lab1

课堂练习

在提示符下依次输入下列语句，观察并分析每次语句的 执行结果。

```
3+ 4;
(*val it=7:int*)
3 + 2.0;
(*
类型不匹配
stdIn:2.1-2.6 Error: operator and operand don't agree [literal]
  operator domain: int * int
  operand:         int * real
  in expression:
    3 + 2.0
*)
it + 6;
(*val it = 13 : int*)
val it = “hello”;
(*val it = "hello" : string*)
it + “ world”;
(*
+号没有重载sting类型的连接，应该使用^号
stdIn:8.3 Error: overloaded variable not defined at type
  symbol: +
  type: string
*)
it + 5;
(*
string + int 类型不匹配
stdIn:10.3 Error: overloaded variable not defined at type
  symbol: +
  type: string
*)
val a = 5;
(*
val a = 5 : int
*)
a = 6;
(*
=用于判断，并非赋值
val it = false : bool
*)
a + 8;
(*
val it = 13 : int
*)
val twice = (fn x => 2 * x);
(*
val twice = fn : int -> int
*)
```

```
twice a;
(*
val it = 10 : int
*)
let x = 1 in x end;
(*
应该在x之前加上‘val’
stdIn:1.2-11.3 Error: syntax error: deleting  LET ID EQUALOP
stdIn:11.5 Error: syntax error found at IN
*)
foo;
(*
无意义的语句
stdIn:1.2-9.1 Error: unbound variable or constructor: foo
*)
[1, “foo”];
(*
列表中应该为类型一致的元素
stdIn:11.1 Error: illegal token
stdIn:11.2 Error: illegal token
stdIn:11.6 Error: illegal token
stdIn:11.7 Error: illegal token
*)
```

1

下列模式能否与类型为 int list的L匹配成功？如果匹配不成 功，指出该模式的类型？（假设x为int类型）

```
1. x::L 可以匹配     非空List
2. _::_ 可以匹配     非空List
3. x::(y::L) 可以匹配    至少有两个元素的List
4. （x::y)::L 不可以匹配   应该为列表的列表
5. [x,y] 可以匹配    有两个元素的列表
```

2

试写出与下列表述相对应的模式。如果没有模式与其 对应，试说明原因。

- list of length 3 -》[x,y,z]
- lists of length 2 or 3 -》不存在，不能拆开表示
- Non-empty lists of pairs -》 (x,y)::L
- Pairs with both components being non-empty lists -》(d1::L1,d2::L2)

3

```
分析下述程序段（左边括号内为标注的行号）：
(1) val x : int = 3;
(2) val temp : int = x + 1;
(3) fun assemble (x : int, y : real) : int =
(4)     let val g : real =
            let val x : int = 2
```

```
(5)         val m : real = 6.2 * (real x)
(6)         val x : int = 9001
(7)         val y : real = m * y (* y=12.4*3 *)
(8)         in y - m (* 12.4*3 -12.4 *)
(9)         end
(10)    in
(11)    x + (trunc g) (* g=24.8 *)
(12)    end;
(13)
(14) val z = assemble (x, 3.0);
```

试问：第4行中的x、第5行中的m和第6行中的x的声明绑定的类型和值分别为什么？第14行表达式 `assemble(x, 3.0)`计算的结果是什么？

- 第4行的x： 绑定的类型为int，值为2
- 第5行的m：绑定的类型为real, 值为12.4
- 第6行的x ： 绑定的类型为int,值为9001
- 结果为：27

4

编写函数实现下列功能：

1. zip: string list * int list -> (string * int) list

   其功能是提取第一个string list中的第i个元素和第二个int list中的第i 个元素组成结果list中的第i个二元组。如果两个list的长度不同，则结 果的长度为两个参数list长度的最小值。

