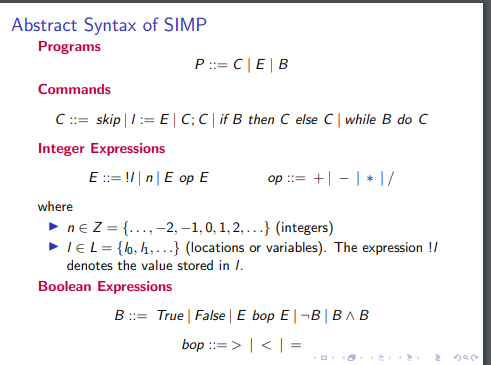
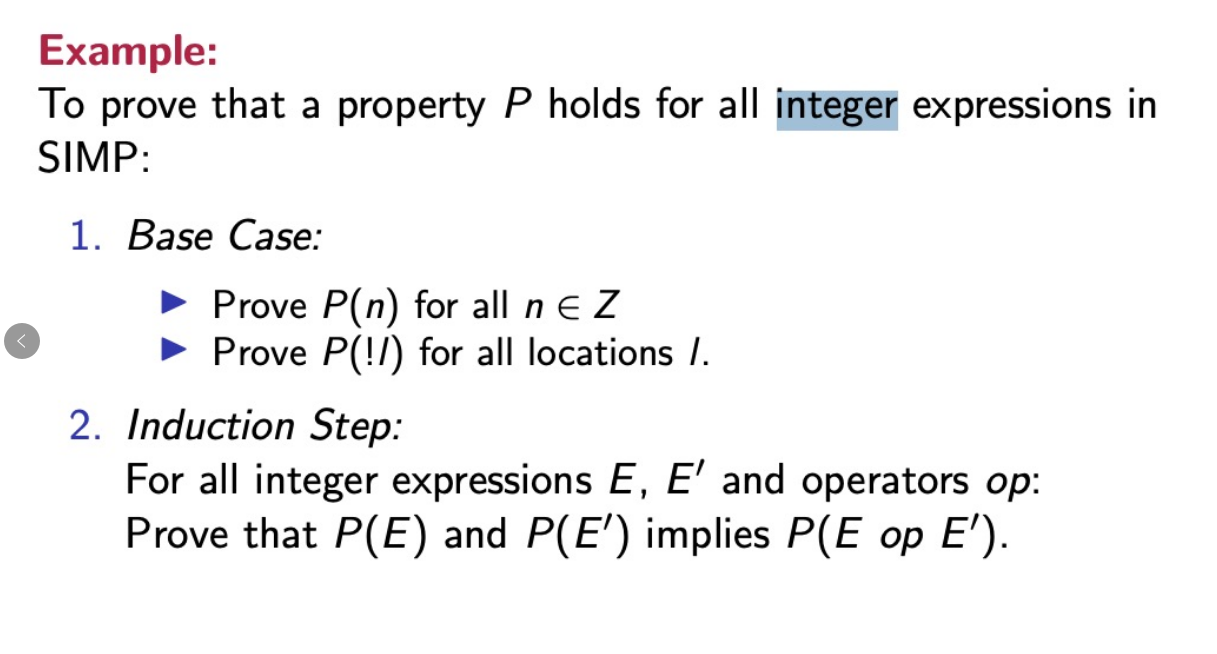
# **ÍStructural Induction 重点，出现次数多**

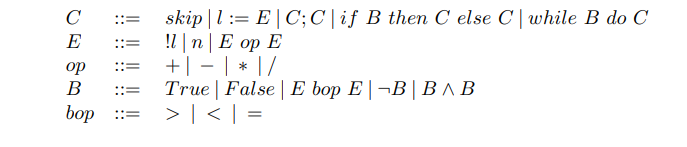


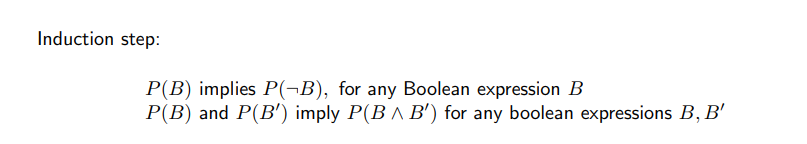
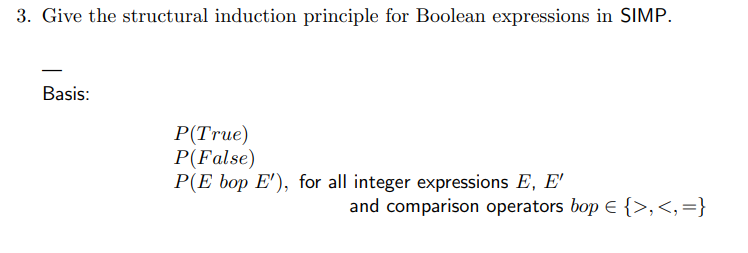


也许对：

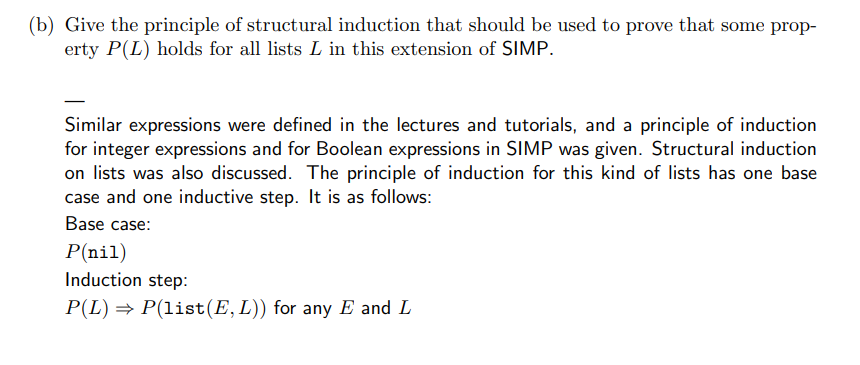
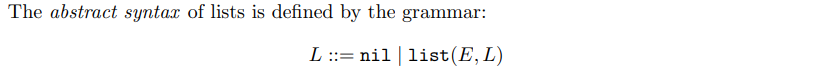
首先明确一点，我们不需要自己证，所以咋说都行

比如上面的例子，我们想证明property holds for all integer expressions in SIMP，也就是E，我们首先要证明至少这个property 能hold for E里面的东西比如 !l 和 n，这是base case。如果我们连这个都无法证明，就更别谈证更多的东西了。因为all Expression hold的前提是，some expression 能hold

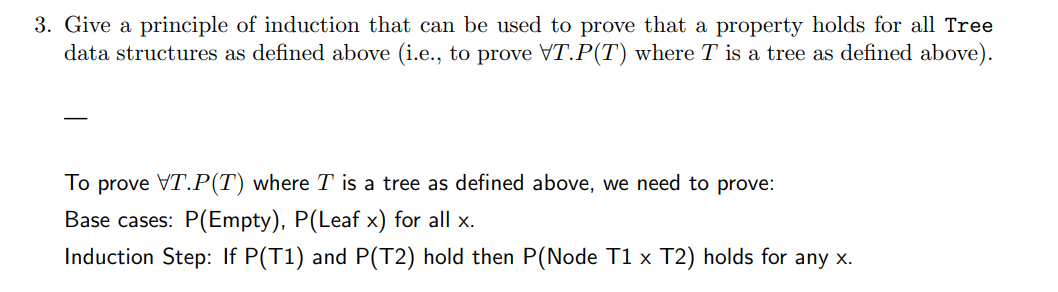
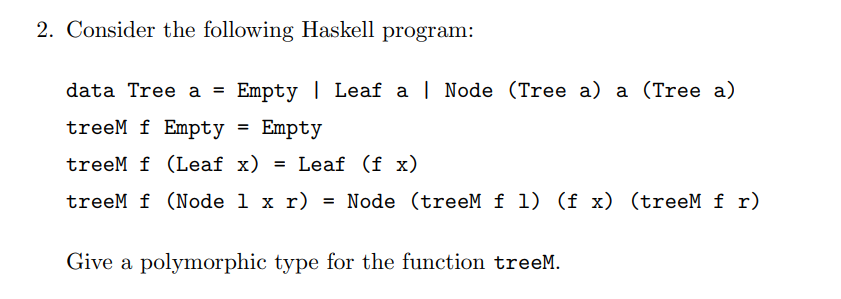




（2020）Assume 所有不带B的（这个例子里）都是satisfy property，也就是base case，然后如果一个property 在B上hold的话在not B上也会hold，因此可以推导出来。同时只要P（B）和P（B’）hold P(B^B)也能hold



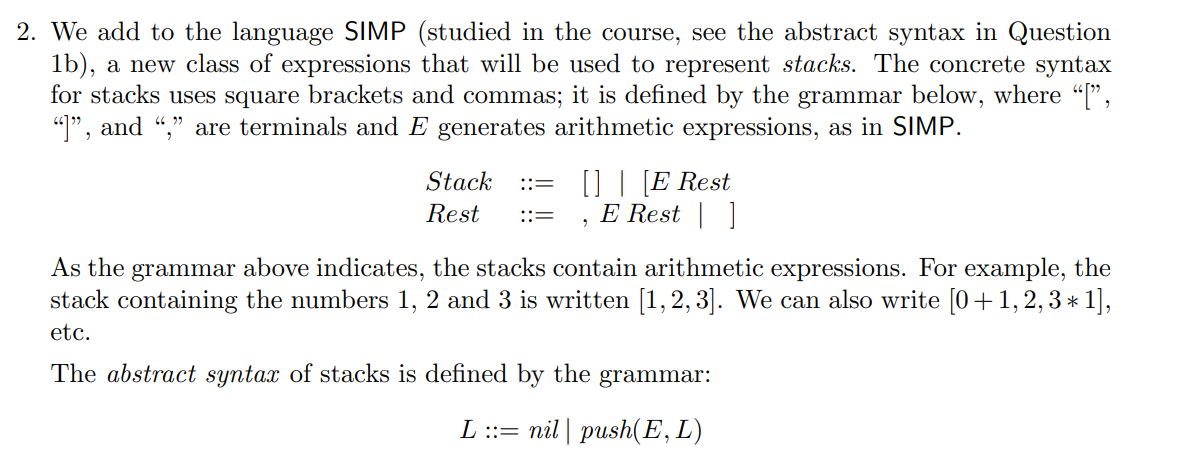
（2014）这个也是同理，assume 所有名字中不带L的也就是nil hold然后P（L）可以推导出剩下所有带L的

