SML - PRACTICE EXERCISES

Exercise 1

Explain what is wrong in the following expressions and propose a correction.

```
hd([]);
explode (["toto"]);
implode("a","b");
["t] :: ["o","p"];
6 @ 10;
```

SOLUTION

hd([]);
uncaught exception Hd
explode ([”toto”]);
std_in:0.0-0.0 Error: operator and operand don’t agree (tycon mismatch)
operator domain: string
operand: string list
in expression:
explode (”toto” :: nil)
implode(”a”,”b”);
std_in:0.0-0.0 Error: operator and operand don’t agree (tycon mismatch)
operator domain: string list
operand: string * string
in expression:
implode (”a”,”b”)
[”t”] :: [”o”,”p”];
std_in:0.0-0.0 Error: operator and operand don’t agree (tycon mismatch)
operator domain: string list * string list list
operand: string list * string list
in expression:
:: (”t” :: nil,”o” :: ”p” :: nil)
6 @ 10;
std_in:0.0-0.0 Error: operator and operand don’t agree (tycon mismatch)
operator domain: ’Z list * ’Z list
operand: int * int
in expression:

@ (6,10)

Exercise 2

(3,4) and (3,4,5) have the same type? [3,4] and [3,4,5] have the same type?

SOLUTION

(3,4) and (3,4,5).
No.
- (3,4);
val it = (3,4) : int * int
- (3,4,5);
val it = (3,4,5) : int * int * int
[3,4] and [3,4,5].
Yes.
- [3,4];
val it = [3,4] : int list
- [3,4,5];
val it = [3,4,5] : int list

Exercise 3

Consider the following definitions:

```
fun fact n = if n=0 then 1
                    else n * fact(n-1);

fun new_if (a,b,c) = if a then b else c;
```

Write a function **new_fact** using **new_if**.
Explain why **new_fact** does not compute the factorial.

Note: What is the evaluation of recursive function in SML?

SOLUTION

fun fact n = if n=0 then 1 else n * fact(n-1);
val fact = fn : int → int
fun new_if (a,b,c) = if a then b else c;
val new_if = fn : bool * 'a * 'a → 'a
fun new_fact n = new_if(n=0,1,n*new_fact(n-1));
val new_fact = fn : int → int
Innermost occurrence of f (for SML).
f(x,y)=if x=0 then0 else f(x-1,f(x,y))

f(1,1)=>f(0,f(1,1))=>f(0,f(0,f(1,1)))=¿...

Outermost occurrence of f.

f(x,y)=if x=0 then 0 else f(x-1,f(x,y))

f(1,1)=>f(0,f(1,1))=0

Innermost evaluation does not always terminate.

Outermost evaluation does always terminate.

Innermost evaluation is more efficient than outermost evaluation (Convergence).

new_fact 2 = new_if(2=0,1,2*new_fact(1))

new_fact 2 => new_if(2=0,1,2*new_if(1=0,1,1*new_fact(0)))

new_fact 2 => new_if(2=0,1,2*new_if(1=0,1,1*new_if(0=0,1,0* new_fact(-1))))

new_fact 2 => new_if(2=0,1,2*new_if(1=0,1,1*new_if(0=0,1,0* new_if(-1=0,1,-1*new_fact(-2))))) ...

Exercise 4

Link the variable x to the value 0 when constructing forms to match the following expressions.

For example given the expression **(false,"bonjour",0)** the form (_,_,x) permits us to link x to 0 when we write: val (_,_,x) = (false,"bonjour",0).

```
{a=3,b=0,c=false} - record.
[~2,~1,0,1,2] - ~ unary minus.
[(3,1),(0,9)]
```

SOLUTION

val (_,_,x) = (false,"bonjour",0);

val x = 0 : int

a=3,b=0,c=false;

a=3,b=0,c=false is a record.

val it = a=3,b=0,c=false : a:int, b:int, c:bool

val a=_,b=x,c=_ = {a=3,b=0,c=false};

val x = 0 : int

[2, 1,0,1,2]

val _::_:: x :: _ = [2, 1,0,1,2];

std_in:23.1-23.33 Warning: binding not exhaustive

:: _ :: x :: _ = ...

