```
function SethiUllman(t, n)
    pre 0 ≤ n ≤ K − min(K, need[t]) ∧ K ≥ 2 ∧ need well formed
    post (reg[t] = r_i ∧ 0 ≤ i < K ⇒ need[t] > 0)∧
           (need[t] > 0 ⇒ reg[t] = r_n)∧
           code has been emitted to evaluate t into reg[t],
           modifying only registers r_i such that n ≤ i < K
    if t has two children, u_left and u_right
        if need[u_left] ≥ K ∧ need[u_right] ≥ K
            assert n = 0
            SethiUllman(u_right, 0)
            emit instruction to store reg[u_right]
            SethiUllman(u_left, 0)
            reg[u_right] ← "r_1"
            emit instruction to fetch reg[u_right]
        else if need[u_left] ≥ need[u_right]
            assert need[u_right] < min(K, need[t])
            SethiUllman(u_left, n)
            SethiUllman(u_right, n + 1)
        else need[u_left] < need[u_right]
            assert need[u_left] < min(K, need[t])
            SethiUllman(u_right, n)
            SethiUllman(u_left, n + 1)
        reg[t] ← "r_n"
        emit OPER(instruction[t], reg[t], [reg[u_left], reg[u_right]])
    else if t has one child, u
        SethiUllman(u, n)
        reg[t] ← "r_n"
        emit OPER(instruction[t], reg[t], [reg[u]])
    else if t is nontrivial but has no children
        reg[t] ← "r_n"
        emit OPER(instruction[t], reg[t], [])
    else t is a trivial node TEMP(r_i)
        reg[t] ← "r_i"
```