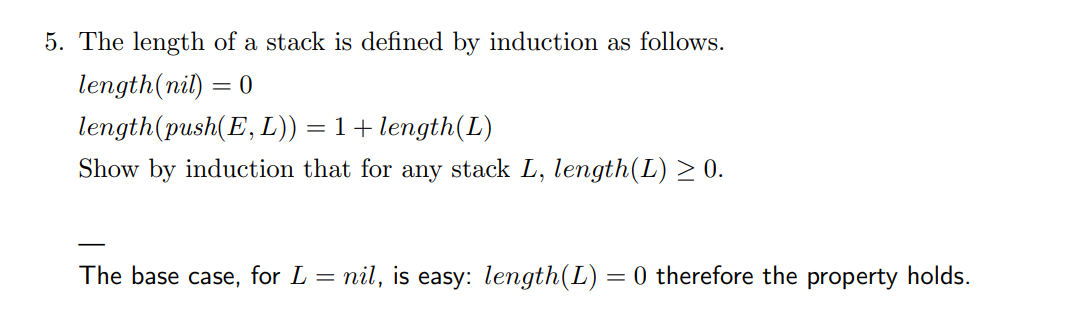
（2021）

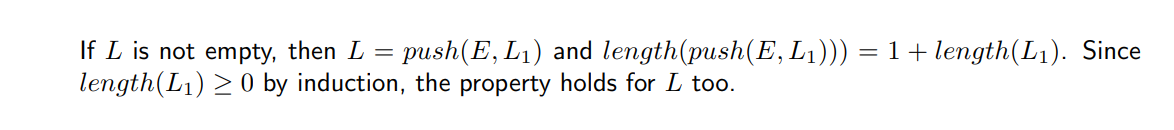
这是一个变种，本质上和prove abstract grammar 是一样的，原理也是一样的。把这里面所有不带Tree的都设为base case。只要里面的Tree a 和 Tree b hold property，我们就能进一步证吗

P（Node Treea Treeb）也hold property

2015

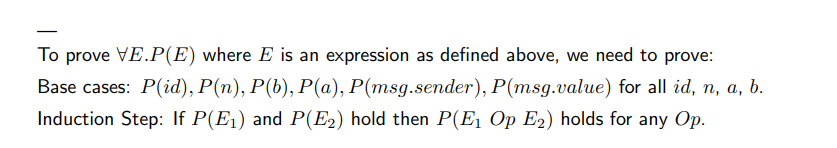
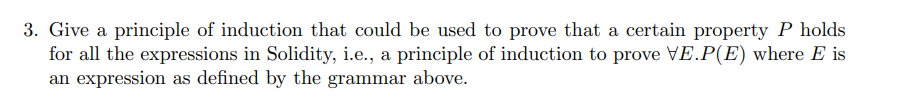
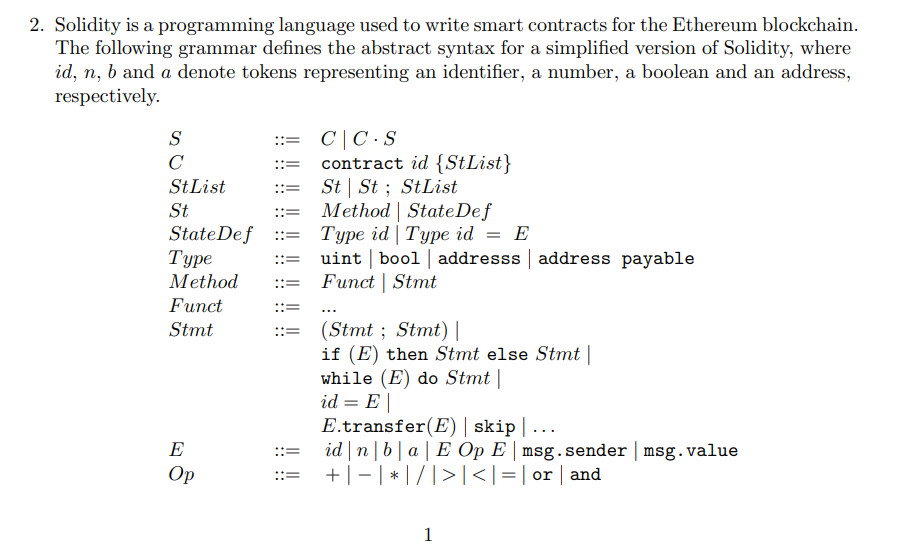




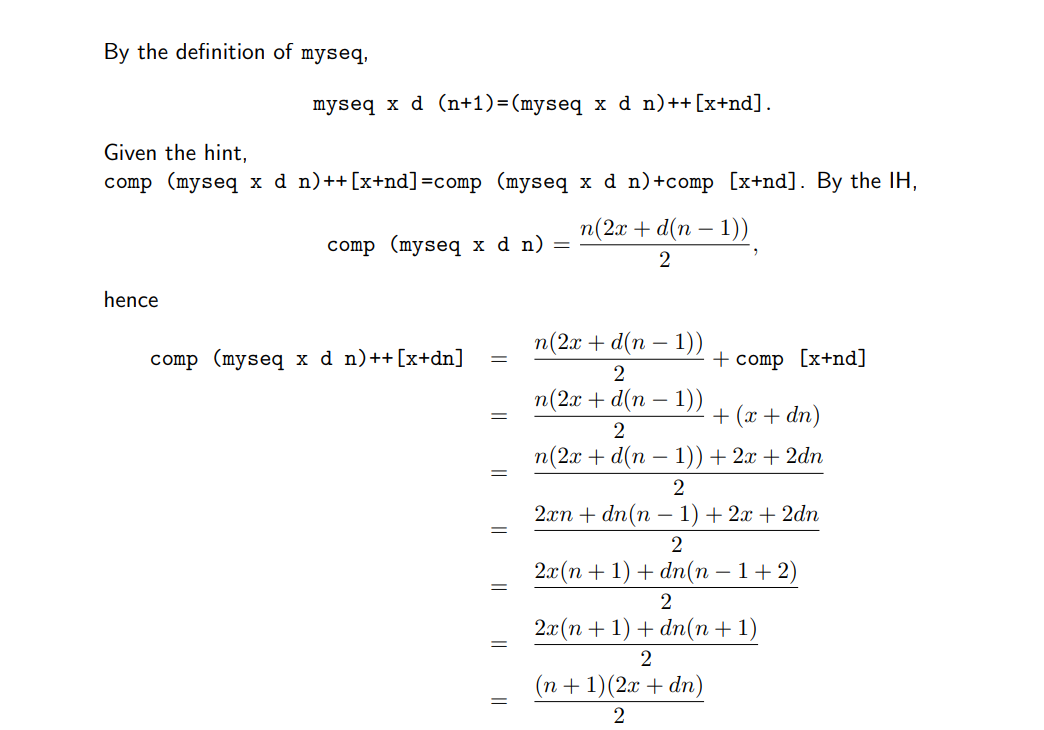
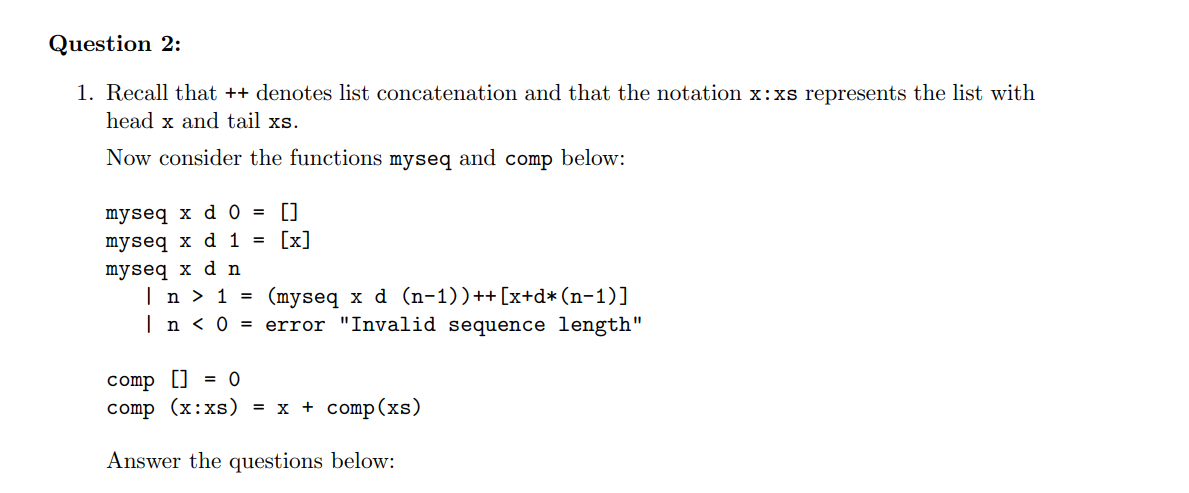


这题比较特殊，不再是要求你给出structural induction，而是直接让你去证明。

这里我们先证明base case也就是nil，已知length nil = 0，因此base case hold。然后我们再去证induction case，假设L不是empty的，那他可以拆分为length（push（E，L1）），**这里的重点是不管怎么样一定要用到所有他提供的东西**。然后已知length（push E，L1）= 1+length L1，因此就算L1也不是empty的我们也可以一直加下去，一直到Ln = nil = 0，因此L 也hold propertys