val x = 0 : int

val _::_:: x :: _:: _ = [2, 1,0,1,2];

std_in:0.0-0.0 Warning: binding not exhaustive

_ ::_ :: x :: _ :: _ = ...

val x = 0 : int

val l1 :: x :: l2 = [2, 1,0,1,2];

std_in:25.1-25.33 Warning: binding not exhaustive l1 :: x :: l2 = ...

val l1 = 2 : int

val x = 1 : int

val l2 = [0,1,2] : int list

val [_,_,x,_,_] = [2, 1,0,1,2];

[(3,1),(0,9)]

Warning: binding not exhaustive

val _ :: (x,_) :: _ = [(3,1),(0,9)];

val [_,(x,_)] = [(3,1),(0,9)];

val _ :: [(x,_)] = [(3,1),(0,9)];

Exercise 5

Write a function **power_of_two** that tests if an int is a power of 2.

Write each steps of the evaluation of **(power_of_two 8)**.

SOLUTION

fun power_of_two n = (n=1) orelse ((n <> 0) andalso ((n mod 2) = 0 andalso (power_of_two (n div 2))));

exception error;

fun power_of_two 0 = raise error

| power_of_two n = (n=1) orelse ((n <> 0) andalso ((n mod 2) = 0 andalso (power_of_two (n div 2))));

val power_of_two = fn : int $\rightarrow$ bool

power_of_two 8 = (8=1) orelse ((8 mod 2 = 0 andalso (power_of_two (8 div 2)))

power_of_two 8 => (false) orelse ((0 = 0 andalso (power_of_two (4)))

power_of_two 8 => power_of_two (4)

power_of_two 8 => (4=1) orelse ((4 mod 2 = 0 andalso (power_of_two (4 div 2)))

power_of_two 8 => (false) orelse ((0 = 0 andalso (power_of_two (2)))

power_of_two 8 => power_of_two (2)

power_of_two 8 => (2=1) orelse ((2 mod 2 = 0 andalso (power_of_two (2 div 2)))

power_of_two 8 => (false) orelse ((0 = 0 andalso (power_of_two (1)))

power_of_two 8 => power_of_two (1)

power_of_two 8 => true

Exercise 6

What is the type of the following function , justify your answer.

```
fun f (x,y,z,t) =
    if x=y then z+1
           else if x > y then z else y+t;
```

SOLUTION

val f = fn : int * int * int * int → int

The entry tuple is of type T1 * T2 * T3 * T4.

The type of the result is T5.

The type of the function is T1 * T2 * T3 * T4 → T5.

z is of type int because of $z + 1$.

$y + t$ is of the same type as z and $z + 1$ is of type type int. The type of y and t is int. So T3=int and T4=int.

$z + 1$, z and $z + t$ are of type int so T5 = int.

From $x = y$ and y is an int we can deduce that y is an int so T1=int.

The type of f is int * int * int * int → int.

Exercise 7

Write 2 functions **odd** and **even** that define if an int is even or odd using mutual recursion.

SOLUTION

fun even 0 = true

| even n = odd(n-1)

and

odd 0 = false

| odd n = even(n-1);

val even = fn : int → bool

val odd = fn : int → bool

parentheses in odd(n-1) and even(n-1) are needed.

even 980;

val it = true : bool

odd 980;

val it = false : bool

Exercise 8

What are the results of the following declarations and expression. Each one is independent.

```
val x = 2 and y = x+1;
```

```
val x = 1; local val x = 2 in val y = x+1 end; val z = x + 1;
```

```
let val x = 1 in let val x = 2 and y = x in x + y end end ;
```

SOLUTION

val x = 2 and y = x+1;

std_in:2.19 Error: unbound variable or constructor: x

x and $x + 1$ are not linked. x was not defined previously.

let val x = 2 in x+1 end;
val it = 3 : int
- val x = 3;
val x = 3 : int
- val x = 2 and y = x+1;
val x = 2 : int
val y = 4 : int
val x* = 1;
local val x** = 2 in val y = x**+1 end;
val z = x* + 1;
val x = 1 : int
val y = 3 : int
val z = 2 : int
let val x* = 1 in let val x** = 2 and y = x* in x** + y end end ;
val it = 3 : int

Exercise 9

What are the results of the following expressions evaluations.

```
val x = 1 and y = 2 and z = 3;

let val x = x+1 and z=x+4 in x+z end;

let val t = x+1 in let val x = x+1 in x end end;
```

SOLUTION

val x* = 1 and y = 2 and z = 3;
let val x** = x*+1 and z***=x*+4 in x**+z*** end;
val x = 1 : int
val y = 2 : int
val z = 3 : int
val it = 7 : int
let val t = x+1 in (let val x = x+1 in x end) end;
val it = 2 : int

Exercise 10

Write a function **insert** that inserts an int in a (ascending) sorted list.

