## SYNTAX EEN PHUNKSIONS

lucky :: (Integral a) => a -> String

lucky 2 = "sorry, you are out of luck, pal!"

SayMe :: (Integral a) > a → String
sayMe 1 = "One!"

Say Me 2 = "Two!"

sagme 3 = "Three!"

say Me 4 = "Four!"

Say Me 5 = "Five!"

sagme 2 = "Not between I and 5"

factorial:: (Integral a) => a -> a
factorial 0 = 1

bacterial n = n \* factorial (n-1)

char Name : Char -> String Char Name 'a' = "Albert"

charName 'b' = "Broseph"

charName 'c' = "Cecil"

2 char Name al

"Albert"

2 charName 'b'

"Braseph"

2 charName 'h' X

add vectors :: (Numa) => (a, a) -> (a,a) -> (a,a) add Vectors a b = (fst a + fst b, snd a + snd b)

add vectors:: (Numa) => (a,a) -> (a,a) -> (a,a)

add Vectors (x1, y1) (x2, y2) = (x1+x2, y1+y2)

first :: (a, b, c) -> a

first (2, \_ \_) = 2

second :: (a, b, c) -> b second (=, y, =) = y ...

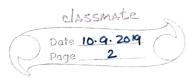
third: (a, b, c) -> c third (-, a, 42) = Z

2 let xs = [(1,3), (4,3), (2,4), (5,3) (5,6) (3,1)] 2 [a+b | (a,b) <- xs]

[4,7,6,8,11,4] [1,2,3] = 1:2:3:[] tuse for pattern matching

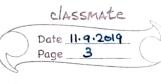
n: 25 - pattern matching tred head, tail. x: 4: 2: 25

head :: [a] -> a head [] = error "Can't call head on an empty list, dommy!" head (x: \_) = x



2 head [4, 5, 6] 2 head "Hello" 'H! tell: (show a) => [a] -> String tell [] = "The list is empty" tell (x: []) = "The list has one element:" ++ . Show 2 tell (n:y:[]) = "The list has two elements:" ++ Show x [yy] ++ "and" ++ show y tell (2: y: ) = "The list is long. The first two elements are:" ++ show x ++ show y length :: (Numb) => [a] -> b length [] = 0 length' (-: xs) = 1 + length'xs Sum' : (Noma) => [a] > a sum' [] = 0 sum (x: 28) = x + sum 25@(x: y: ys) but thing (no need to represent x: y: ys) (xs ++ ys) X (xs++ [x, y, z]) x

(xs ++ [x]) X



capital :: String -> String Capital " = " Empty string whoops!" capital all@(x:xs) = "The first letter of" ++ all ++ "is"++[z] 2 capital "bracula" "The first letter of Dracula is D" bmiTell :: (RealFloat a) > a > String bmitell bmi 1 bmi <= 18.5 = "You are underweight, you emo, you! | bmi <= 25.0 = "You are supposedly normal, Pffft, i bet" 1 bmi <= 30.0 = "You're fat! Lose some weight, batty!" 1 otherwise = "You're a whale, compratulations!" otherwise = True no guard matches => fall through to next pattern λ bmi Tell 85 1.90 "You're supposedly normal. Potot, I bet gov're ugly!" max' :: (ord a) => a -> a -> a max! a b 1 a>b = a 1 otherwise = b my Compare: (orda) => a -> a -> Ordering a my Compare b larb = GT 1 a == b

= EQ

1 otherwise = LT

gat 30.0	= 25.0 \ align	where bmi = weight / height 12	You've a whale	1.	= "You're supposedly no	1 bmi <= skinny = Vou're underweight you	omitell weight height	bmitell: (RealFloot a) => a -> a -> String	i = welaht / heig	1 otherwise = "You're a whale"	1 bmi <= 25.0 = "You're supposedly normal"	1 bmi <= 185 = You're underweight, you	omitell weight height	bmitell: (Real Float a) => a -> a -> String	TOUTE OF SE STATE	# P V V V V V V V V V V V V V V V V V V	2 <= 30.0 = "You're fait! Lose	2 <= 25.0 = "You're supposedly r	1 weight / height > 2 == 18.5 = "You're underweight, you"	broiTell weight height	bmitell: (RealFloat a) => a -> a -> String	GT.	2 3 my Compare 2	Page 7

Date 11.9.2014
Page 5 Classmate

where initials initials where (skinny, pag. first name String (1:-Cf: -) 11 1 weight normal 11: 11 String lastrame = lastname firstname / height bat -String 11 ++[4]++ (18.5 = Pattern match = Ü 0 30.0 =

calc Bmis cale Bmis where Z, bmi = weight height = (Real Float a) )) Lbm: wh <u>"</u> 6 (w, h) <weight a) for [a] xs height ?

let where 5 bindings 1000 expression 85 also and nested. it can used any where

where

applies

7

to

entire

function body

Cylinder (RealFloata) 15 0 1

Partern matching 0/150 Supports cylinder let 3 side Area top Avea sideAvea 5 11 + 11 11 2\* 01 \* N D × topAvco 3 と 5 CXPYESSION

4 5 > 3 then 11 Woo 11 else 11 Boo 11

بح = Wor 4 \* Barll â 0 then ~ S the Foo = 0 else else "Bar" ] 0 + N

2

1000

	empty lists!"					emote eight!"   creactly	Similar
head :: [a] > a	head [] = error "No nead her empty lists"	head (x; -) = x		head' :: [a] -> a.	head his = case is not	[] -> error "No head for empty with "	$(\kappa, .) \rightarrow \kappa$
			,				

2

cohat

"The lost is

Ş

describe List

"empty"

what El

cohere

-> String

[0]

describe List

Cest "

"a singleton

1

[7]

"empto"

8

"The list

lr

describelist 25

describe List:

Prist."

"a longer !

STOR

⋾

"a singleton list.

cohat [x] =

lest "

lenger

≈ٍ م

what xs