这个也比较特殊，但是原理是一样的，把所有不带E的都算base就行



Prove by induction 的逻辑

原始公式 A（x），这里是comp（myseq x d n）

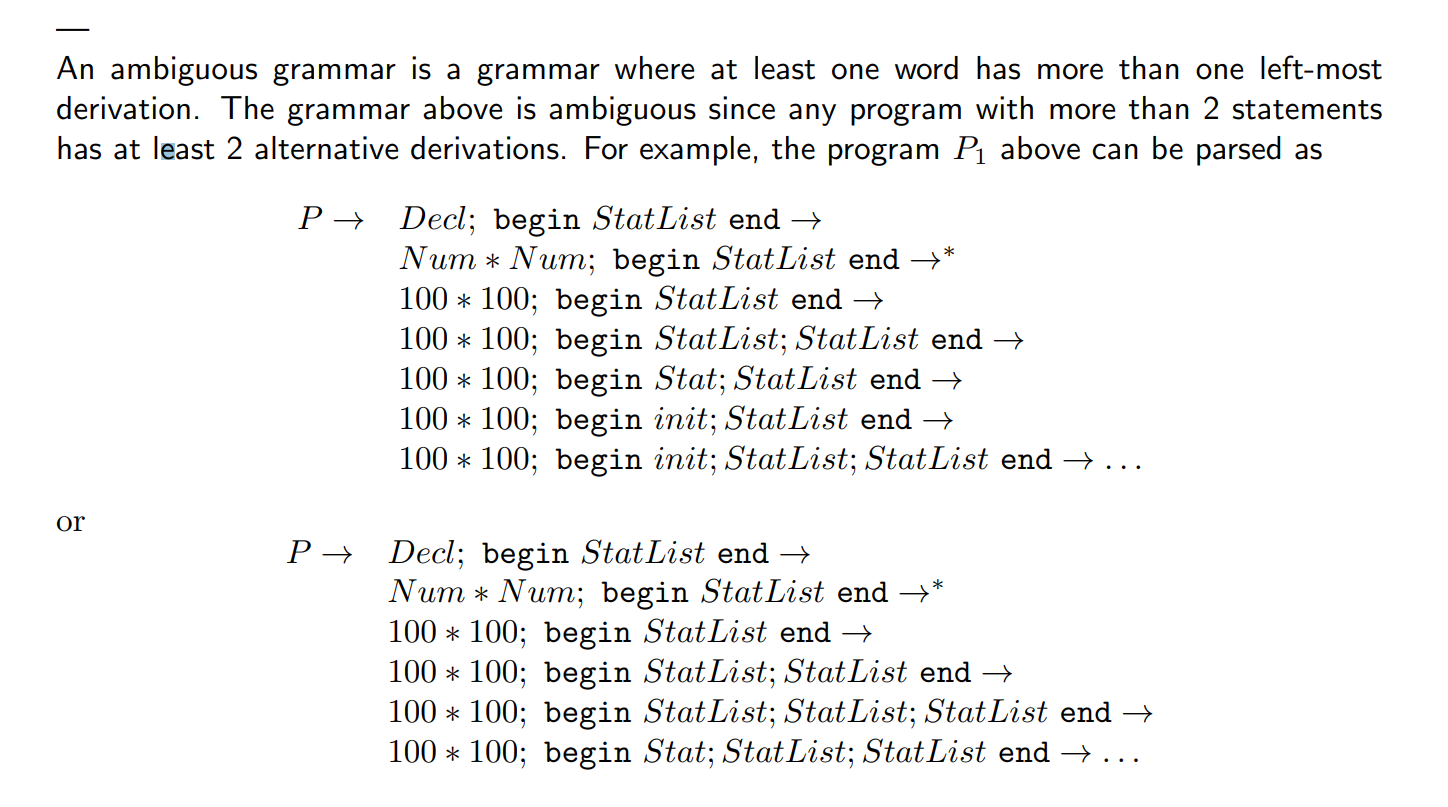
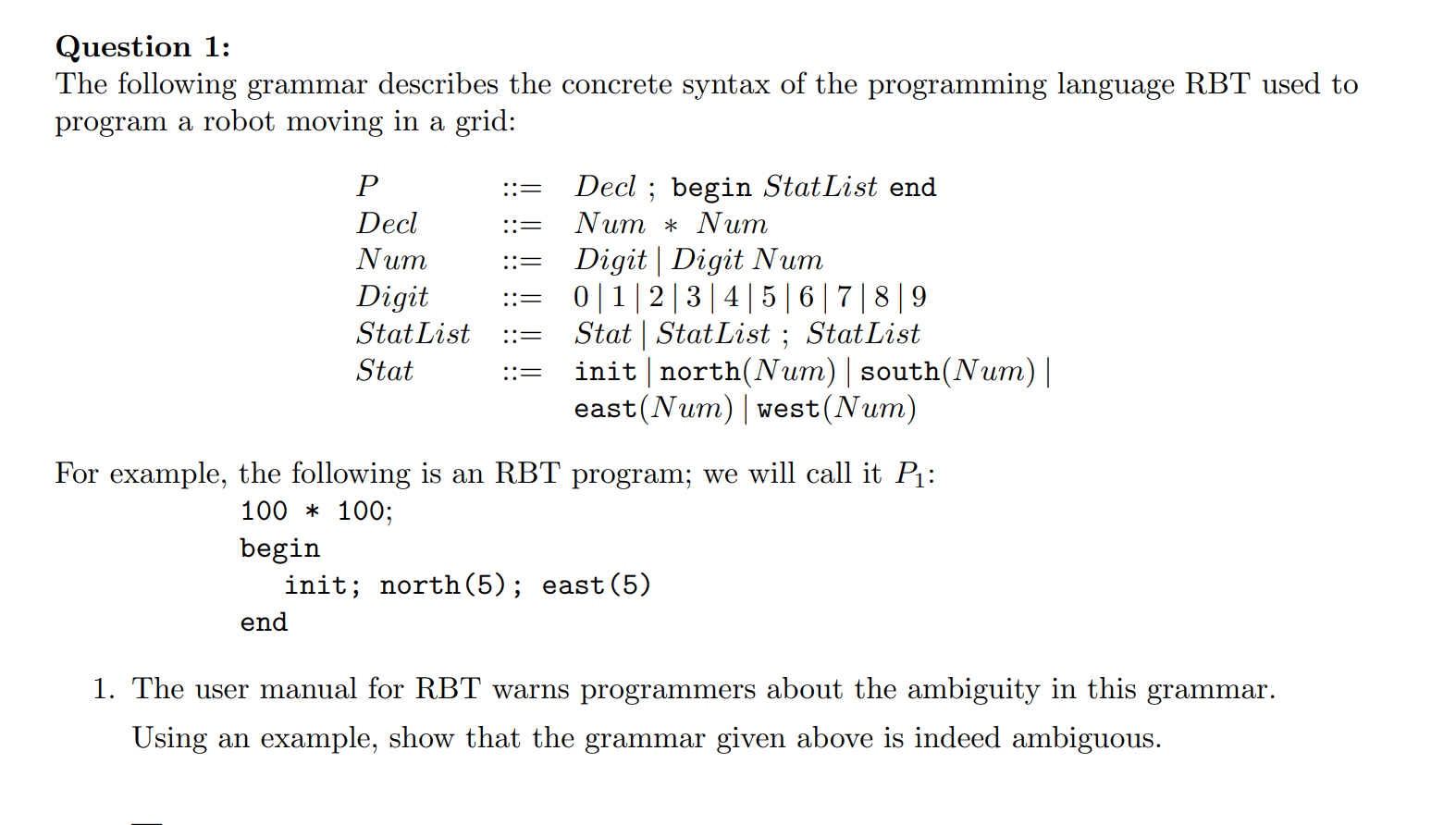
需要证明的公式 B（x）， 这里是n(2x+d(n-1))/2

首先证base case也就是 A(0) = B(0)

然后assume A(n) = B(n) hold

完成这步后再根据给予的便利将 A(n+1) 转换成 B(n) + 随便啥东西，最后再转换成 B(n+1)

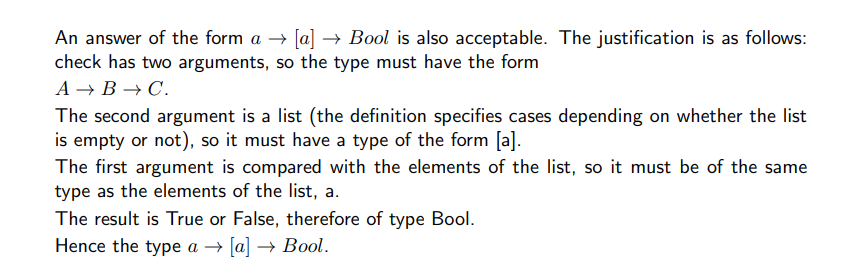
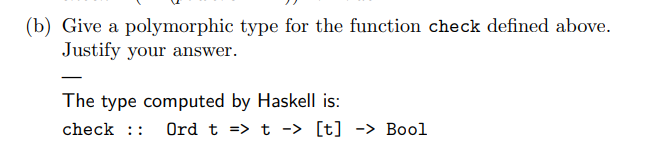
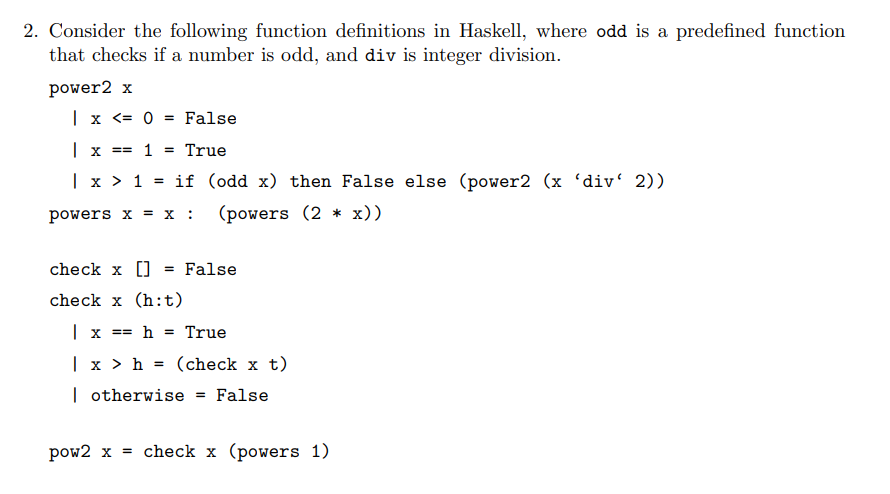
# **Find ambiguity(次数1)2017**



