] else [2] } \mid x<-[1 . .3], y<-[1 . .3]\right]} \\
& {[(x, y, z) \mid x<-[1.3], y<-[1 . .3], z<-[1.3], x<y, y<z]} \\
& {[(x, y, z) \mid z<-[1 . .3], y<-[1 . .3], x<-[1 . .3], x<y, y<z]}
\end{aligned}
$$

