# SML (2): SYNTAX AND FUNCTIONS

Comp3031 Lab 1
Fall 2014

JIA Xiaoying, SU Zhiyang,

# Identifiers: Value Binding

- Values can be bound to names using values declarations.
- Value declarations are introduced by keyword val.
- Syntax: val <identifier > = <expression> ;
- Examples:

```
-val a = 2;
-a;
-3 * a + 4 * a * a;
-val c = 2 * a;
-val a = 3;
-a;
-let val a = 5 in a + 6 end;
-a;
```

***Remark: New binding will hide the old one***!

# Patterns

- Values can be bound to all the identifiers in a pattern. This is useful in defining functions.
- Examples:
  - val (fst,snd) = (4,4.45);
  - val (a,b,c) = (1,3,4);
  - val {a = x,b = y} = {a = 1.2, b = 2.3};
  - val (x,x) = (2,3); (* Error! *)

If you want to declare a tuple, you should declare it in form: val tupleA = (1,2);

# More Patterns

- val(x,_,y) = (1,2,3);
- val head :: _ = [1,2,3];
- val (x,y,_) :: _ = [(2,3,4),(5,6,7)];
- *Remark: the wildcard pattern "_"(underscore symbol) may be used for terms that you don't care in pattern matching.*

# More examples

- val helloworld = "Hello World!";
- val fib_6 = [1,1,2,3,5,8];
- val twice = (fn x => 2*x);
- val president = "JinPing"^"Xi";

# Let statement

**let … in … end;**

- Example1
  - let
  - val x = 3
  - val y = 5
  - in
  - x*x + 3*y
  - end;
  - val it = 24: int

- Example2
  - -val action

-let val h = "go home“

- val t = "go to tutorial"

-in if 13>12 then h else t

-end;

# Let & Fun

```
 1 fun add z =
 2     let val x = 1
 3         val y = 2
 4     in
 5       x + y + z
 6     end;
 7
 8 let val m = add 3
 9 in
10    m + 1
11 end;
~
```

# SML function

- It is “funny”
- Example:
  - fun square(x) = x*x; (* Calculate square of an int*)
- We can also define a function by enumerating ALL cases with pattern matching
  - fun say 1= “one”
    | say 2 = “two”
    | say 3 = “three”;

- Examples:
  - fun floatage(rho, v) =
    let val rWater = 1.0 val g = 10.0
    in if rho>rWater then rWater*v*g else rho*v*g
    end;
  - fun plusmul(x,y) =
    let val y=x+2 val x=y*2
    in x*y
    end;

# Recursive Functions

- fun fact(n) = if n = 0 then 1 else n * fact(n-1);
- fun gcd(n,m) = if m = 0 then n else gcd(m, n mod m)

# Examples

- Write the following SML functions:
  - len (L) = fn: ‘a list -> int. It returns the length of a list
  - *fun len([]) = 0 | len(head::tail) = 1+len(tail);*
  - exist(v, L) = fn: ‘‘a * ‘‘a list -> bool. It returns true if v is an element in L and false otherwise. The double-apostrophe stands for "all types that are comparable".
  - *fun exist(v,[]) = false |*

    *exist(v,h::t)= if h=v then true else exist(v,t);*

# Exercise

- indexOf(v, L) = fn : ''a * ''a list -> int. If v is an element in L, it returns the index (ranged from 0 to length(L)-1) of the first occurrence of v in L; otherwise, it returns -1.

- fib(n) = fn:int -> int. Returns the *n*th Fibonacci number. The first and second Fibonacci numbers are both 1. For *n*>2, the *n*th Fibonacci number is the sum of the *n-1*th and the *n-2*th.