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 *)
```

```
Standard ML of New Jersey v110.75 [built: Sat Sep 29 12:51:13 2012]
- use "E:\\SML\\lab3\\1.sml";
[opening E:\SML\lab3\1.sml]
val thenAddOne = fn : ('a -> int) * 'a -> int
val doubleX = fn : int -> int
val it = 21 : int
val it = () : unit
-
```

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] *)
```

```
- use "E:\\SML\\lab3\\2.sml";
[opening E:\SML\lab3\2.sml]
val mapList = fn : ('a -> 'b) * 'a list -> 'b list
val doubleX = fn : int -> int
val list1 = [1,2,3,4,5] : int list
val it = [2,4,6,8,10] : int list
val it = () : unit
-
```

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] *)
```

```
- use "E:\\SML\\lab3\\3.sml";
[opening E:\SML\lab3\3.sml]
val mapList' = fn : ('a -> 'b) -> 'a list -> 'b list
val doubleX = fn : int -> int
val list1 = [1,2,3,4,5] : int list
val it = [2,4,6,8,10] : int list
val it = () : unit
-
```

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 *)
```

```
- use "E:\\SML\\lab3\\4.sml";
[opening E:\SML\lab3\4.sml]
val findOdd = fn : int list -> int option
val list1 = [1,2,3,4,5] : int list
val it = SOME 1 : int option
val list2 = [2,4,6,8,10] : int list
val it = NONE : int option
val it = () : unit
-
```

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 *)
```

```
- use "E:\\SML\\lab3\\5.sml";
[opening E:\SML\lab3\5.sml]
val subsetSumOption = fn : int list * int -> int list option
val list1 = [1,2,3,4,5] : int list
val it = SOME [] : int list option
val it = SOME [1,2,3] : int list option
val it = NONE : int list option
val it = () : unit
-
```

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 eve
ry 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 *)
```

```
- use "E:\\SML\\lab3\\6.sml";
[opening E:\SML\lab3\6.sml]
val exists = fn : ('a -> bool) -> 'a list -> bool
val forall = fn : ('a -> bool) -> 'a list -> bool
val lessThan10 = fn : int -> bool
val list1 = [1,2,3,4,5] : int list
val list2 = [6,7,8,9,10] : int list
val list3 = [11,12,13,14,15] : int list
val it = true : bool
val it = true : bool
val it = false : bool
val it = true : bool
val it = false : bool
val it = false : bool
val it = () : unit
-
```

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)) *)
```

```
- use "E:\\SML\\lab3\\7.sml";
[opening E:\SML\lab3\7.sml]
datatype 'a tree = Empty | Node of 'a tree * 'a * 'a tree
val treeFilter = fn : ('a -> bool) -> 'a tree -> 'a option tree
val lessThan10 = fn : int -> bool
val tree1 = Node (Node (Empty,5,Empty),10,Node (Empty,15,Empty)) : int tree
val it = Node (Node (Empty,SOME #,Empty),NONE,Node (Empty,NONE,Empty))
  : int option tree
val it = () : unit
-
```

CS1701-熊逸钦-U201714501