SOLUTION

fun insert (x:int) [] = [x]

```
| insert x (y :: l) = if x <= y then x :: y :: l
else y :: (insert x l);
val insert = fn : int → int list → int list
insert 4 [1,3,6,8];
val it = [1,3,4,6,8] : int list
```

Exercise 11

Write a function **merge** that mergees 2 lists of (ascending) sorted int.

SOLUTION

```
fun merge [] x = x
| merge x [] = x
| merge (l1 as ((h1:int) :: tl1)) (l2 as (h2 :: tl2)) =
if h1 <= h2
then h1 :: (merge tl1 l2)
else h2 :: (merge l1 tl2);
val merge = fn : int list → int list → int list
merge [1,3,5,100,200] [1,4,99,101,205,250];
val it = [1,1,3,4,5,99,100,101,200,205,250] : int list
```

Exercise 12

Write a function **insertion_sort** that implements insertion sorting.

SOLUTION

```
fun insertion_sort [] = []
| insertion_sort ((x:int) :: l) =
let fun insert x [] = [x]
| insert x (y :: l) = if x <= y then x :: y :: l
else y :: (insert x l)
in insert x (insertion_sort l)
end;
val insertion_sort = fn : int list → int list
insertion_sort [4,5,1,9,5,2,10,1];
val it = [1,1,2,4,5,5,9,10] : int list
```

Exercise 13

Bubble sort

1. Define a function **iteration** that repeat the treatment of a data while a condition on this data is not true.

2. Define a function **is_sorted** that returns true if a list is sorted, false otherwise.
3. Write a function **buble** that implements the Bubble sort.

SOLUTION

Principles :

Go through a list and each time you find two elements that are not in the right order you exchange them.

Iterative Algorithm:

procedure bubles-sort(var t : array[1...n] of integer);

var i, j : integer;

begin

i=1;

while i<n begin

for j=n downto i+1 do

if t[j]<t[j-1] then t[j] $\leftrightarrow$ t[j-1];

i=i+1

end

end

Complexity :

Let n be the number of elements of the considered list. The number of comparisons is $O(n^2)$. The number of exchanges is in the worst case $O(n^2)$.

Example :

101 11 5 30 63 47 20

11 101 5 30 63 47 20

11 5 101 30 63 47 20

11 5 30 101 63 47 20

11 5 30 63 101 47 20

11 5 30 63 47 101 20

11 5 30 63 47 20 101

5 11 30 63 47 20 101

5 11 30 47 63 20 101

5 11 30 47 20 63 101

5 11 30 20 47 63 101

5 11 20 30 47 63 101

Solution :

– fun iteration p suc l = if (p l) then l else iteration p suc (suc(l));

val iteration = fn : ('a → bool) → ('a → 'a) → 'a → 'a

– fun buble ([]) = []

| buble ([a]) = [a]

| buble (x::y::l) = if (x:int) <= y then x::buble(y::l) else y::buble(x::l);

val buble = fn : int list int list

– fun is_sorted [] = true

| is_sorted [a:int] = true

```
| is_sorted (x::l) = x <= hd(l) andalso is_sorted(l);
val is_sorted = fn : int list → bool
```

Execution :

```
is_sorted [2,1];
val it = false : bool
iteration;
val it = fn : ('a → bool) → ('a → 'a) → 'a → 'a
iteration is_sorted;
val it = fn : (int list → int list) → int list → int list
iteration is_sorted buble;
val it = fn : int list → int list
iteration is_sorted buble [2,1];
val it = [1,2] : int list
iteration is_sorted buble [1,2];
val it = [1,2] : int list
iteration is_sorted buble [2,1,4,3];
val it = [1,2,3,4] : int list
iteration is_sorted buble [101,11,5,30,63,47,20];
val it = [5,11,20,30,47,63,101] : int list
```

Exercise 14

Write a function that computes the subsets of a set. How to represent a set?

