## Mission 1

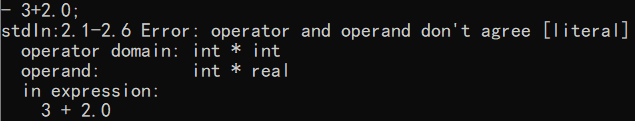
**1）**

3 + 4;

(\* val it = 7 \*)

**2）**

3 + 2.0;

****(\* 错误，real和int类型不匹配，改成3.0 + 2.0 \*)

**3）**

it + 6;

****(\* val it = 13 \*)

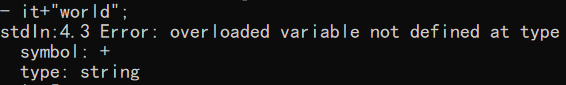
**4）**

**val** it = "hello";

****(\* val it = "hello" \*)

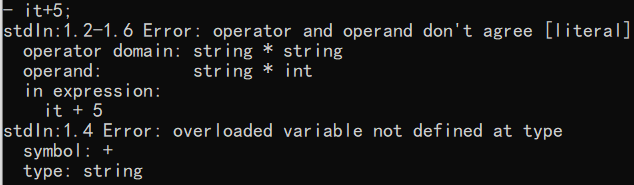
**5）**

it + "world";

****(\* 错误，字符串类型没有+操作符，要用^操作符拼接\*)

**6）**

it + 5;

****(\* 错误，string和int类型不匹配 \*)

**7）**

**val** a = 5;

****(\* val a = 5 \*)

**8）**

a = 6;

****(\* val it = false \*)

**9）**

a + 8;

****(\* val a = 13 \*)

**10）**

**val** twice = (fn x => 2 \* x);

****(\* val twice = fn : int -> int \*)

**11）**

twice a;

****(\* val it = 10 \*)

**12）**

let x = 1 in x end;

****(\* 错误，x=1是bool值，给x赋值要用val x = 1 \*)

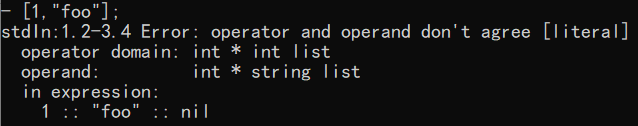
**13）**

foo;

****(\* 错误，foo未绑定 \*)

**14）**

[1,"foo"];

****(\* 错误，list里面不能包含不同类型的成员 \*)

## Mission 2

(\* mult : int list -> int \*)

(\* REQUIRES: true \*)

(\* ENSURES: mult(L) evaluates to the product of the integers in L. \*)

**fun** mult [] = 1

    | mult(x::L) = x \* (mult L);

## Mission 3

(\* 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 [] = 1

    | Mult(r::R) = (mult r)\*(Mult R);

## Mission 4

(\* mult' : int list \* int -> int \*)

(\* REQUIRES: true \*)

(\* ENSURES: mult'(L) evaluates to the product of the integers in L and a. \*)

**fun** mult' ([],a) = a

    | mult' (x::L,a) = mult'(L,x\*a);

## Mission 5

(\* 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 (1:int):int = 1

    | square n = double(n) + double(n-2) + square(n-2);

## Mission 6

(\* divisibleByThree : int -> bool \*)

(\* REQUIRES: true \*)

(\* ENSURES: divisibleByThree n evaluates to true if n is a multiple of 3 and to false otherwise \*)

**fun** divisibleByThree (n:int):bool =

    n = (n div 3) \* 3;

## Mission 7

(\* oddP : int -> bool \*)

(\* REQUIRES: n>=0 \*)

(\* ENSURES: oddP n evaluates to true if n is odd \*)

**fun** oddP (0:int):bool = false

    | oddP 1 = true

| oddP n = oddP(n-2);

**CS1701-熊逸钦-U201714501**