;Problem 1

(define apply-continuation (lambda (k v) (k v)))

(define member?-cps (lambda (sym ls k)

(cond

[(null? ls) (apply-continuation k #f)]

[(equal? sym (car ls)) (apply-continuation k #t)]

[else (member?-cps sym (cdr ls) k)])))

(define make-cps

(lambda (proc)

(lambda (arg k)

(apply-continuation k (proc arg)))))

(define map-cps

(lambda (proc-cps ls k)

(cond [(null? ls) (apply-continuation k '())]

[else

(map-cps proc-cps (cdr ls)

(lambda (cdr-result)

(proc-cps (car ls)

(lambda (proc-result)

(apply-continuation k

(cons proc-result cdr-result))))))])))

(define andmap-cps

(lambda (pred-cps ls k)

(cond

[(null? ls) (apply-continuation k '#t)]

[else (pred-cps (car ls)(lambda (proc-result)

(if proc-result

(andmap-cps pred-cps (cdr ls)

(lambda (cdr-result)

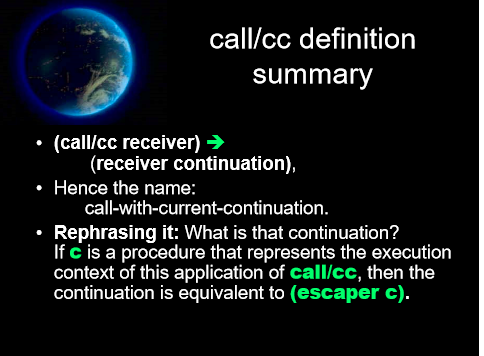
(apply-continuation k cdr-result)))

(apply-continuation k '#f))))])))

(define append-cps

(lambda (ls1 ls2 k)

(cond [(null? ls2) (apply-continuation k ls1)]

 [else

(append-cps

(reverse (cons (car ls2) (reverse ls1)))

(cdr ls2)

k)])))