SOLUTION

```
fun cons h t = h :: t;
fun subset_list [] = [[]]
| subset_list (h :: t) =
let val ept = subset_list t
in ept @ (map (cons h) ept)
end;
val subset_list = fn : 'a list → 'a list list
subset_list [1,2,3];
val it = [[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]] : int list list
```

Exercise 15

What is the type of C:

```
fun C f g x = f (g x);
```

SOLUTION

val C = fn : ('a → 'b) → ('c → 'a) → 'c → 'b

type of C: type of f → type of g → type of x → type of f (g x)

type of g: type of x → type of g x

type of x: 'c

type of g: 'c → type of g x

type of g x: 'a

type of C : type of f → ('c → 'a) → 'c → type of f (g x)

type of f (g x): 'b

type of C: type of f → ('c → 'a) → 'c → 'b

type of f: type of g x → type f (g x)

type of f: 'a → 'b

type of C: ('a → 'b) → ('c → 'a) → 'c → 'b

Exercise 16

1. Write a function F that takes 2 parameters: a function O and a list l and processes the following way:

   $F(O, l) = O(a_1, O(a_2, O(a_3, ..., O(a_{n-1}, a_n)...)$ où $l = [a_1, ..., a_n]$.

   The list l has 2 or more elements.

   SOLUTION

   ```
   exception ErrorF;
   fun F (O,e1::e2::nil) = O(e1,e2)
   | F (O,e1::nil) = raise ErrorF
   | F (O,nil) = raise ErrorF
   | F (O,e1::l) = O(e1,F(O,l));
   val F = fn : ('a * 'a → 'a) * 'a list → 'a
   example:
   fun f (n,m) = n+ m +1;
   val f = fn : int * int → int
   F (f,[1,2,3]);
   val it = 8 : int
   ```

2. Write a function G that returns the elements of a list l that satisfy the condition *cond*. What is the type of G? Why?

   SOLUTION

   ```
   fun G cond nil = nil
   | G cond (e::l) = if (cond e) then e::(G cond l) else G cond l;
   val G = fn : ('a → bool) → 'a list → 'a list
   fun cond ch = ch > 3;
   ```

val cond = fn : int → bool

G cond [1,2,5,6,0,5,6,2];

val it = [5,6,5,6] : int list

val rec map = fn f => fn l =>

if (null(l)) then nil

else f(hd(l))::map f (tl(l));

val map = fn : ('a → 'b) → 'a list → 'b list

fun G cond l = map cond l;

G cond [1,2,5,6,0,5,6,2];

val it = [false,false,true,true,false,true,true,false] : bool list

3. Using F and the function max that returns the maximum of 2 integers (write max) what is the maximum of a list of int, for example $[2, 6, 3, 15, 18, 1, 55, 22]$)?

SOLUTION

fun max (n:int,m) = if n >= m then n else m;

val max = fn : int * int → int

F (max,[2,6,3,15,18,1,55,22]);

val it = 55 : int

4. Using F and the function *conc* that returns the concatenation of 2 strings (write conc) what is the concatenation of all the strings of a list, for example $[”a”, ”b”, ”c”, ”d”]$?

SOLUTION

fun conc (ch1,ch2) = ch1 ĉh2;

val conc = fn : string * string → string

F (conc, [”a”,”b”,”c”,”d”]);

val it = ”abcd” : string

5. Consider the function $fold$. What is its type?

```
fun fold F nil y = y
| fold F (x::l) y = F(x,(fold F l y));
```

SOLUTION

The type of $fold$ is of the form $T_F \rightarrow T_{nil} \rightarrow T_y \rightarrow T_y$ using the first equality and $T_F \rightarrow T_{x::l} \rightarrow T_y \rightarrow T_{F(x,(foldFly))}$ using the second equality.

So we have: $T_y = T_{F(x,(foldFly))}$.

Let $T_y =' b$.