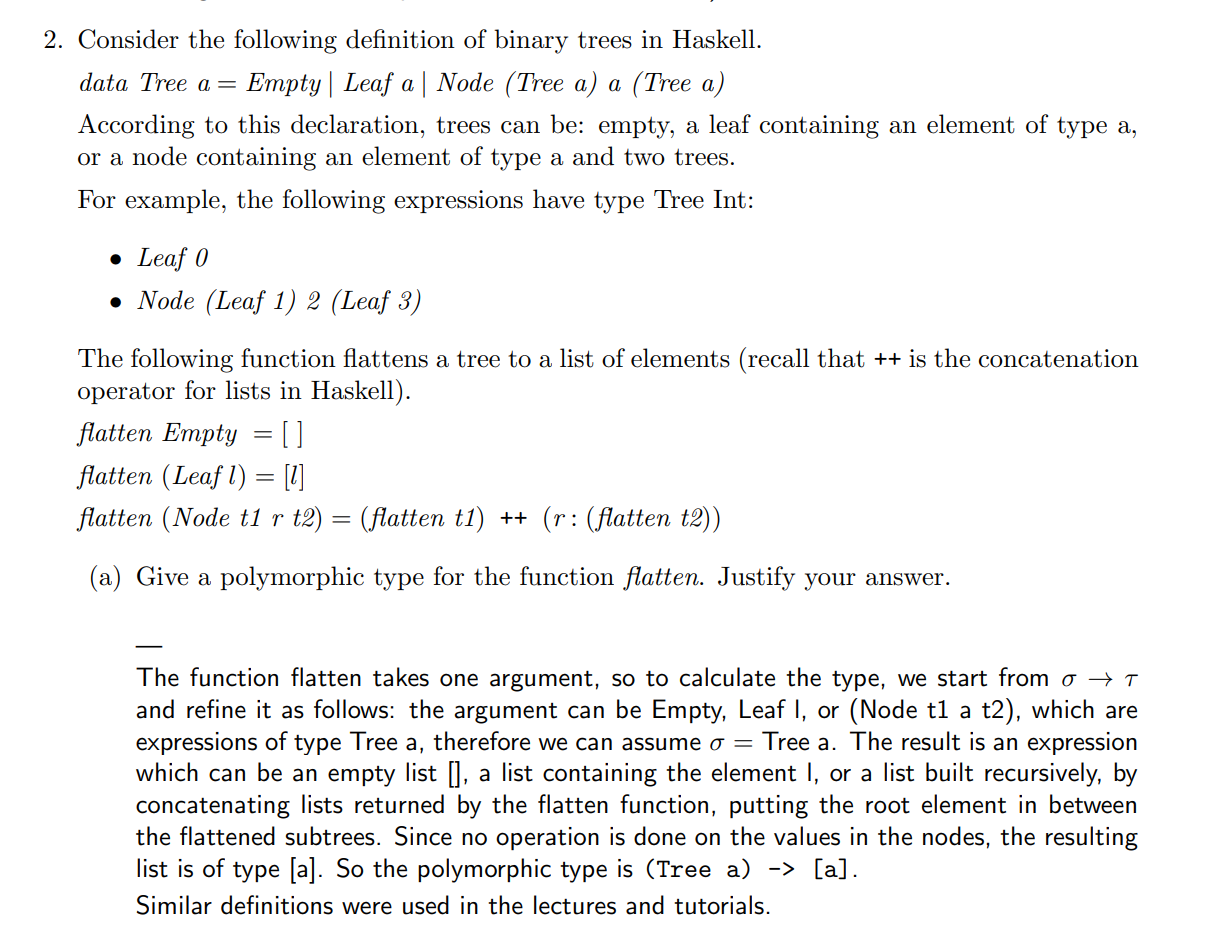
Ambiguous grammar 就是这个grammar可以拆分成一种以上的方式，如上因为StatList可以是StatList;StatList 因此可以是 StatList-> StatList;StatList -> Stat;StatList; -> init;StatList -> init;StatList;StatList，也可以是 StatList-> StatList;StatList -> StatList;StatList;StatList -> init;;StatList;StatList,因此就可能会出现歧义

# **Polymorphic type**

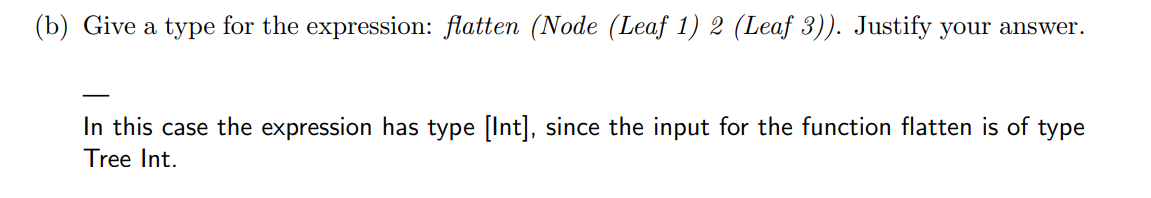
2017



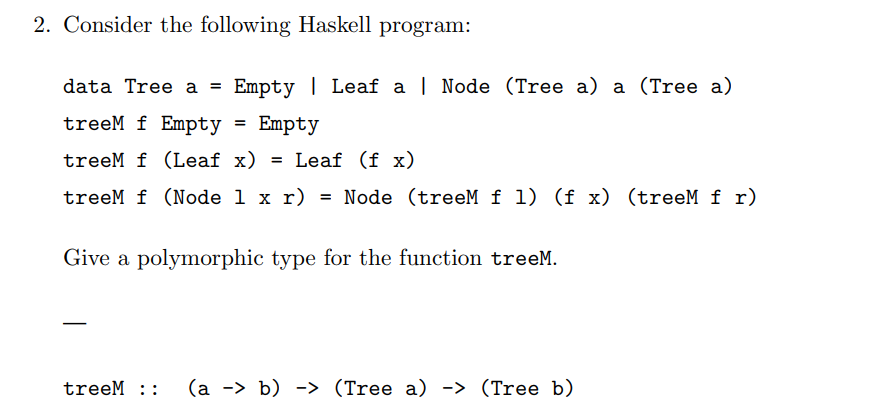
根据上面的公式可以得知 x 和 (h:t)里面的值是同一个type，check中将x和list中的第一个值进行了比较，因此他们必须是一样的值否则不能比较，同时最后的结果是True或者False也就是bool type



这题也很特殊，这题他自己定义了一个tree type，因此就是tree type -》 list

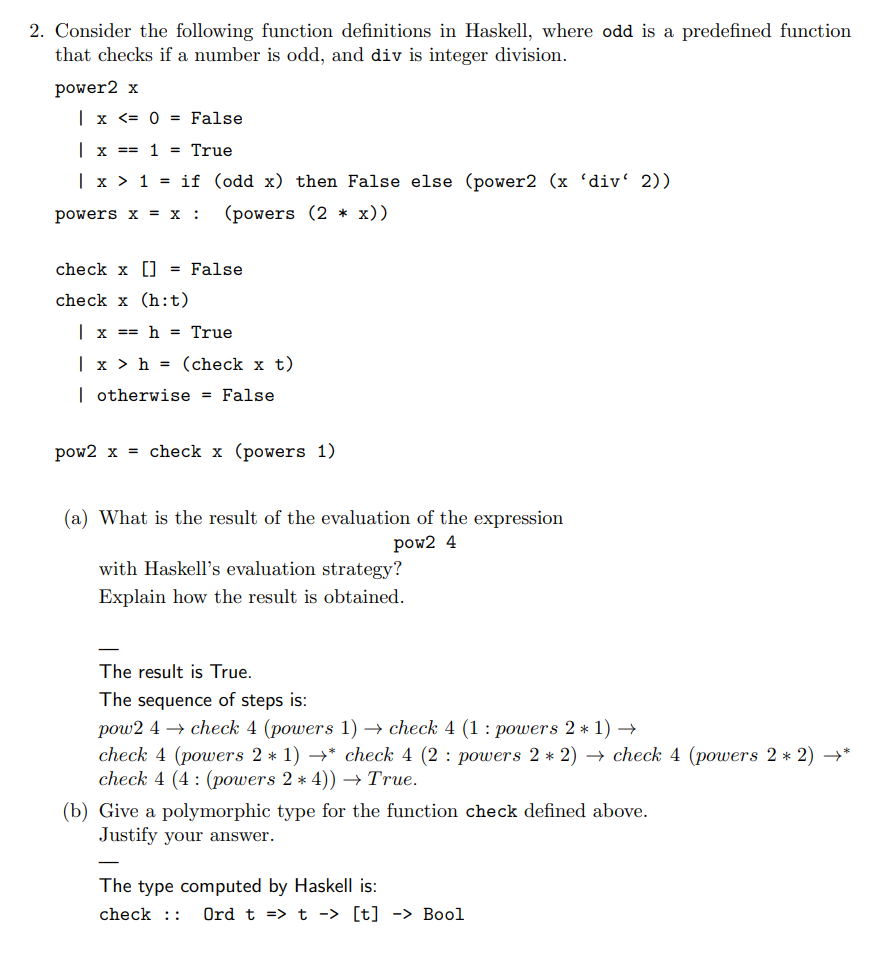


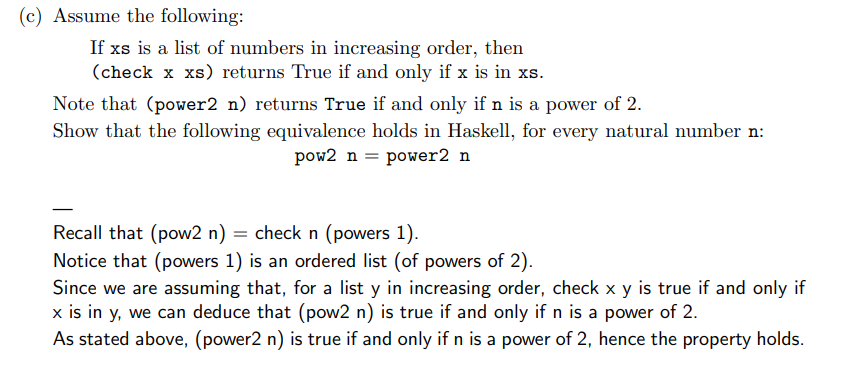
同时还有这题的b部分，这题实际上真的是让你给一个真实存在的type



f 可以得知是一个function，且根据第三行的Leaf(f x)可以得知他是一个接收和Tree a内部包含的一样是type a的值，且没有证据表明它输出的是a因此就是a -> b。同时显而易见，另一个输入的值是一个Tree，注意这里要写明Tree a。最后这个program会输出Tree b