2. unzip: (string * int) list -> string list * int list

   其功能是执行zip函数的反向操作，将二元组list中的元素分解成两个 list，第一个list中的元素为参数中二元组的第一个元素的list，第二个 list中的元素为参数中二元组的第二个元素的list。 对所有元素L1: string list和L2: int list，unzip(zip (L1, L2)) = (L1, L2)是否成 立？如果成立，试证明之；否则说明原因。

```
//第1题
fun zip([]:string list,d2::L2:int list): (string * int) list =[]
| zip([]:string list,[]:int list):(string*int)list =[]
| zip(d1::L1:string list,[]:int list) :(string * int) list=[]
| zip(d1::L1:string list,d2::L2:int list): (string * int) list=
(d1,d2)::zip(L1,L2);

val a=["a","b","c"];
val b=[1,2,3];
val c=zip(a,b);

val a=["a","b"];
val b=[1,2,3];
val c=zip(a,b);

val a=["a","b","c"];
val b=[1,2];
val c=zip(a,b);
```

```
//第2题
fun unzip([]: (string * int) list): string list * int list=([],[])
| unzip((x,y)::L)=
    let
    val(a,b)=unzip(L);
    in
    (x::a,y::b)
    end;

val yy=unzip([("a",1),("b",2)]);
```

并非对所有的都不成立。

5 指出下列代码的错误：

```
(* pi: real *)
val pi : real = 3.14159;
(* fact: int -> int *)
fun fact (0 : int) : int = 1
| fact n = n * (fact (n - 1));
(* f : int -> int *)
fun f (3 : int) : int = 9
(x)f _ = 4;(*少加了|*)
(* circ : real -> real *)
(x)fun circ (r : real) : real = 2 * pi * r (*应该为2.0*)
(* semicirc : real -> real *)
(x)fun semicirc : real = pie * r(*没有定义pie这个变量*)
(* area : real -> real *)
(x)fun area (r : int) : real = pi * r * r
    (*应该为r:real*)
```

实验上机

```
(* mult : int list -> int *)
(* REQUIRES: true *)
(* ENSURES: mult(L) evaluates to the product of the integers in L. *)

fun mult[]=0
|   mult(x::L)=
    if L=[] then
        x
    else
        x*mult(L);

val a=[1,2,3,4,5];
val b=mult(a);

(* mult : int list list -> int *)
(* REQUIRES: true *)
(* ENSURES: mult(R) evaluates to the product of all the integers in the
lists of R. *)

fun Mult []=0
    | Mult(r::R)=
    if R=[] then
```

```
        mult(r)
    else
        mult(r)*Mult(R);

val a=[[1,2,3,4],[5,6]];
val b=Mult(a);

(* mult' : int list * int -> int *)
(* REQUIRES: true *)
(* ENSURES: mult'(L, a) … (* FILL IN *) *)
(* 函数功能：mul([1,2,3],4)-> 1*2*3*4    *)

fun mult'([],a)=a
| mult'(x::L,a)=mult'(L,x*a);

fun Mult'([],a)=a
| Mult'(r::R,a)=mult'(r,a)*Mult'(R,1);

val A=[[1,2,3],[4,5,6]];
val B=2;
val C=Mult'(A,B);

(* double : int -> int *)
(* REQUIRES: n >= 0 *)
(* ENSURES: double n evaluates to 2 * n.*)
fun double (0 : int) : int = 0
| double n = 2 + double (n - 1)

(* square: int -> int *)
(* REQUIRES: n >= 0*)
(* ENSURES: square n evaluates to n * n.*)

fun square (0:int):int =0
| square(n)=square(n-1)+double(n)-1;

val a=11;

val b=square(a);


(* divisibleByThree : int -> bool *)
(* REQUIRES: true *)
(* ENSURES: divisibleByThree n evaluates to true if n is a multiple of 3 and
to false otherwise *)

fun divisibleByThree (0:int):bool = true
    | divisibleByThree(1:int):bool = false
    | divisibleByThree(2:int):bool = false
    | divisibleByThree(n:int):bool =
    if n<0 then divisibleByThree(abs(n))
    else divisibleByThree(n-3);

val a=7;
val b=divisibleByThree a;

val a=9;
val b=divisibleByThree a;
```

```
(* evenP : int -> bool *)
(* REQUIRES: n >= 0 *)
(* ENSURES: evenP n evaluates to true iff n is even. *)
fun evenP (0 : int) : bool = true
| evenP 1 = false
| evenP n = evenP (n - 2)


(* oddP : int -> bool *)
(* REQUIRES: n >= 0 *)
(* ENSURES: oddP n evaluates to true iff n is odd. *)
fun oddP (0:int):bool =false
| oddP(1:int):bool =true
| oddP n =oddP(n-2);

val a=7;
val b=oddP(a);
```