The type of $fold$ is: $T_F \rightarrow T_{nil} \rightarrow' b \rightarrow' b$ or $T_F \rightarrow T_{x::l} \rightarrow' b \rightarrow' b$.

Let 'a be the type of x. $x :: l$ is a list of type 'a list. With respect to the definition of fold, nothing permits us to say that 'a = 'b.

The type of $fold$ is: $T_F \rightarrow' alist \rightarrow' b \rightarrow' b$ or $T_F \rightarrow' alist \rightarrow' b \rightarrow' b$.

From $F(x,(foldFly))$ we deduce that F is a function with one parameter that is tuple.

$T_{F(x,(foldFly))} = T_y =' b$.

T_F is of type $T_x * T_{foldFly} \rightarrow T_{F(x,(foldFly))}$. So T_F is of type $'a *' b \rightarrow' b$.

Note: $T_{foldFly}$ is of type T_y i.e. 'b.

So:

val fold = fn : $('a *' b \rightarrow' b) \rightarrow' alist \rightarrow' b \rightarrow' b$

Exercise 17

Consider the function f:

```
fun f (x,nil) = nil
| f (x,a::aa) = if x(a) then a::f(x,aa) else f(x,aa);
```

Let T_e be the type of an expression e. We construct using f the following system of equations.

```
(1) T_f = 'a -> 'b
(2) 'a = T_x * T_{nil}
(3) 'a = T_x * T_{a::aa}
(4) 'b = T_{nil}
(5) 'b = T_{f(x,aa)}
(6) T_{x(a)} = bool
```

1. Justify each line of this system of equations.
2. Compute the type of f.

SOLUTION

1.

(1) $T_f =' a \rightarrow' b$

f is a function.

(2) $'a = T_x * T_{nil_g}$

The first parameter of f is a tuple of type 'a (1). $'a = T_x * T_{nil_g}$ is obtained from the first equality.

(3) $'a = T_x * T_{a::aa}$

The first parameter of f is a tuple of type 'a (1). $'a = T_x * T_{a::aa}$ is obtained from the second equality.

(4) $'b = T_{nil_d}$

From the first equality we deduce that the type of the result of the function f is of the same type as nil and from (1) it is 'b, $'b = T_{nil_d}$.

(5) $'b = T_{f(x,aa)}$

From the second equality the type of the result of the function f is of the same type as $f(x,aa)$ and from (1) we deduce that it is 'b, $'b = T_{f(x,aa)}$.

(6) $T_{x(a)} = bool$

$x(a)$ is after an if, so $T_{x(a)} = bool$.

2. The 'b is a list. Let 'b = 'c list. The type of f is $'a \rightarrow' b$. By replacements we obtain:

$T_x * T_{nil_g} \rightarrow' b$

$T_x * T_{a::aa} \rightarrow' b$

The type of $T_{x(a)}$ is bool. a is of type 'c because we have: $a :: f(x, aa)$ as result of f that is of type 'b so 'c list. T_x is of type $'c \rightarrow bool$ and $T_{a::aa}$ is of type 'c list.

The type of f is $('c \rightarrow bool) *' clist \rightarrow' clist$.

Exercise 18

Let consider f:

```
fun f (x,nil) = nil
| f (x,a::aa) = if x(a) then x(a)::f(x,aa) else f(x,aa);
```

Let T_e be the type of an expression e. We construct a set of equations from the definition of f.

Write this system and compute the type of f.

SOLUTION

(1) $T_f =' a \rightarrow' b$

f is a function.

(2) $'a = T_x * T_{nil_g}$

The first parameter of f is a tuple of type 'a (1). $'a = T_x * T_{nil_g}$ is obtained from the first equality.

(3) $'a = T_x * T_{a::aa}$

The first parameter of f is a tuple of type 'a (1). $'a = T_x * T_{a::aa}$ is obtained from the second equality..

(4) $'b = T_{nil_d}$

From the first equality the type of the result of f is of the same type as nil. From (1) it is 'b, $'b = T_{nil_d}$.

(5) $'b = T_{f(x,aa)}$

From the second equality the type of the result of f is of the same type as $f(x, aa)$. From (1) it is 'b, $'b = T_{f(x,aa)}$.

(6) $T_{x(a)} = bool$

$x(a)$ is after an if, so $T_{x(a)} = bool$.

