```lisp
(in-package "CL-USER")


(defun do-comp (e sl)
  (let ((res (comp-01 (expand e) nil))
        (globs nil))
    (setf res (comp-11 res))
    (format t "~%")
    (pprint (car res))
    (setf res (comp-2 (expand (car res)) nil))
    (format t "comp-2 = ~S~%" res)
    (pprint res)
    (format t "~%~%")
    (setf res (comp-11 (expand res)))
    (format t "comp-2 analyzed = ")
    (pprint (car res))
    (format t "~%~%")
    (setf res (comp-3 (car res) nil))
    (setf *xx* res)
    (pprint res)))

(defun do-comp-1 (e sl)
  (let ((res (comp-11 (expand e)))
        (globs nil))
    (setf res (comp-2 (expand (car res)) nil))
    (format t "comp-2 = ~S~%" res)
    (pprint res)))




(defun comp-x (e)
  (let ((res nil))
    (setf res (expand e))
    (setf res (comp-11 res))
    (pprint res)
    (format t "~%************~%")
    (setf res (comp-2 (car res) nil))
    (pprint res)
    (format t "~%************~%")
    (setf res (comp-11 (expand res)))
    (pprint res)
    (format t "~%************~%")
    (setf res (comp-3 (car res) nil))
    (setf res (expand res))
    (pprint res)
    (format t "~%************~%")
    (stat-init *stat*)
    (comp-gen res *stat*)
    (comp-gen-embrace *stat*)
    (setf *tt* (make-pretemplate *stat*))
    (format t "*stat* = ~A~%" *stat*)))


(defun tst1 (xx)
  (setf steps 100000)
  (setf *tst00* xx)
  (comp-x *tst00*)
  (setf tp (msys::cl-to *tt*))
  (emu::emu-init tp)
  (emu::run-processor steps))


(defparameter *tstgb* '(progn
```



```lisp
(defun list (&rest x) x)
(defun not (x) (if x nil 1))

(defun deriv-aux (a) (list '/ (deriv a) a))

(defun cadr (x) (car (cdr x)))
(defun caddr (x) (cadr (cdr x)))
(setq t 1)

(defun mapcar (fun lis)
  (cond ((eq lis nil) nil)
        (1 (cons (funcall fun (car lis)) (mapcar fun (cdr lis))))))

(defun deriv (a)
   (cond
         ((not (consp a))
          (if (eq a 'x) 1 0))
         ((eq (car a) '*)
          (list '* a (cons '+ (mapcar #'(lambda (z) (deriv-aux z)) (cdr a)))))
         ((eq (car a) '+)
          (cons '+ (mapcar #'(lambda (z) (deriv z))(cdr a))))
         ((eq (car a) '-)
          (cons '- (mapcar #'(lambda (z) (deriv z)) (cdr a))))
         ((eq (car a) '/)
          (list '-
                (list '/ (deriv (cadr a)) (caddr a))
                (list '/
                      (cadr a)
                      (list '* (caddr a) (caddr a) (deriv (caddr a))))))
         (t 'error)))

(%print (caddr '(1 2 3)))

(defun deriv-run ()
  (let ()
        (%print (deriv '(+ (* 3 x x) (* a x x) (* b x) 5)))
        (deriv '(+ (* 3 x x) (* a x x) (* b x) 5))
        (deriv '(+ (* 3 x x) (* a x x) (* b x) 5))
        (deriv '(+ (* 3 x x) (* a x x) (* b x) 5))
        (deriv '(+ (* 3 x x) (* a x x) (* b x) 5))))

(deriv-run)))




(defparameter *tstxa* '(let* ((a 1)
                             (b (%+ 10 a))
                             (c (%* b b)))
                        (%print a)
                        (%print b)
                        (%print c)))

(defparameter *tstxb* '(let ()
                        (defun cc (x) (cons x x))
                        (defun mapcar (f l)
                          (cond ((eq l nil) nil)
                                (1 (cons (funcall f (car l)) (mapcar f (cdr l))
))))
                        (let ((ccc #'(lambda (x) (cc (cc x)))))
                          (%print (mapcar #'ccc '(1 2 3 4 5 6))))))

(defparameter *tstxc* '(let ((a 1))
                        (setq a a)
```



```lisp
                        (%print a)))

(defparameter *tstxd* '(let ((a 1)(b nil)) (let ((c 3)(d 4)) (%print (%+ a c)) (%
print (cons b d)))))


(defparameter *tstxz* '(progn
                        (let ((a 1) (b 2))
                          (defun adder (x) #'(lambda (a) (%+ x a)))
                          (setq add3 (adder 3))
                          (setq add12 (adder 12))
                          (%print (funcall add3 7))
                          (%print (funcall add12 18)))))




(defparameter *tstxy* '(progn
