**(define remove-cps**

**(lambda (item ls k)**

**(if (null? ls)**

**(k '())**

**(remove-cps item (cdr ls)**

**(if (eq? item (car ls))**

**k**

**(lambda (v) (k (cons (car ls) v))))))))**

**(define union-cps**

**(lambda (los1 los2 k)**

**(if (null? los1)**

**(k los2)**

**(union-cps (cdr los1) los2**

**(lambda (v)**

**(remove-cps (car los1) v**

**(lambda (v)**

**(k (cons (car los1) v)))))))))**

**(define free-vars-cps**

**(lambda (exp k)**

**(cond [(symbol? exp)**

**(k (list exp))]**

**[(eq? (1st exp) 'lambda)**

**(free-vars-cps (3rd exp)**

**(lambda (body-free-vars)**

**(remove-cps (car (2nd exp)) body-free-vars k)))]**

**[else (free-vars-cps (1st exp)**

**(lambda (rator-free-vars)**

**(free-vars-cps (2nd exp)**

**(lambda (rand-free-vars)**

**(union-cps rator-free-vars rand-free-vars k)))))])))**