2. The type 'b is a list. Let 'b = 'c list. $'b = T_{f(x,aa)}$ and in the definition of f we have: $x(a) :: f(x, aa)$ and $T_{x(a)} = bool$ so 'c = bool and 'b = bool list.

The type of f is $'a \rightarrow' b$. By replacements we obtain:

$T_x * T_{nil_g} \rightarrow' b$

$T_x * T_{a::aa} \rightarrow' b$

$T_x * T_{a::aa} \rightarrow' clist$

The type of $T_{x(a)}$ is bool. Let 'd be the type a. 'd is not linked with 'a, 'b or 'c. The type of x is $'d \rightarrow bool$. $T_{a::aa}$ is of type 'd list.

The type of f is $('d \rightarrow bool) *' dlist \rightarrow' boollist$.

Exercise 19

1. Write a *datatype* **COORDS** that defines the coordinates of a point in 3D.
2. Give examples of the use of COORDS.
3. Using **COORDS** write a function *distance* that computes the distance between 2 points.
4. Give an examples of the use of distance.

SOLUTION

1. datatype COORDS = Cords of real * real * real;

   datatype COORDS

   con Cords : real * real * real − > COORDS

2. val point1 = Cords(1.2, 3.4, 0.0);

   val point1 = Cords (1.2,3.4,0.0) : COORDS

3. fun square(x:real) = x*x;

   val square = fn : real − > real

   fun distance (Cords(x1,y1,z1): COORDS) (Cords(x2,y2,z2):COORDS) =

   sqrt(square(x2-x1) + square(y2-y1)+square(z2-z1));

   Note: We have to write COORDS in the definition.

   val distance = fn : COORDS − > COORDS − > real

4. distance (Cords(1.2, 2.6, 4.5)) (Cords(2.5, 5.5, 7.5));

   Note: We have to put the parentheses.

   val it = 4.37035467668243 : real

Exercise 20

Create a datatype PERSON that defines a person defined by its name, fname, age and datebirth.

SOLUTION

datatype PERSON = P of {NAME : string, FNAME : string, AGE : int, DBIRTH : int};

P {NAME = "Dupond", FNAME = "Alice", AGE = 10, DBIRTH = 101072};

val it = P AGE=10,DBIRTH=101072,NAME="Dupond",FNAME="Alice" : PERSON

Exercise 21

1. Create a reference variable i whose value is a reference to 10.
2. Increment the value of i of 1.
3. Decrement the value of i of 1.
4. Change the value of i to 20.

SOLUTION

1. val i = ref 10;
2. inc(i);
3. dec(i);
4. i := 20;

Exercise 22

while $< expression >$ do $< expression >$ has the following semantics in SML:

a. Evaluate the first expression.

b. If the first expression is false, exit. Else, evaluate the second expression and go to step a.

We want to code the following algorithm in SML:

```
i = 1
while i <= 10
do
afficher i
i= i+1
end
```

1. Why is the use of references needed?
2. Write the SML code.

SOLUTION

1. They give an imperative view to programming in functional language. A reference contains a value that can change. This is what we need in a while statement.

   In the given code the evaluation of the second expression change the value of the first expression, this is why the utilization of references permits us to simplify programming.

2. $val\ i\ =\ ref\ 1;$

   $while\ !i\ <=\ 10\ do\ (print(!i);\ print("\ ");inc(i));$

   $1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ val\ it\ =\ ()\ :\ unit$

Exercise 23

Write a function *factorial* that returns:

– 1 for factorial(0)

– generates an exception for a negative parameter and return 0

– n! for a strictly positive parameter.

SOLUTION

```
exception Negative of int;
exception Negative of int
fun factorial1(0) = 1
| factorial1(n) =
if n < 0 then raise Negative(n)
else n * factorial1(n - 1);
val factorial1 = fn : int -> int
fun factorial(n) = factorial1(n) handle
Negative(n) => (print ("Error : n = ");
print(Int.toString(n));
print("n");
0);
val factorial = fn : int -> int
factorial(0);
val it = 1 : int
factorial(2);
val it = 2 : int
factorial(3);
val it = 6 : int
factorial(0 - 5);
Error : n = 5
val it = 0 : int
```