                        (defun f (a b c) (cons (cons a b) c))
                        (defun lis (&rest x) x)
                        (%print (lis 1 2 3 4 5 6))
                        (%print (apply f (lis 1 2 3)))))

(defparameter *tstxx* '(progn
                        (defun f (a b c) (cons (cons a b) c))
                        (defun lis (&rest x) x)
                        (%print (lis 1 2 3 4 5 6))
                        (%print (apply f (apply f 1 '(77 88)) 3 (lis (apply f 1
 '(5 6)))))))


(defparameter *tst01* '(progn
                        (defun f (n) (cond ((%<= n 1) 1)
                                           (1 (%+ (f (%- n 1)) (f (%- n 2))))))
                        (defun g (n) (cond ((%<= n 0) nil)
                                           (1 (cons (f n) (g (%- n 1))))))
                        (defun app (l x) (cond ((eq l nil) (cons x nil))
                                               (1 (cons (car l) (app (cdr l) x)
))))
                        (defun rev (x) (cond ((eq x nil) nil)
                                             (1 (app (rev (cdr x)) (car x)))))
                        (%print (rev '(-1 -2 "juergen" -4 -5)))
                        (defun mapcar (f l) (cond ((eq l nil) nil)
                                                  (1 (cons (funcall f (car l)) (m
apcar f (cdr l))))))
                        (%print (mapcar #'(lambda (z) (cons "juergen" (%* z -1200
00))) '(1 2 3 4 5)))
                        (%print (mapcar #'(lambda (z) (cons z z)) '(#\v #\n #\r
 #\g)))
                        (%print (mapcar #'(lambda (z) (apply app z)) '(((1 2 3
4) 99) ((5 6 7 8) 33))))))


(defparameter *tst00* '(labels ((f (n) (%* n n)) (g (n) (%+ (f n) (f n))))
                        (%print (g 12))))

(defparameter *tst02*
  '(progn
     (setq x 10)
     (let ((a 1))
     (labels ((f (n) (cond ((%<= n a) a)
                           (1 (%+ (f (%- n 1)) (f (%- n 2))))))
              (g (n) (cond ((%<= n 1) nil)
                           (1 (cons (f n) (g (%- n 1)))))))
       (%print (g x))
```



```lisp
       (%print (g 5))))))


(defparameter *tst03*
  '(progn (labels ((f (n) (%* n n)))
             (setq ff #'(lambda (z) (f z))))
     (%print (funcall ff 12))))



;;;; (defparameter *tst00* '(let ((a 1)
;;;;                              (b 2)
;;;;                              (c 3))
;;;;                          (let ((d 4))
;;;;                            (%print (vector a b c d)))))


(defparameter *tst04*
  '(progn
     (defun f (a &rest b) (cons a (cons b nil)))
     (%print (f 1 2))
     (let ((a 1) (b 10) (c 100))
       (%print (%* b (%- a c))))))


(defparameter *tst05*
  '(let ((a 1) (b 2))
     (if (and (%= a b) (%= a a)) (%print a) (%print b))
     (if (and (%= a a) (%= a b)) (%print a) (%print b))
     (if (and (%= a a) (%= b b)) (%print a) (%print b))

     (if (or (%= a b) (%= a a)) (%print a) (%print b))
     (if (or (%= a a) (%= a b)) (%print a) (%print b))
     (if (or (%/= a a) (%/= b b)) (%print a) (%print b))))



(defparameter *tst06*
  '(let ((a 1))
     (%print (do-until ((%>= a 10) (%* a a)) (%print a) (setq a (%+ a 1))))))



(defparameter *tst* '(let ((x 1) (y 2))
                      (let ((u 3) (v 4))
                        (let ((p 5))
                          (let ()
                            (setq lp (lambda (t1 t2)
                                       (let ()
                                         (setq t1 t2)
                                         (setq u t1)
                                         (setq cc t1)
                                         (setq dd u)
                                         (setq t1 ee)
                                         (setq t2 v))))
                            (setq x u)
                            (setq aa x)
                            (setq y bb)
                            (setq lp1 (lambda (t) (setq ff t))))))))

(defparameter *tst1* '(defun f (n) (cond ((<= n 1) 1) (1 (* 2 (f  (- n 1)))))))

