1. 用归纳法，树在深度为k的位置分割，要么在根结点分割，
   1. 当k = 1 时显然p(t)成立
   2. K>2 时 分割点的左节点会被放在左边树上，分割点的右节点会被放到右边的树上，故p(t)成立
   3. int\*(string list)-> string list
   4. (‘a\*int -> int) \* ‘a list -> int
   5. string -> string

3.

O(n)时间复杂度

fun prefixsum(l*:*int list)*:*int list *=*

prefixsumhelp'(l,0);

fun prefixsumhelp'([],*\_*) *=* []

| prefixsumhelp'(x::L,sum) =

 (x+sum)::prefixsumhelp'(L,x+sum);

O(n^2)时间复杂度

fun sum [] *=* 0

| sum(x::L) = (x+sum L);

fun help(l*:*int list ,x::[]) *=* [sum(l@([x]))]

| help(l*:*int list ,x::L) =  sum(l@([x]))::help(l@([x]),L);

fun prefixsum2(l*:*int list)*:*int list *=* help([],l);

4.

fun treecompare(Node(*\_*,t1,*\_*),Node(*\_*,t2,*\_*)) *=* compare(t1,t2)  ;

swapdown 函数

// work 时间复杂度O(nlog(n))

// span为O(d)

fun swapdown Empty *=* Empty

| swapdown(t as Node(Empty,t1,Empty)) = t

| swapdown (t as Node(Node(l,m,r),n,Empty)) =

    if m>=n then t else Node (Node (Empty,n,Empty),m,Empty)

| swapdown (t as Node(Empty,n,Node(l,m,r))) =

    if m>=n then t else Node (Empty,m,Node (Empty,n,Empty))

| swapdown( t as Node(l as Node(l1,m,r1),n,r as Node(l2,q,r2))) =

    if n <= m  andalso n<= q then t

    else if n > m andalso m > q then Node(l,q,swapdown(Node(l2,n,r2)))

    else Node(swapdown(Node(l1,n,r1)),m,r);

// heapify函数

// work 时间复杂度O(log(n)^2)

// span为O(d^2)

fun heapify(Empty) *=* Empty

| heapify(t as Node(l,t1,r)) =

        swapdown(Node(heapify(l),t1,heapify(r)));