1. val all = fn : int \* string list -> string list

返回一个string list，该list里包含int your个”are belong to us”

val funny = fn : ('a \* int -> int) \* 'a list -> int

参数为一个函数fn : ('a \* int -> int)和一个未知类型list，返回值是int，当第二个参数list为空时返回0，

val it = fn : 'a -> 'b -> 'a

1. (1)首先L是有序的，假设为递增排序的话，L里的所有元素都大于y，当x>y的时候，那么就返回y::ins(x,L)，显然结果依然有序。否则返回x::y::L，显然也是有序的。

(2)首先isort一定会递归到它的参数list为空为止，那么isort(x::[]) = ins(x,isort L);

这时候会返回[x]，接着返回上一层isort(y::[x]) = ins(y,isort [x]);因为ins函数会把y插入到[x]中，并且使插入后得到的list有序，也即isort(y::[x])返回值有序。不断回溯，每次都得到有序list。

1. O(2n)

fibber(n) = (n,n+1) n>0

fibber(0) = (1,1)

时间复杂度O(n)

1. fun Three(0) = true | Three(1) = false | Three(2) = false | Three(n) = Three(n - 3);
2. fun oddP(1) = true | oddP(0) = false | oddP(n) = oddP(n-2);
3. fun interleave(A,[],l) = l @ A

| interleave([],A,l) = l @ A

| interleave(x1::L1,x2::L2,l) = interleave(L1,L2,l@[x1,x2]);

fun interleave([],\_) = []

| interleave(\_,[]) = []

| interleave(x :: L, y :: R) = x::y::interleave(L,R)

1. fun reverse[] = []

| reverse [a] = [a]

| reverse (a::L) = reverse L @ [a];

fun reverse'([],y) = y

| reverse'(x::L,y) = reverse'(L,x::y);

1. fun PrefixSum([]) = []

| PrefixSum(L@A) = PrefixSum()@