17-1) clo-conv : $R_5 \Rightarrow R_4$

```
cloconv : e -> e x list(def)
cloconv (Num n) = (Num n) x []
cloconv (Add l r) = (Add l' r') x (lds ++ rds)
    where l', lds = cloconv l
          r', rds = cloconv r
cloconv (App rator rands) =
  (Let (Id f) rator'
    (App (VectorRef (Id f) 0)
         (cons (VectorRef (Id f) 1)
               rands')))  x  (rator_ds ++ rands_ds)
where rator', rator_ds = cloconv rator
      rands', rand_ds = map cloconv rands
cloconv (FunRef f) =
   (Vector (FunRef f) Unit)  x  []
cloconv (Define f rty dom body) =
  (Define f rty (cons (Param "_" Unit) dom)
    body') x body_ds
where body', body_ds = cloconv body
```

```
cloconv (Lambda r rng dom body) =
  (Vector (FunRef nf)
          (Vector x0 ... xn)) x ds'
where
clon, nf = a fresh names
  body', body_ds = cloconv body
  ds' = nfd : body_ds
  nfd = (Define nf rng (clo_arg : dom) body'')
  clo_arg = (Pair clon clo_ty)
  clo_ty = (VectorTy t0 ... tn)
  body'' = (Let* ((Pair r (vector (FunRef nf) (Id clon)))
                  (Pair x0 (VectorRef (Id clon) 0))
                  ....)
                 body')
(x0 : t0) ... (xn : tn) are free variables in body
```

17-3)

```
cloconv (Program info ds e) =
  (Program info ds'' e')
where
    ds', nds0 = map cloconv ds
    e', nds1 = cloconv e
    ds'' = ds' ++ nds0 ++ nds1
```