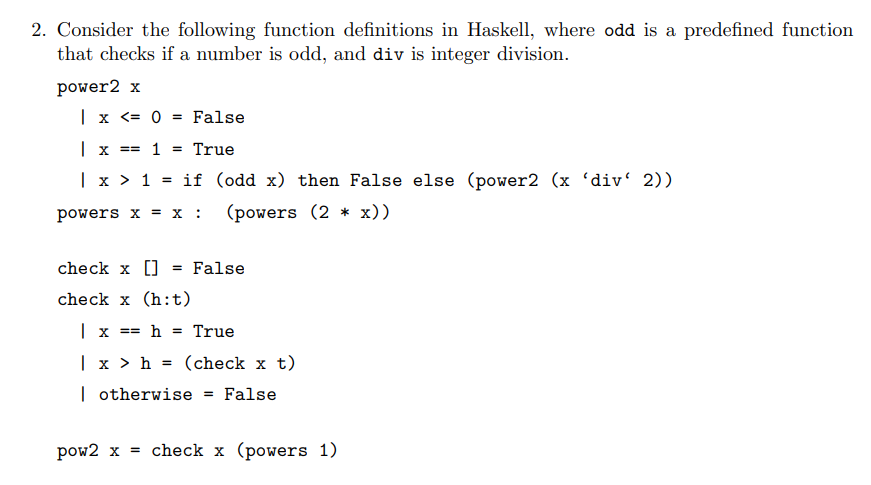
# **Haskell**

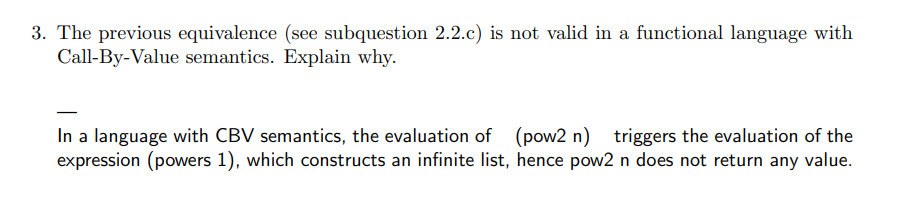


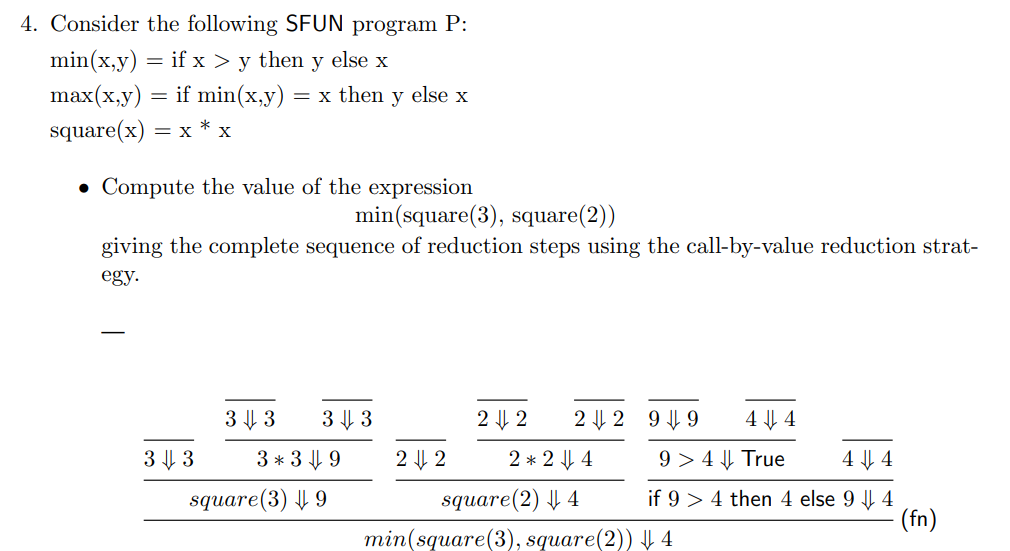


首先一直power2 n 是一个检查n到底是不是2的倍数的function，且这时我们得知条件，check会且只会在x 在 xs里面时才会return True。同时powers 1 是一个2的倍数的order list，因此在条件x 会且只会在xs里才return True，而check n又是将n与输入的list按顺序逐个对比，同时会在list里面的值比x大时return False，因此我们可以说，pow2 n 就是一个将数字与2的倍数比较的function，也就是和power2的功能一样。

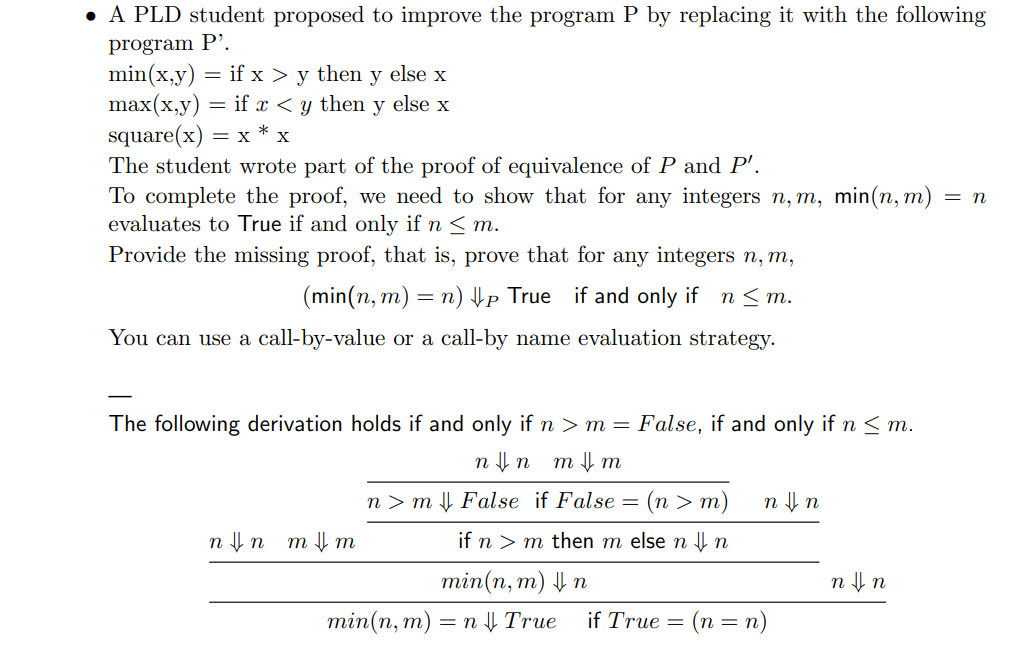
# **Call by value/name 几乎必考**



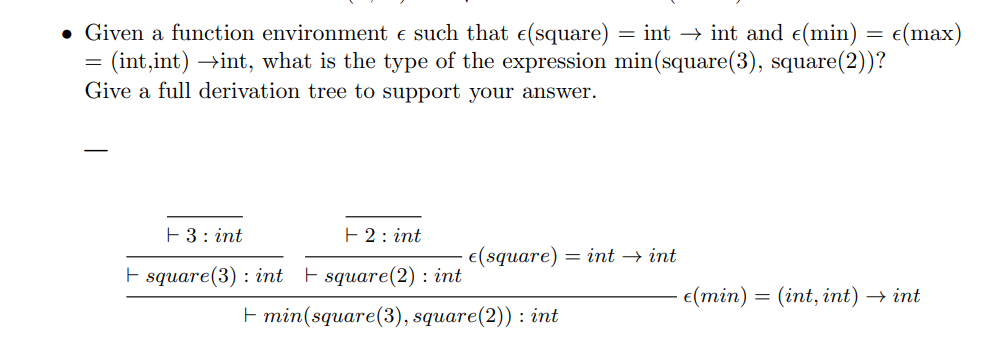




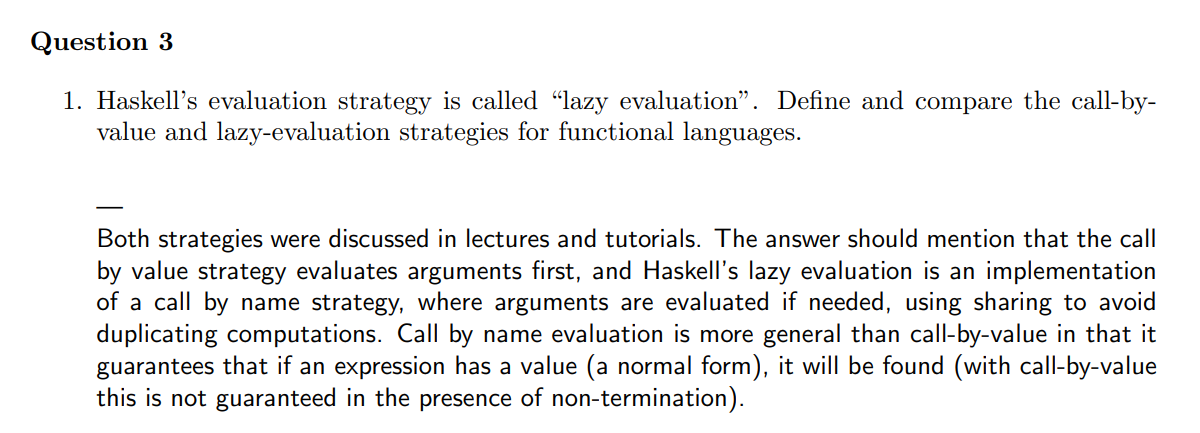
(2021)



这算是一种变体吧

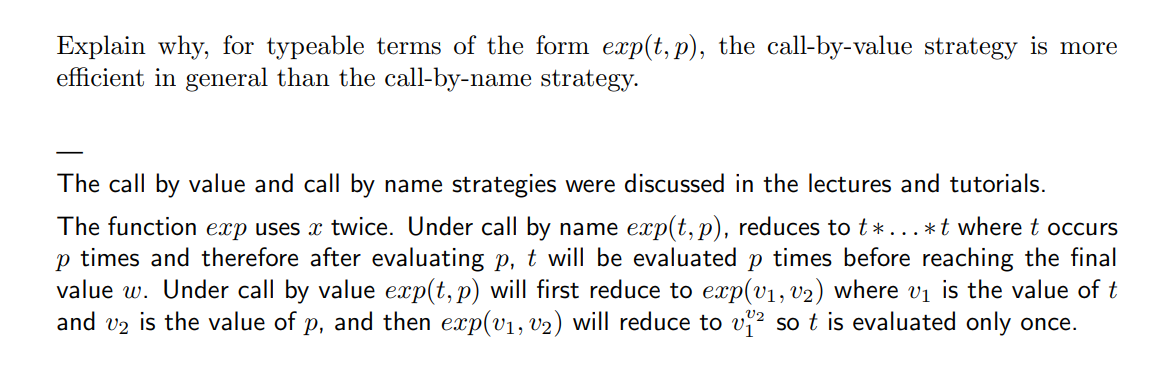
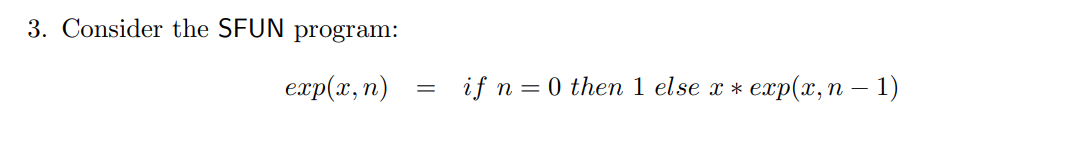


