hw05

BY ZIHAN ZHOU

1

2

2.1

```
'a list * string => string
```

2.2

this does not work

```
line 1: 'a tree*'b='c list
```

we know f return some list however in line 2 there is

```
node ( f(L, n+1),...
```

we know node take only tree, list is definitely not tree

2.3

```
int => int
```

2.4

2.4.1

4, int. everything in let and in in the same scope. x is changed in line 7

2.4.2

12.0 float, m is computed in line 8 ,becasue tmp is global and is 3.0 and x is 4.0

2.4.3

2, int, because the change in line 7 is in a scope that has ended in line 13

2.4.4

float, 63.

2.5

2.5.1

4,int , because

```
line 3: y= x+3
```

2.5.2

7,int because

```
line 5: y=7
```

2.5.3

5,int, because in line 4 y = 4 and

```
line 4 : g z = z + y
```

2.5.4

6,int in line 6 line 7 , h n is (y-x)*n, y-x here is 6, so h x = 6

2.5.5

11 because gx=5 hx =6

3

3.1

3.1.1 work

$W(0) = k_0$

$W(d) = k_1 + W(d-1)$

$W(d) \in O(d)$

in the best case $d = \log n$, so it is $O(\log n)$

in the worst case $d = n$, so it is $O(n)$

3.1.2 span

everything is sequential, so the span is exatly the same as the work.

3.2

3.3

```
fun ILsort' ([])= empty
  | ILsort' (x::L)=Insert(x,ILsort'(L))
fun ILsort t = ILsort'(inorder(t))
```

Lemma 1.

For all x: int and t: int tree, If sorted (inorder t) $\cong$ *true,*

then sorted (inorder (Insert(x, t)))$\cong=$ *true.*

Proposition 2.

For all L: int list,
sorted (inorder (ILsort' L)) ≅ true

Proof. by tree list induction on L

two cases:

L= nil or L = x::L'

Base case: L= nil:

sorted (inorder (ILsort' nil))≅sorted(inorder(empty))≅sorted(nil) ≅ true

Inductive case:

To show:

sorted (inorder (ILsort' x::L')) ≅ true

IH:

sorted (inorder (ILsort' L')) ≅ true

Proof:

$$\begin{array}{lcll} \text{sorted}(\text{inorder}(\text{ILsort}'\, x :: L')) & \cong & \text{sorted}(\text{inorder}(\text{Insert}(x, \text{ILsort}\prime(L)))) & \text{by def of ILsort}' \\ & \cong & \text{true} & \text{by Lemma 2 and IH} \end{array}$$ □

Proposition 3.

For all t: int tree,
sorted (inorder (ILsort t)) ≅ true

Proof.

sorted (inorder (ILsort t))=sorted (inorder (ILsort' inorder(t)))=true □

3.4

everything is sequential work is the same as the span, i will only do work.

$W_S(n) = W_{S'}(n) + k_0$

$W_{S'}(0) = k_1$

$W_{S'}(n) = W_S(n-1) + W_{\text{Insert}}(n-1) + k_2$

In worst case:

$W_{S'}(n) = W_S(n-1) + k_3(n-1) + k_2$

$W_S \in W_{S'} \in O(n^2)$

In best case:

$W_{S'}(n) = W_S(n-1) + k_3(\log(n-1)) + k_2$

$W_S \in W_{S'} \in O(n \log n)$

3.5

always rebalance the tree after each insert

3.6

Work is the same, Span is worse than merge. Because Insertion sort does not allow parallel computation.

4