## Mission 1

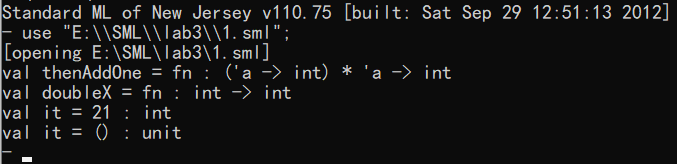
(\* thenAddOne: ((int->int)\*int)->int \*)

**fun** thenAddOne (f, x) = f x + 1;

(\* 测试 \*)

**val** doubleX = fn x => 2\*x;

thenAddOne(doubleX,10);     (\* 正确应得出21 \*)

****

## Mission 2

(\* mapList: (('a->'b)\*'a list)->'b list \*)

**fun** mapList(f, []) = []

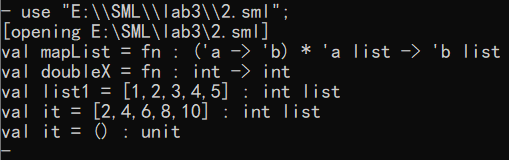
    | mapList(f, x::L) = (f x)::mapList(f, L);

(\* 测试 \*)

**val** doubleX = fn x => 2\*x;

**val** list1 = [1,2,3,4,5];

mapList(doubleX, list1);(\* 正确应得出[2,4,6,8,10] \*)

****

## Mission 3

(\* mapList': ('a->'b)->('a list->'b list) \*)

**fun** mapList' f =

    fn L => case L of

       [] => []

     | x::R => (f x)::(mapList' f R);

(\* 或可以直接写为：

fun mapList' f [] = []

    | mapList' f x::L = (f x)::(mapList' f L);

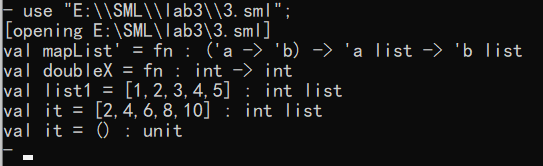
 \*)

(\* 测试 \*)

**val** doubleX = fn x => 2\*x;

**val** list1 = [1,2,3,4,5];

mapList' doubleX list1; (\* 正确应得出[2,4,6,8,10] \*)

****

## Mission 4

(\* findOdd: int list -> int option \*)

**fun** findOdd [] = NONE

    | findOdd (x::L) =

    if (x mod 2)=1 then

     SOME x

    else

     findOdd L;

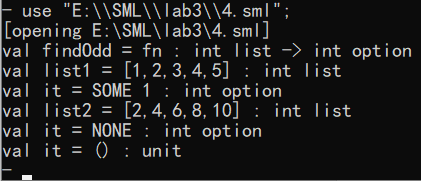
(\* 测试 \*)

**val** list1 = [1,2,3,4,5];

findOdd list1;     (\* 正确应得出SOME 1 \*)

**val** list2 = [2,4,6,8,10];

findOdd list2;     (\* 正确应得出NONE \*)

****

## Mission 5

(\* subsetSumOption:int list\*int->int list option \*)

**fun** subsetSumOption ([], s) = NONE

    | subsetSumOption (L, 0) = SOME []

    | subsetSumOption (x::L, s) =

    if subsetSumOption (L, s-x) = NONE then

     subsetSumOption (L, s)

    else

     SOME (x::valOf(subsetSumOption (L, s-x)));

(\* 测试 \*)

**val** list1 = [1,2,3,4,5];

subsetSumOption(list1, 0);

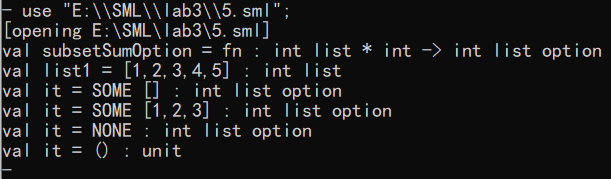
(\* 正确应得出SOME [] \*)

subsetSumOption(list1, 6);

(\* 正确应得出SOME [1,2,3] \*)

subsetSumOption(list1, 66);

(\* 正确应得出NONE \*)

****

## Mission 6

(\* exists: ('a->bool)->'a list->bool \*)

(\* ENSURE: exists p L => true if there is an x in L such that p x = true, false otherwise \*)

**fun** exists p [] = false

    | exists p (x::L) =

    if (p x) then

    true

    else

    exists p L;

(\* forall: ('a->bool)->'a list->bool \*)

(\* ENSURE: forall p L => true if p x = true for every item x in L, false otherwise \*)

**fun** forall p [] = false

    | forall p [x] =

    if (p x) then

    true

    else

    false

    | forall p (x::L) =

    if (p x) andalso (forall p L) then

    true

    else

    false;

(\* 测试 \*)

**val** lessThan10 = fn x => x<10;

(\* x小于10返回true，其他情况false \*)

**val** list1 = [1,2,3,4,5];

**val** list2 = [6,7,8,9,10];

**val** list3 = [11,12,13,14,15];

exists lessThan10 list1;

(\* 正确应得出true \*)

exists lessThan10 list2;

(\* 正确应得出true \*)

exists lessThan10 list3;

(\* 正确应得出false \*)

forall lessThan10 list1;

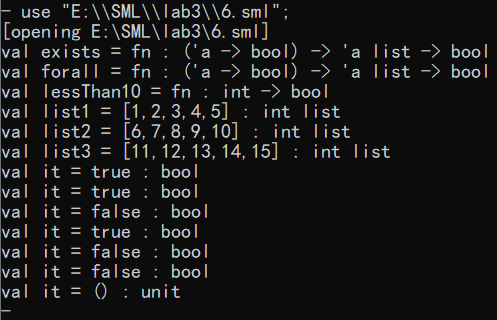
(\* 正确应得出true \*)

forall lessThan10 list2;

(\* 正确应得出false \*)

forall lessThan10 list3;

(\* 正确应得出false \*)

****

## Mission 7

(\* 类型定义 \*)

datatype 'a tree = Empty | Node of 'a tree \* 'a \* 'a tree;

(\* treeFilter: ('a->bool)->'a tree->'a option tree \*)

(\* ENSURE: 将树中满足条件P ('a->bool)的节点封装成option类型保留，否则替换成NONE \*)

**fun** treeFilter P Empty = Empty

    | treeFilter P (Node(t1, x, t2)) =

    if (P x) then

    Node(treeFilter P t1, SOME x, treeFilter P t2)

    else

    Node(treeFilter P t1, NONE, treeFilter P t2);

(\* 测试 \*)

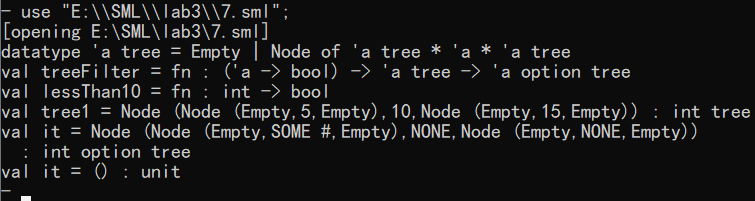
**val** lessThan10 = fn x => x<10;

(\* x小于10返回true，其他情况false \*)

**val** tree1 = Node(Node(Empty,5,Empty),10,Node(Empty,15,Empty));

treeFilter lessThan10 tree1;

(\* 正确应得出Node(Node(Empty,SOME 5,Empty),NONE,Node(Empty,NONE,Empty)) \*)

****

**CS1701-熊逸钦-U201714501**