2015



这里考的是lazy evaluation和call by value，其实也就是call by name 和 call by value的区别。

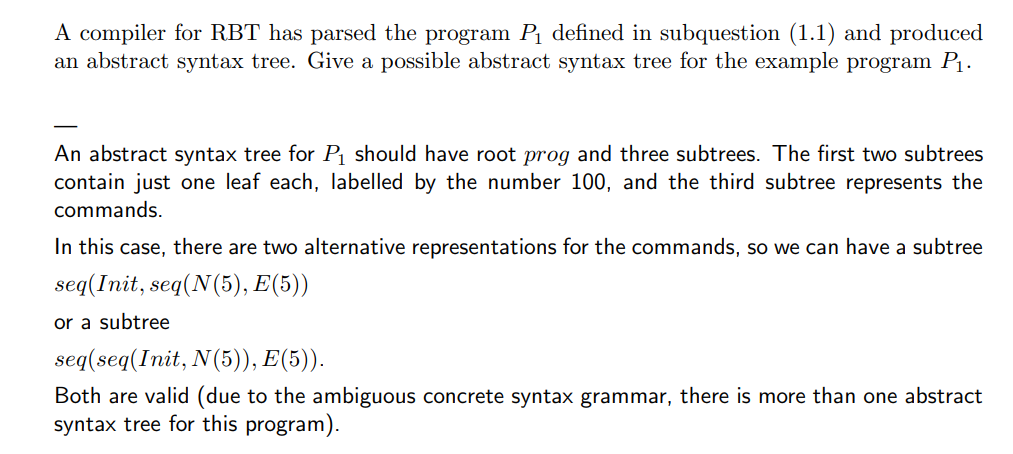
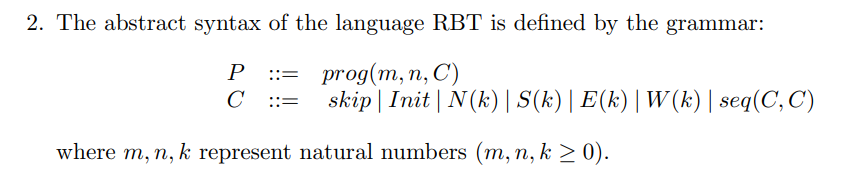
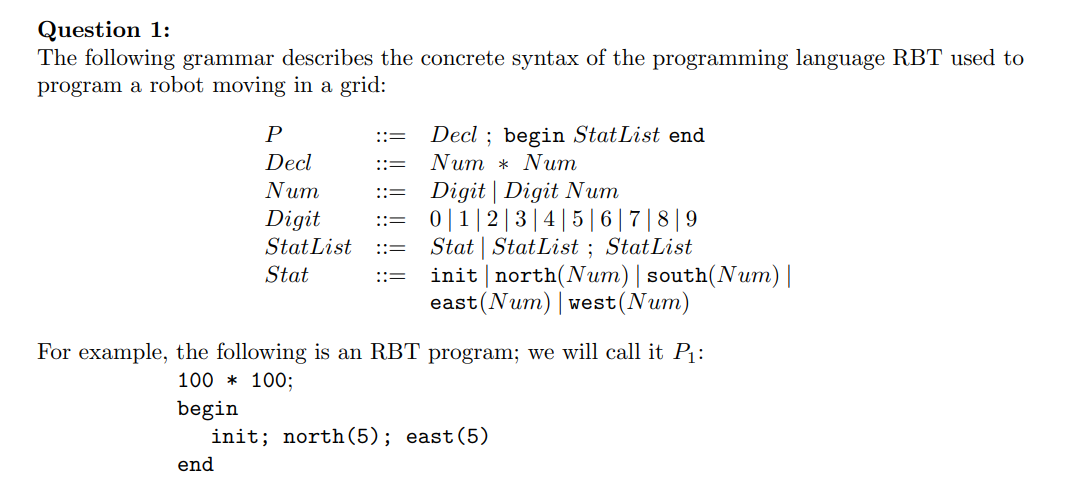
2015



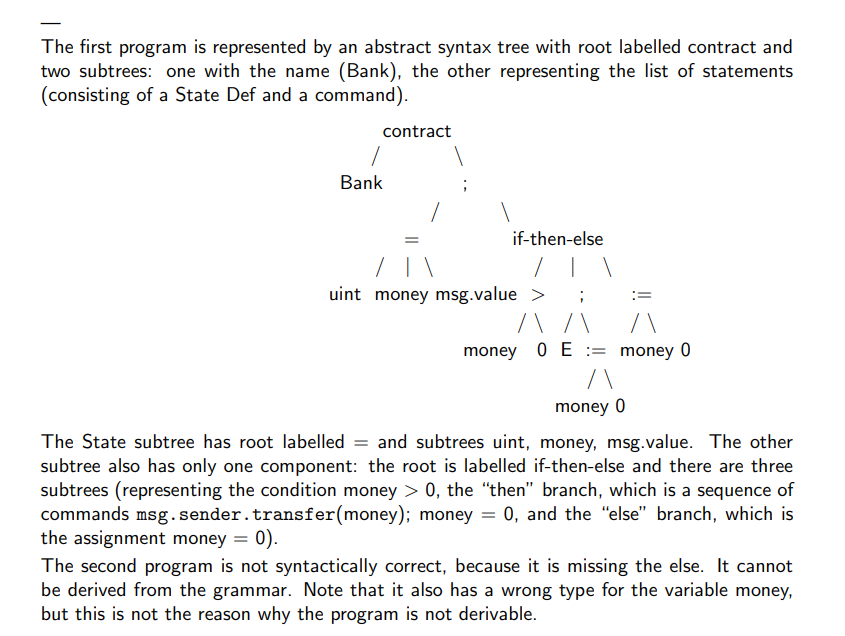
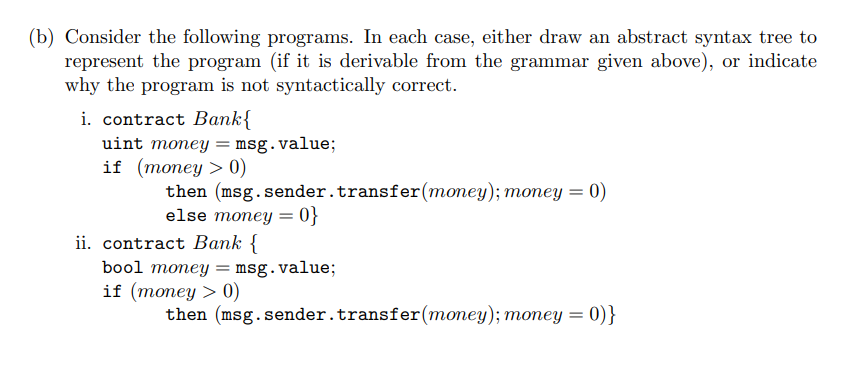
# 

# **Abstract Syntax Tree 几乎必考**

2017

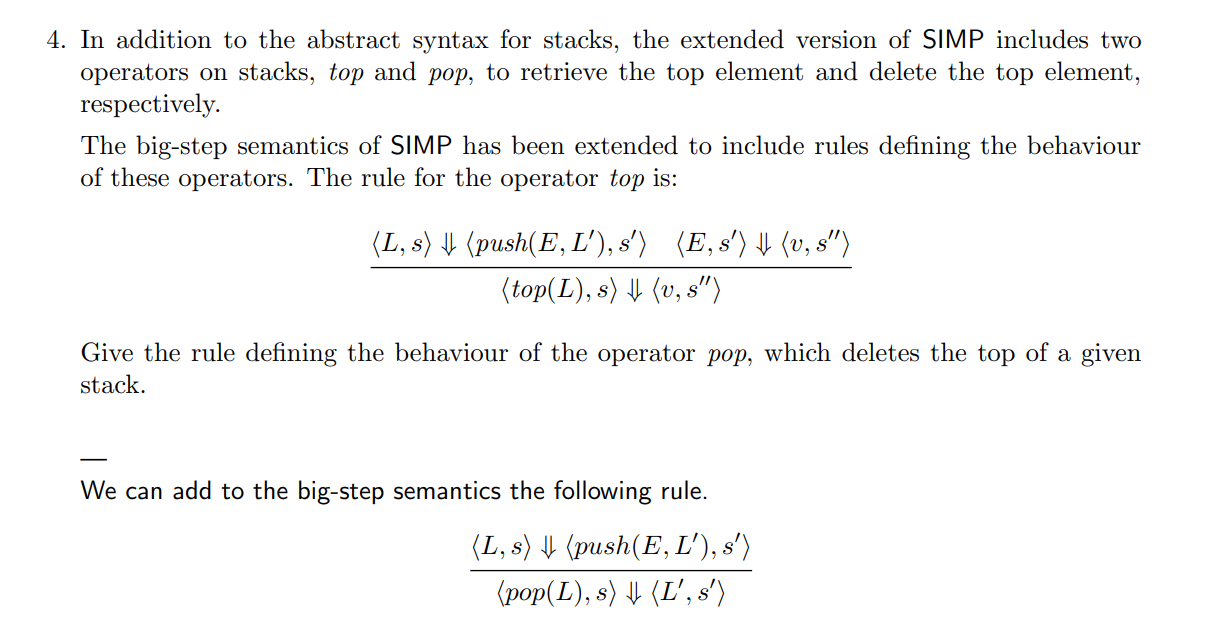
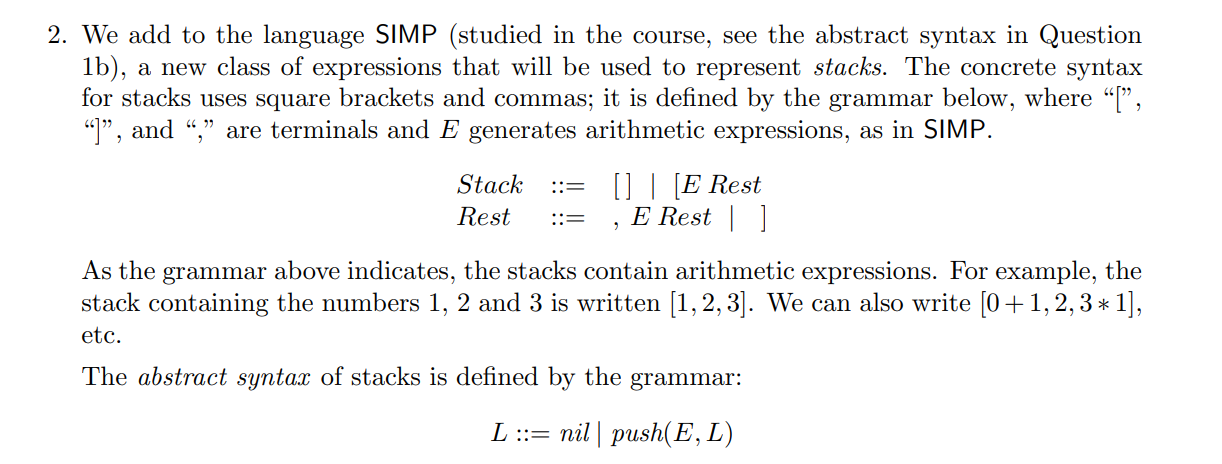


