hw1

1.

- x::L 成功
- _::_ 成功
- x::(y::L) 不成功 L长度至少为1
- (x::y)::L 不成功 'Z list list
- [x,y] 成功

2.

- [x,y,z]
- 不存在，list长度需确定
- (x,y)::L
- (x::L1,y::L2)

3.

- 第4行中的x 类型为int 值为2
- 第5行中的x 类型为real 值为2.0?
- 第6行中的x 类型为int 值为9001
- 第14行表达式计算结果是27

4.

1.
```
fun zip ([],[]) = []
   |zip ([],il:int list) = []
   |zip (sl:string list,[]) = []
   |zip (x::sl,y::il) = (x,y)::zip(sl,il);
```

2.
```
fun unzip [] = ([],[])
   |unzip((x:string,y:int)::pairs) =
            let val (xs,ys) = unzip pairs
            in (x::xs,y::ys) end;
```

5.

- fun f (3:int) : int = 9

  | f _ = 4;
- fun circ (r: real) : real = 2.0 * pi * r;
- fun semicirc r = pi * r;
- fun area (r : real) : real = pi * r * r;

6.

1. M: (fib n=> if n <= 2 then 1 else fib(n-1) + fib(n-2))

   令n = 5，代入函数

   fib 5 =>M 5

   $\Rightarrow^{(5)}$ (M 4) + (M 3)

=>$^{(5)}$(M 3) + (M 2) + (M 3)

=>$^{(5)}$(M 2) + (M 1) + (M 2) + (M 3)

=>$^{(5)}$(M 2) + (M 1) + (M 2) + (M 2) + (M 1)

=>$^{(2)*5}$1 + 1 + 1 + 1 + 1

=>$^{(4)}$5

M 5 => if 5 <= 2 then 1 else

=> fib(5-1) + fib(5-2)

=> M(5-1) + M(5-2)

=> M(4) + M(5-2)

=>M(4) + M(3)

由此推出：

for all n>=1,fib n => 2^n

用W_{fib}(n)表示程序 fib(n) 的执行时间：

W_{fib}(0) = c_0，W_{fib}(n) = $W_{fib}(n-1)$+$W_{fib}(n-2)$=c_0+2^n

近似运行时间为：O(2^n)

2.

M:(fun fibber (0: int) : int * int = (1,1)

| fibber (n:int) : int * int =

let val (x :int,y :int) = fibber (n-1)

in (y,x+y)

end)

令 n = 5，代入函数

fib 5 => M 5

=>$^{(4)}$M 4

=>$^{(4)}$M 3

=>$^{(4)}$M 2

=>$^{(4)}$M 1

=>$^{(3)}$(1,1)

=>$^{(3)}$(1,2)

=>$^{(3)}$(2,3)

=>$^{(3)}$(3,5)

=>$^{(3)}$(5,8)

M 5 => fibber (5: int) : int * int

=> let val (x :int,y :int) = fibber(5-1)

=> M(5-1)

=> M(4)

由此推出：

for all n>=1,fibber n =>7n - 3

用W_{fibber}(n)表示程序 fibber(n) 的执行时间：

$W_{fibber}(0) = c_0$，$W_{fibber}(n) = W_{fibber}(n-1) + c_1 = c_0 + n*c_1$

近似运行时间为：O(n)

hw2

1. 证明：For all L:int list,
   msort(L) = a <-sorted permutation of L.

1.
```
fun msort [ ] = [ ]
  | msort [x] = [x]
  | msort L =  let
                 val (A, B) = split L
               in
                 merge (msort A, msort B)
               end
```

2.

当t=Empty时，P(t)显然成立；

当t=Node(t1, x, t2)时，假设P(t1)，P(t2)成立，则有

- 当x>y时，SplitAt(y, t) = let val (l1, r1) = SplitAt(y,t1) in (l1, Node(r1, x, t2))
- 当x=y时，SplitAt(y, t) = t
- 当x<y时，SplitAt(y, t) = let val (l2, r2) = SplitAt(y, t2) in (Node(t1, x, l2), r2)

综上所述，对所有有序树t, P(t)成立, 命题得证。

3.

- val all =fn : int * string list -> string list
- val fun = fn : ('a * int -> int) * 'a list -> int
- val it = fn:'a -> ('b -> 'a)

4.

1.
```
fun PrefixSum([] : int list) : int list = []
  | PrefixSum([x]) = [x]
  | PrefixSum(x::y::L) = [x]@PrefixSum((x+y)::L);
```

2.
```
fun PrefixSumHelp(x:int,[]:int list):int list = []
  | PrefixSumHelp(x,y::L) = (x+y)::PrefixSumHelp(x+y,L);
fun fastPrefixSum L:int list = PrefixSumHelp(0,L);
```

5.

- ```
  fun treecompare(Empty:tree,_:tree):order = GREATER
    | treecompare(_,Empty) = LESS
    | treecompare(Node(_,x,_),Node(_,y,_))=
          case Int.compare(x,y) of
              GREATER=>GREATER
            | EQUAL => EQUAL
            | LESS => LESS;
  ```

- ```
  fun SwapDown(Empty:tree):tree = Empty
    | SwapDown(Node(Empty,x,Empty)) = Node(Empty,x,Empty)
    | SwapDown(Node(t1,x,t2))=
          case treecompare(t1,t2) of
          LESS =>
              let val Node(l,y,r) = t1
              in case Int.compare(x,y) of
                  GREATER => Node(SwapDown(Node(l,x,r)),y,t2)
                | _ => Node(t1,x,t2)
              end
        | _ =>
              let val Node(l,y,r) = t2
              in case Int.compare(x,y) of
                  GREATER => Node(t1,y,SwapDown(Node(l,x,r)))
                | _ => Node(t1,x,t2)
              end;
  ```

- ```
  fun heapify(Empty:tree):tree = Empty
    | heapify(Node(t1,x,t2)) =
          SwapDown(Node(heapify(t1),x,heapify(t2)));
  ```

hw3

2.

1. ```
   fun toInt b [] = 0
     | toInt b (x::L) = (toInt b L) * b + x;
   ```

2. ```
   fun toBase b 0 = []
     | toBase b x = (x mod b) :: (toBase b (x div b));
   ```

3. ```
   fun convert (b1:int,b2:int) =
       fn L=> toBase b2 (toInt b1 L);
   ```