(defparameter *tst2* '(lambda (a) (defun f (n) (cond ((<= n 1) (setq x 1) (setq y
b) (setq a a)) (1 (* a f (- n 1)))))))
```

```
(defparameter *tst3* '(let ((x 1)(ff nil)) (let ((i 0) (ff1 nil))
                                          (setq ff1 (do-until ((%> i 5) ff)
                                                      (let ((y 1))
                                                        (setq i (%+ i 1))
                                                        (setq ff (cons #'(lambda (z)
                                                                           (cond ((%
= z 0) x)
                                                                                 (1
 (setq x z)))) ff)))))
                                          (%print 12)
                                          (%print ff1)
                                          (%print 33)
                                          (%print (funcall (car (cdr ff)) 22))
                                          (%print (funcall (car ff) 0))
                                          (%print (funcall (car (cdr (cdr ff))) 57
))
                                          (%print (funcall (car (cdr ff)) 0)))))



(defparameter *tst4* '(let ((x 1) (y 2))
                         (let ((u 3) (v 4))
                           (let ((p 5))
                             (let ()
                               (setq lp (lambda (t1 t2)
                                          (let ()
                                            (setq t1 t2)
                                            (setq u t1)
                                            (setq cc t1)
                                            (setq dd u)
                                            (setq t1 ee)
                                            (setq t2 v))))
                               (setq x u)
                               (setq aa x)
                               (setq y bb)
                               (setq lp1 (lambda (t) (setq ff tt))))))))

(defparameter *tst5* '(lambda (a b c)
                         (labels ((f (m n) (progn (setq a m) (setq a c) (setq n b
)))
                                  (g (r s) (progn (setq r s) (setq f s) (setq r g
))))
                           (progn (f p q) (g a b) (setq f (lambda (t) (setq t g))
)))))

(defparameter *tst51* '(lambda (a b c)
                          (labels ((f (m n) (progn (setq a m) (setq a c) (setq n
b)))
                                   (g (r s) (progn (setq r #'(lambda (t) (setq a s
)))  (setq f s) (setq r g))))
                           (progn (f p q) (g a b) (setq c (lambda (t) (setq t g))
)))))

(defparameter *tst6* '(progn (defun h (a)
                         (labels ((f (n) (cond ((%<= n 1) 1)
                                               (1 (%+ (f (%- n 1)) (f (%- n 2))))
))
                                  (g (n) (cond ((%<= n 1) nil)
                                               (1 (cons (f n) (g (%- n 1))))))) (
g a))) (h 6)))

(defparameter *tst61* '(defun h (a)
                         (labels ((f (n) (cond ((%<= n 1) 1)
                                               (1 (%+ (f (%- n 1)) (f (%- n 2))))
))
```



```
                                  (g (k) (cond ((%<= k 1) nil)
                                               (1 (cons (f k) (g (%- k 1))))))) (
g a))))

(defparameter *tst62* '(defun h (a &rest a1)
                          (labels ((f (n &rest n1) (cond ((<= n 1) 1)
                                                         (1 (+ (f (- n 1)) (f (- n 2))))))
                                   (g (k) (cond ((<= k 1) nil)
                                                (1 (cons (f k) (g (- k 1))))))) (g
 a))))

(defparameter *tst7* '(defun h (a)
                          (labels ((f (n) (cond ((<= n 1) 1)
                                                (1 (+ (foo (- n 1)) (bar (- n 2)))
)))
                                   (g (n) (cond ((<= n 1) nil)
                                                (1 (cons (f n) (g (- n 1))))))) (g
 a))))
```