Question 1

1.1 Recursive to Iterative CPS Transformations

b. Prove that append$ is CPS-equivalent to append. That is, for lists lst1 and lst2 and a continuation procedure cont, (append$ lst1 lst2 cont) = (cont (append lst1 lst2)). Prove the claim by induction (on the length of the first list), and using the applicative-eval operational semantics. Write your proof in ex5.pdf.

Proposition: For any lists lst1 and lst2 and a continuation procedure cont, (append$ lst1 lst2 cont) = (cont (append lst1 lst2)).

Proof: By induction on the length of lst1

Base: For the case of a lst1 of length 0 [the empty list], the value of (append lst1 lst2) is lst2, and the value of (append$ lst1 lst2 cont) is (cont lst2), which implies (append$ lst1 lst2 cont) = (cont (append lst1 lst2)).

Induction step: We assume the proposition holds for lst1 of length n, and show the proposition holds for lst1 of a length n+1.

(a) According to the code, the value of (append lst1 lst2) is (cons (car lst1) (append (cdr lst1) lst2)).
(b) According to the code ,the value of (append$ lst1 lst2 cont) is (append$ (cdr lst1) lst2 cont2), where cont2 is the continuation procedure defined in lines 6-7.

Since the first operand of (append$ (cdr lst1) lst2 cont2) is a list of length n, according to the induction assumption: (cont2 (append (cdr lst1) lst2)) = (append$ (cdr lst1) lst2 cont2).

- ⇨ (cont (cons (car lst1) (append (cdr lst1) lst2))) = (append$ (cdr lst1) lst2 cont2) ;;; code of cont 2
- ⇨ (cont (append lst1 lst2)) = (append$ (cdr lst1) lst2 cont2) ;;; (a)
- ⇨ (cont (append lst1 lst2)) = (append$ lst1 lst2 cont) ;;; (b)

Question 3 - Logic programing

3.1 Unification

What is the result of these operations? Provide algorithm steps, and explain in case of failure.

1. unify[t(s(s), G, s, p, t(K), s), t(s(G), G, s, p, t(K), U)]

1. s = {} , A = t(s(s), G, s, p, t(K), s) , B = t(s(G), G, s, p, t(K), U)
2. s = {G = s}

   A∘s = t(s(s), s, s, p, t(K), s)

   B∘s = t(s(s), s, s, p, t(K), U)
3. s = {G = s, s = U}

   A∘s = t(s(s), s, s, p, t(K), s)

   B∘s = t(s(s), s, s, p, t(K), s)

Answer → s = {G = s, U = s}

2. unify[g(l,M,g,G,U,g,v(M)), g(l,v(U),g,v(M),v(G),g,v(M))]

1. s = {}

   A = g(l, M, g, G, U, g, v(M))

   B = g(l, v(U), g, v(M), v(G), g, v(M))
2. s = {M = v(U)}

   s∘A = g(l, v(U), g, G, U, g, v(v(U)))

   s∘B = g(l, v(U), g, v(v(U)), v(G), g, v(v(U)))
3. s = {M = v(U), G = v(v(U))}

   s∘A = g(l, v(U), g, v(v(U)), U, g, v(v(U)))

   s∘B = g(l, v(U), g, v(v(U)), v(v(v(U))), g, v(v(U)))
4. s = {M = v(U), G = v(v(U)), U = v(v(v(U)))} → illegal substitution

Answer → no such substition.

3. unify[m(M,N), n(M,N)]

1. $s = \{\}$

   $A = m(M,N)$

   $B = n(M,N)$

2. the first elements are different ("m" and "n"), so unification fails.

   answer $\Rightarrow$ no such substition.

4. unify[p([v | [V | VV]]), p([[v | V] | VV])]

1. $s = \{\}$

   $A = p([v|[V|VV]])$

   $B = p([[v|V]|VV])$

2. $s = \{v = [v|V]\}$ $\Rightarrow$ illegal substitution

   answer $\Rightarrow$ no such substition.

5. unify[g([T]), g(T)]

1. $s = \{\}$

   $A = g([T])$

   $B = g(T)$

2. $s = \{g([T]) = g(T)\}$ $\Rightarrow$ illegal substitution

   answer $\Rightarrow$ no such substition.

Logic programming

3.3 Proof tree

Draw the proof tree for the query:

?- le(X,s(zero)),times(X,s(s(zero)),Y).