

ML – Functions

Syntax:

<def> ::= **fun** <function-name> <parameter> = <expression>
<function-name> ::= an appropriate function name
<parameter> ::= a variable name
 | a tuple of variable names
<expression> ::= any expression discussed so far including function calls

Example (default type int):

```
- fun add2 (a,b) = a + b;  
val add2 = fn : int * int -> int
```

└───┬───
tuple
└───┬───
function type

Example (real):

```
- fun add2r (a:real,b:real):real = a + b;  
val add2r = fn : real * real -> real
```

function return type
↓

ML – Functions

Using Functions:

```
- fun add2 (a,b) = a + b;  
val add2 = fn : int * int -> int  
- add2(1,2);  
val it = 3 : int
```

ML – Functions

Example: write a function that computes the negative of a real value.

```
- fun neg(v:real):real = v * ~1.0;  
val neg = fn : real -> real
```

ML – Functions

Example: write a function that computes the sum of the elements of a list of integers.

Cases:
[1,2,3,4,5] → 15
[6,12] → 18
[7] → 7
[] → 0

```
- fun listsum (L:int list) = if null(L) then 0  
  else hd(L) + listsum(tl(L));
```

↖ recursion!

ML – Functions

Example: write a function that computes the length of a list.

Cases:

[1,2,3] → 3

[1,2] → 2

[1] → 1

[] → 0

```
- fun listlength(l) = if null(l) then 0
  else 1 + listlength(tl(l));
```

↙ recursion!

ML – Functions

Example: write a function that will reverse the order of the elements in a list.

Cases:

[1,2,3] → [3,2,1]

[1] → [1]

[] → []

```
- fun reverse (L) = if null(L) then []
  else reverse(tl(L)) @ [hd(L)];
```

ML – Functions

- Programming Assignment #1
 - 5.2, 5.3, 5.4, 5.7
 - due Monday 2/20 in class
 - hand in your source code and a screen shot of your program running with some example data.