CSE 341 Section Handout #1
Cheat Sheet

Types

`int, real, bool, string, char, tuple, list, function`

Constructs

`val` **variableName** `=` **expression**`;`
`fun` **name**`(`**name** `:` **type**`, ..., `**name** `:` **type**`)` `=` **expression**`;`
`if` **booleanExpression** `then` **expression** `else` **expression**

Operators

operator	example	description
+	2 + 2	addition
-	18.4 - 3.8	subtraction
*	3 * 18	multiplication
/	14.5 / 3.4	real division
div	88 div 10	integer (truncated) division
mod	88 mod 10	modulus function
~	~18	arithmetic negation
^	"hello" ^ "there"	string concatenation
::	3 :: [1, 2, 3]	construct a list from head/tail
@	[1, 2] @ [3, 4]	append two lists together
<	4 < 19	less than
>	2.4 > 3.8	greater than
<=	"ab" <= "cd"	less than or equal
>=	1.7 >= 4.7	greater than or equal
=	[1, 2] = [1, 1+1]	equals
<>	3 <> 4	not equal
andalso	2 + 2 = 4 andalso 3 < 4	logical and (short circuited)
orelse	8 > 9 orelse 1.4 < 2.8	logical or (short circuited)
not	not (8 < 9)	logical negation

Built-in Functions

Function	Description
abs	absolute value of int or real
real	convert int to real
floor/ceil	integer just lower or higher than a real
trunc/round	convert real to int
hd	head (first element) of a list
tl	tail (elements after first) of a list
length	length of a list

CSE 341 Section Handout #1
Questions

1. What is the type of each of the following expressions?

```
(3, 4.5)
[4.5, 3.8]
((3, 4), 18.5)
[[3, 4], [7]]
[("hello", 3)]
```

2. For each of the following types, give an expression of that type:

```
int * string * real
string list
int * string list
(int * string) list
(int * string) list list
(int * string list) list
```

3. Write a function called `gcd` that returns the greatest common divisor (GCD) of two integers. You can take advantage of Euclid's formula, which states that the GCD of any number *n* and 0 is *n*, and the GCD of any two non-zero integers *x* and *y* is equal to the GCD of *y* and (*x* mod *y*).

Using your `gcd` function, write a function `lcm` that returns the least common multiple of two integers.

4. Write a function called `grade` that takes a real number representing a percentage as an argument and that returns "A" for percents that are 90 and above, "B" for percents 80 to 90, "C" for 70 to 80, "D" for 60 to 70, and "F" for percents below 60.

5. Write a function called `sum` that returns the sum of a list of integers.
(How would you change it to make it sum a list of reals?)

6. Write a function called `toReal` that takes a list of `int`s as an argument and that returns the result of converting each `int` to a `real`. For example, `toReal([3, 5, ~7])` should return `[3.0, 5.0, ~7.0]`.

7. Write a function called `switchPairs` that takes a list as an argument and that returns the list formed by switching the order of successive pairs of elements in the list. For example, the call of `switchPairs([3, 7, 4, 9, 8, 12])` should return `[7, 3, 9, 4, 12, 8]`.

CSE 341 Section Handout #1
Solutions

1.

Expression	Type
`(3, 4.5)`	`int * real`
`[4.5, 3.8]`	`real list`
`((3, 4), 18.5)`	`(int * int) * real`
`[[3, 4], [7]]`	`int list list`
`[("hello", 3)]`	`(string * int) list`

2.

Type	Value
`int * string * real`	`(3, "hello", 4.5)`
`string list`	`["abc", "def", "gh"]`
`int * string list`	`(38, ["ab", "cd"])`
`(int * string) list`	`[(38, "ab"), (19, "cd")]`
`(int * string) list list`	`[[(38, "ab")], [(19, "cd")]]`
`(int * string list) list`	`[(38, ["ab", "cd"]), (7, ["foo"])]`

3.

```
fun gcd(x, y) =
    if y = 0 then x
    else gcd(y, x mod y)

fun lcm(x, y) = x * y div gcd(x, y)
```

4.

```
fun grade(pct) =
    if pct >= 90.0 then "A"
    else if pct >= 80.0 then "B"
    else if pct >= 70.0 then "C"
    else if pct >= 60.0 then "D"
    else "F";
```

5.

```
fun sum(lst) =
    if lst = [] then 0
    else hd(lst) + sum(tl(lst))
```

6.

```
fun toReal(lst) =
    if lst = [] then []
    else real(hd(lst)) :: toReal(tl(lst))
```

7.

```
fun switchPairs(lst) =
    if length(lst) <= 1 then lst
    else hd(tl(lst)) :: hd(lst) :: switchPairs(tl(tl(lst)));
```