ML

1. Types:
   1. Int
      1. ~ symbol is used for negative numers~1234 means minus
   2. Real
   3. Char #“a”
   4. String “abc”
   5. Bool
      1. True , false
   6. If \_\_ then else
      1. orelse
   7. &&
      1. andalso
   8. and
      1. Is used for mutual recursion
   9. Examples
      1. If 1<2 then #”a” else “bc”
         1. Error char with string
2. Tuples
   1. (val1, val2,…, valn)
   2. Product type 🡪 type1\*type2\*…\*typen
   3. Val pair = (2,3);
   4. Response Val pair = (2,3): int\*int

Let val x = 3+4

Y=5-2

In (x,y) end;

Val it = (7,3): int\*int

1. Clausal function expression

Fun pattern1 = expr1

|pattern2=expr2

.

.

.

|pattern=exprn

1. Fun Fibonaci 1,1,2,3,5,6

fun fib 0 =1

| fib 1 =1

| fib n = fib(n-1) + fib(n-2);

1. Mutual Recursion
   1. Extract the 1st, 3rd, 5th, …, nth odd elements from a list
      1. A1, A2, A3, A4,…, An
      2. Output: A1, A3,…,An

Fun take(L) =

If L=nil then nil

Else hd(L)::skip(tl(L))

//calls another function skip this is the mutual recursion call

//note can also used [hd(L)]@

And

Fun skip(L)=

If L =nil then nil

Else take(tl(L))

fun take(L)=

if L=nil then nil

else hd(L):: take(tl(tl(L)));

1. Increment Function

Fun inc (x)=x+1;

Val inc=fn x=>x+1

These are the same definition of the increment function

Use:

inc 4;

val it =5 : int

1. Square function

Val square=fn x=>x\*x

1. Higher-order function
   1. Compared to first order function
      1. Its arguments and result are all data, not functions
   2. Higher-order functions can pass a function to a function or return a function
   3. Each element we want to apply a function using a MAP

Fun Map(F,nil)=nil

|Map(F,x::xr)=F(x)::Map(F,xr);

What does this return?

How many params? 2

a’ list because nil, x::xr is a list

F is a function so it has another domain and range

With domain a’ or any element by f(x) b’ because we can not guarantee it is going to be a

Val Map = fn: a’🡪b’ \*a’ list 🡪 b’ list

fun