这一题和普通的abstract syntax tree 不太一样，主要体现在这次他有自己定义的abstract syntax 而不是常见的那种公式，因此这次的root是他自己定义的prog，同时m n都是100，C则是他定义的command



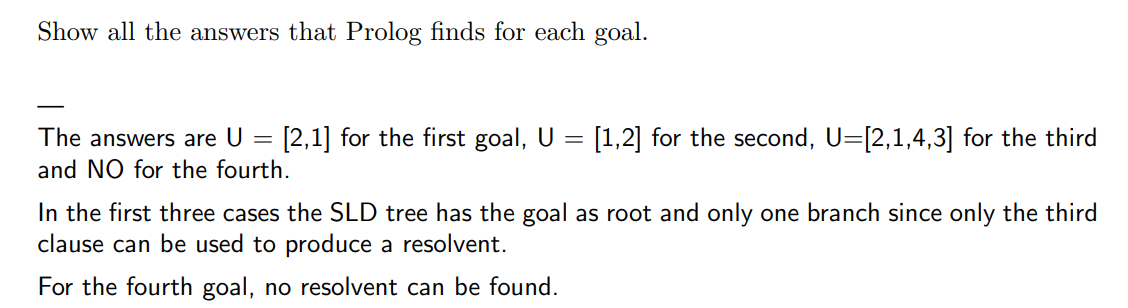
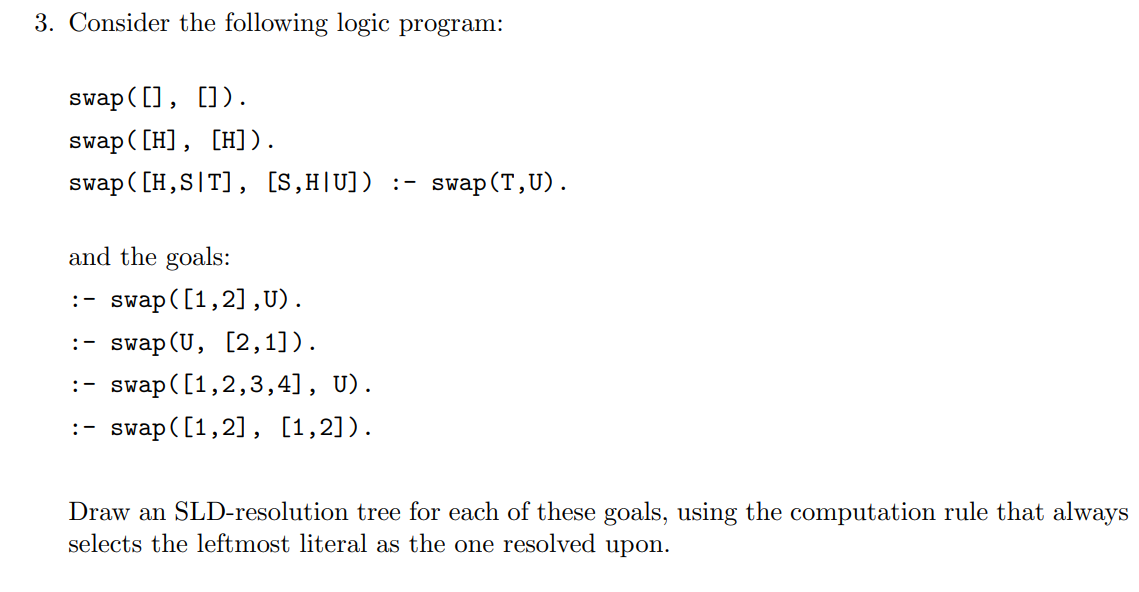
# **Big semantics**

2015

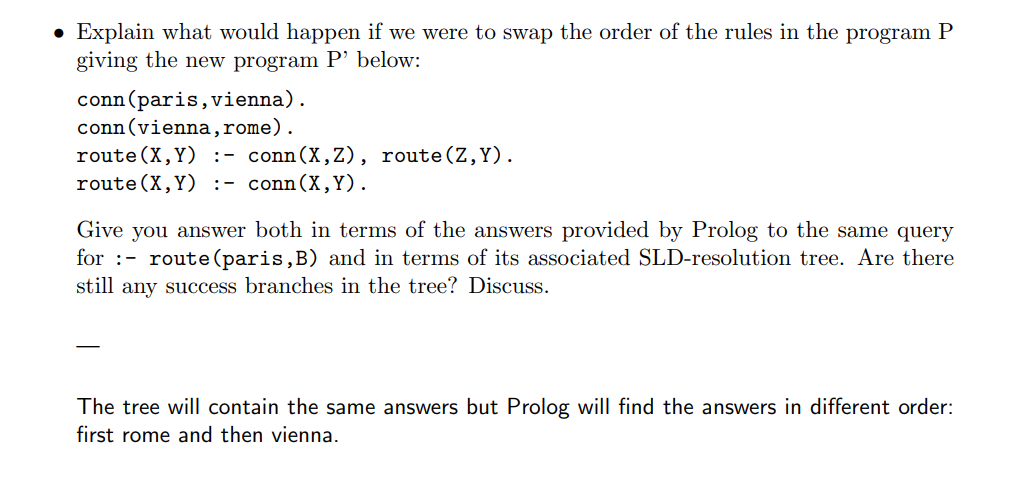
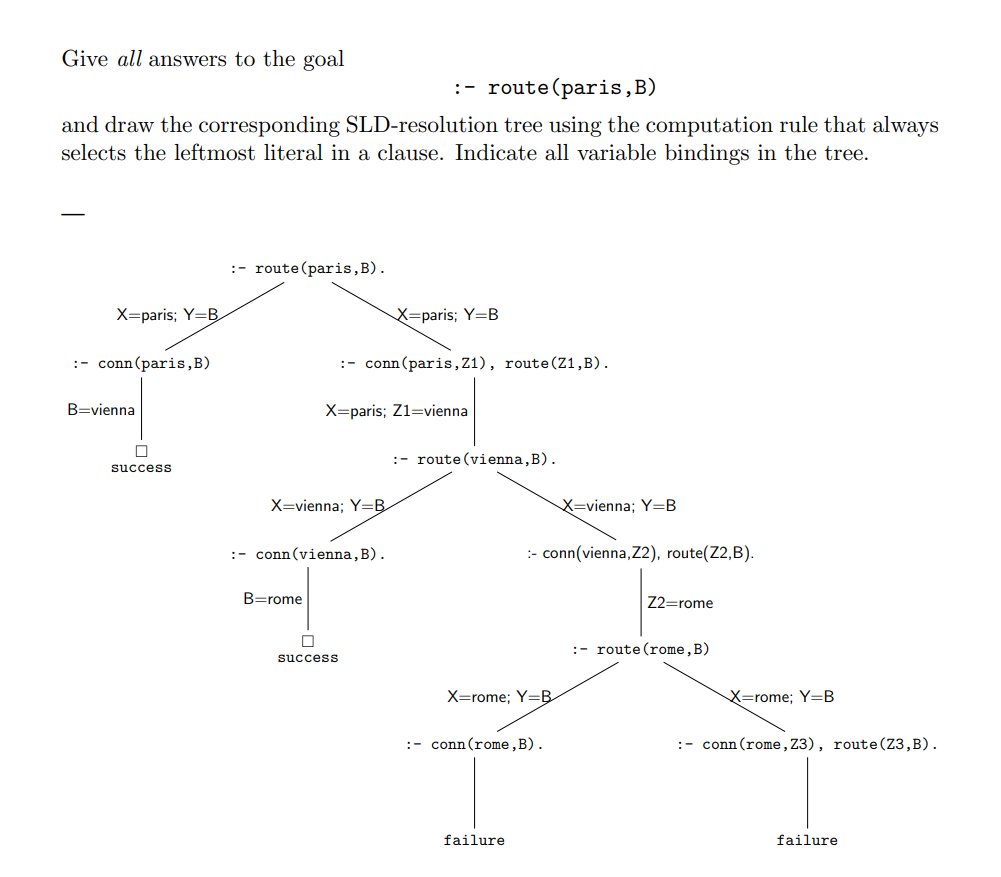
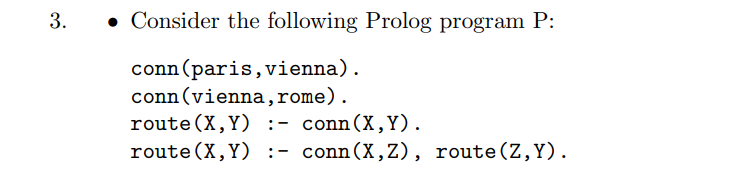


