8-1/ (define (fac n)

J_2: (if (= n 0) 1 (* n (fac (- n 1))))

(fac 5)

J_3: let fac = μ^{ifac} (n) (if (= n 0) 1

J_4: (* n (ifac (- n 1)))) in

(fac 5)

J_3: λ (x ...) e

$J_3 \rightarrow J_4$ J_4: λ x (x ...) e

↑ recursive

8-2/ $E[(l\ v_0 \dots v_n)] = E[e[f \leftarrow l][x_0 \leftarrow v_0] \dots [x_n \leftarrow v_n]]$

where $l = (\lambda\ f\ (x_0 \dots x_n)\ e)$

$$\langle \lambda f\ (x \dots)\ e\ ,\ env\ ,\ k \rangle$$

$$\mapsto \langle c\ ,\ \emptyset\ ,\ k \rangle$$

where $c = clo(\lambda f\ (x \dots)\ e,\ env')$

$env' = env[f \mapsto c]$

8-3/

```
switch ( type (c) ) {
  case LAMBDA:
    envp = make_env (env, c->recname, NULL);
    c = make_clo ( c , envp )
    envp -> val = c;
    env = NULL;
    break;
```

8-4)

desugar ["lambda", [x_0, ..., x_n], e_b]

OLD = (λ (x_0 ... x_n) desugar(e_b))

NEW = (λ rec (x_0 ... x_n) desugar(e_b))

desugar ["lambda", f, [x_0 ... x_n], e_b]

= (λ f (x_0, ..., x_n) desugar (e_b))

8-5/

```
nat-unfold f z n :=
  if (= n 0) z
     (f n (nat-unfold f z (subi n)))
```

desugar

```
do-times x ec ans ed eb) :=
  let last := ec in
  ((λ repeat (x ans)
     (if (x < last)
        (repeat (+ x 1) eb)
        ans))
   0 ed)
```

8-6]

```
do-times i 5 sum 0
         (+ i sum)
```

⇒ "macro"

```
let last = 5 in
((λ repeat (i sum)
   (if (< i last)
       (repeat (+ i 1) (+ i sum))
       sum))
 0 0)
```