这题的解题思路和上面差不多，只不过上面提取的是E，而这里提取的是L’，同时因为不需要像E一样提取出value，因此我们直接return L’,s’就行

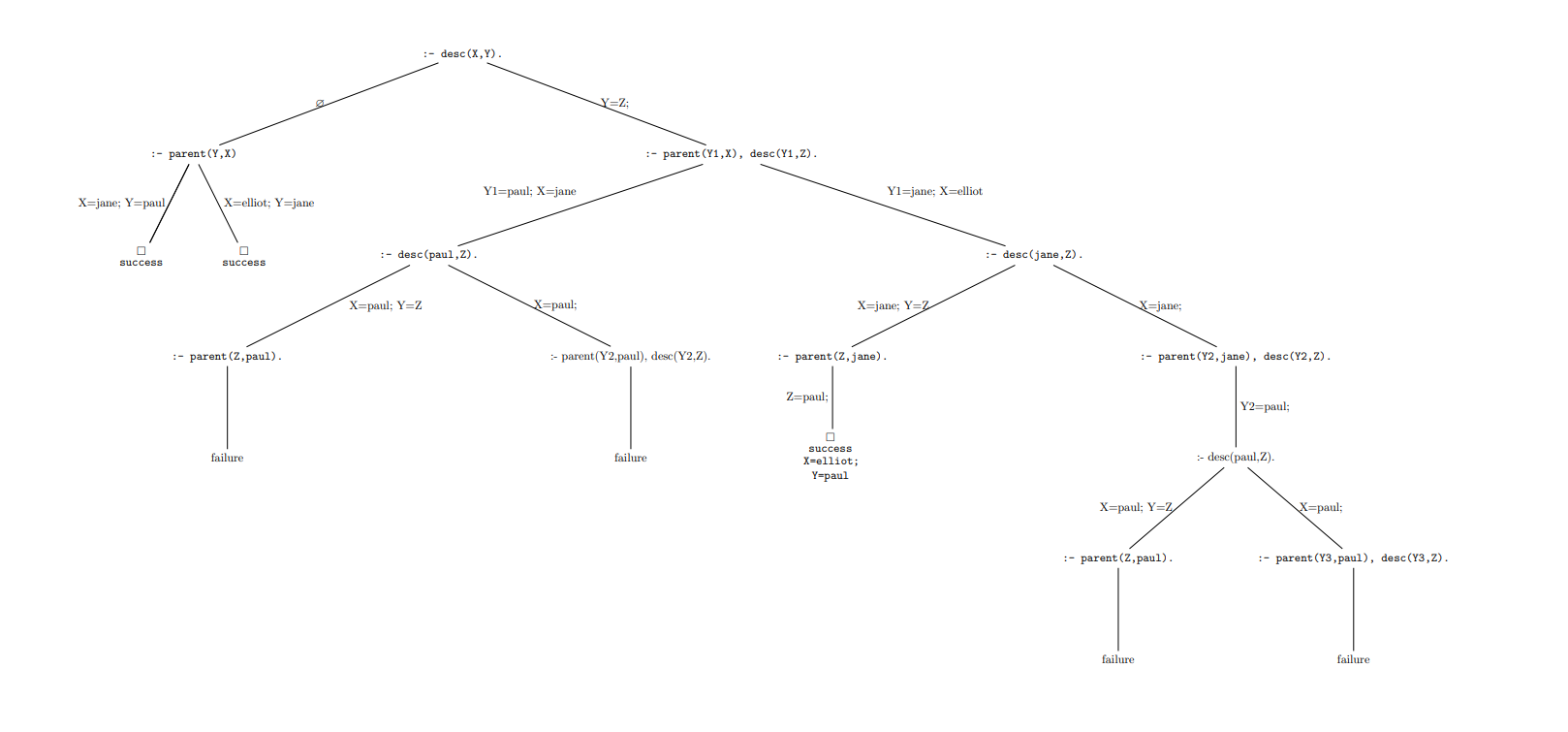
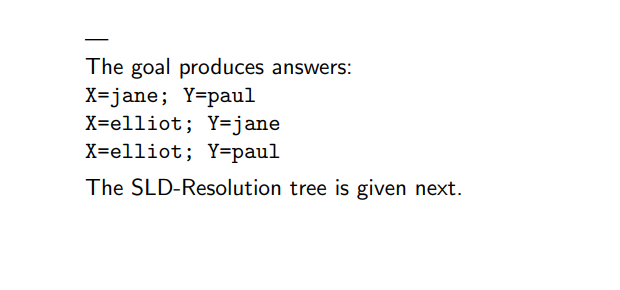
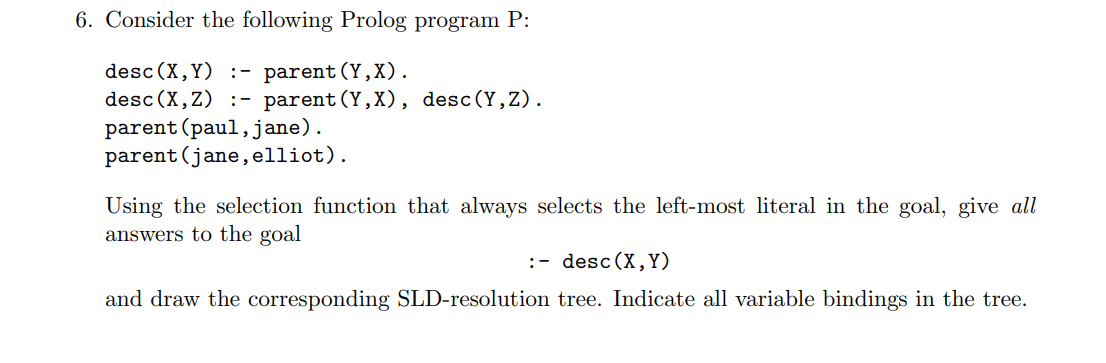
# **SLD resolution tree 几乎必考**



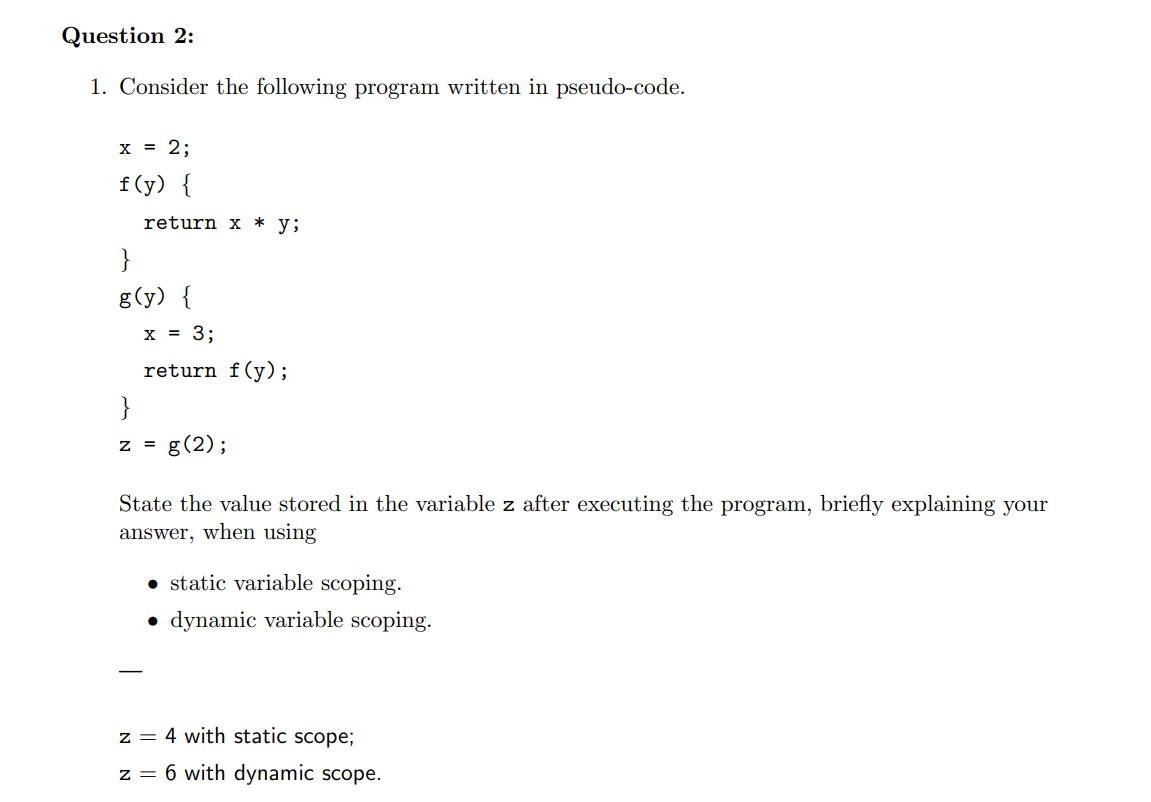
非常classic的SLD resolution tree 问题，前三个过程简单，最后一个之所以不是也是因为没有对的上的，首先[1,2]不等于[]，其次[1,2]有两个数，不符合[H]要求的只有一个。最后[1,2][1,2]不符合3的要求也就是[1,2][2,1]



和上面一页的题一样，这就是迷惑你的，让你以为可能会loop之类的，实际试过就会发现还是可以输入结果，唯一的区别是顺序变了。



# **Static and dynamic variable**



在static variable中，static variable scoping 限制了这个variable的使用范围。比如在外部定义的x，g和f都可以使用这个x = 2。但是此时g改写了x = 3，可是这个改写只是locally over written，并不会影响外部的x = 2，因此当f使用x时，x还是2。

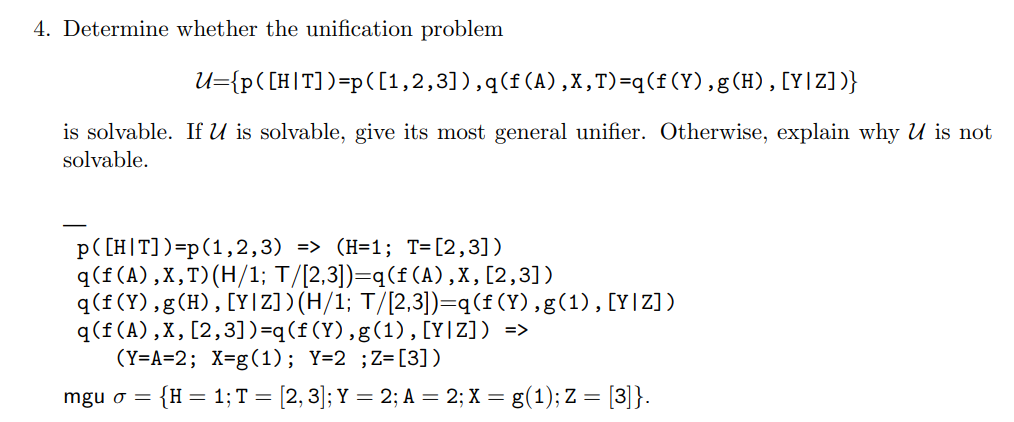
相对的dynamic variable中，g中改写的x是可以影响后面的f使用它的

# **Most general unifier**

这一题比较值得注意的是（c）和（f）

其中c是因为虽然前面是 H 和 T 会让人想到head和tail但是实际上，H和T可以是任何东西，因为他们只是一个variable而已本身不代表任何东西。因此H和T可以两个都是List

f 则是一个比较classic的例子，只要他是按顺序变的就可以转换，并不影响



这一题要求你直接给出mgu的值，也是比较特殊的

# 特殊题



加了if then else，但是没有break之前的逻辑，如果B，则还是while B do C  
