```
(do-test-group |records| :BEFORE (PROGN (SETQ S (QUOTE (FIRST SECOND THIRD)))
(SETQ ALFA "some string")) :AFTER (PROGN (IL:RECORD RECORD-TEST-NAME)
(IL:RECORD RECORD-TEST-NAME1) (IL:RECORD RECORD-TEST-NAME2))

;; record type record

(DO-TEST |setup-record|
     (IL:RECORD RECORD-TEST-NAME
         (ALPHA BRAVO GAMMA)
         (IL:SYNONYM ALPHA A)
         (IL:TYPE? (ODDP (LENGTH IL:DATUM)))))

(DO-TEST |create-record|
      (SETQ RECORD-TEST-RECORD
   (IL:|create| RECORD-TEST-NAME ALPHA IL:_ ALFA BRAVO IL:_ S)))

(DO-TEST |type?-record|
       (IL:|type?| RECORD-TEST-NAME RECORD-TEST-RECORD))

(DO-TEST SYNONYM-record
       (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) ALFA))

(DO-TEST |fetch-record|
       (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) ALFA)
            (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST |replace-record|
       (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                  IL:WITH S) S)
           (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                  IL:WITH ALFA) ALFA)))

(DO-TEST |refetch-record|
       (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) S)
            (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) ALFA)
            (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST rereplace-record
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                         IL:WITH ALFA) ALFA)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                         IL:WITH S) S)))

(DO-TEST |typeglobalvariable-record|
       (EQ (SYMBOL-PACKAGE (IL:\\TYPEGLOBALVARIABLE
            (QUOTE RECORD-TEST-NAME)))
            (FIND-PACKAGE "XCL-TEST")))

(DO-TEST |using-record|
        (SETQ RECORD-TEST-RECORD3
        (IL:CREATE RECORD-TEST-NAME
            IL:USING RECORD-TEST-RECORD GAMMA IL:_ S))
        (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
                 (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
             (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                 (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))
             (EQ (IL:FETCH GAMMA IL:OF RECORD-TEST-RECORD3)
                  S)))

(DO-TEST |reusing-record|
          (SETQ RECORD-TEST-RECORD3
              (IL:CREATE RECORD-TEST-NAME
                  IL:REUSING RECORD-TEST-RECORD))
         (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
```

```
                (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
             (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                 (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))))

(DO-TEST |copying-record|
          (SETQ RECORD-TEST-RECORD2
             (IL:CREATE RECORD-TEST-NAME
                  IL:COPYING RECORD-TEST-RECORD))
         (AND (EQUAL (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                     (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2))
             (NOT (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                 (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2)))))

(DO-TEST |smashing-record|
        (SETQ RECORD-TEST-RECORD4 (IL:CREATE RECORD-TEST-NAME
              IL:SMASHING RECORD-TEST-RECORD2))
        (AND (NULL (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD2))
           (EQ RECORD-TEST-RECORD4 RECORD-TEST-RECORD2) ))

; record type typerecord

(DO-TEST |setup-typerecord|
          (IL:TYPERECORD RECORD-TEST-NAME (ALPHA BRAVO GAMMA)
                (IL:SYNONYM ALPHA A)))

(DO-TEST |create-typerecord|
       (SETQ RECORD-TEST-RECORD
       (IL:|create| RECORD-TEST-NAME
              ALPHA IL:_ ALFA BRAVO IL:_ S)))


(DO-TEST |type?-typerecord|
         (IL:|type?| RECORD-TEST-NAME RECORD-TEST-RECORD))

(DO-TEST SYNONYM-typerecord
          (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) ALFA))

(DO-TEST |fetch-typerecord|
       (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) ALFA)
            (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST |replace-typerecord|
      (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                  IL:WITH S) S)
           (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                  IL:WITH ALFA) ALFA)))

(DO-TEST |refetch-typerecord|
       (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) S)
            (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) ALFA)
            (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST rereplace-typerecord
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                       IL:WITH ALFA) ALFA)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                     IL:WITH S) S)))

(DO-TEST |typeglobalvariable-typerecord|
        (EQ (SYMBOL-PACKAGE (IL:\\TYPEGLOBALVARIABLE
          (QUOTE RECORD-TEST-NAME))) (FIND-PACKAGE "XCL-TEST")))

(DO-TEST |using-typerecord|
     (SETQ RECORD-TEST-RECORD3
```

```
       (IL:CREATE RECORD-TEST-NAME IL:USING RECORD-TEST-RECORD
        GAMMA IL:_ S))
     (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
              (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
          (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))
          (EQ (IL:FETCH GAMMA IL:OF RECORD-TEST-RECORD3) S)))

(DO-TEST |reusing-typerecord|
   (SETQ RECORD-TEST-RECORD3
     (IL:CREATE RECORD-TEST-NAME IL:REUSING RECORD-TEST-RECORD))
     (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
              (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
          (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
              (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))))

(DO-TEST |copying-typerecord|
     (SETQ RECORD-TEST-RECORD2
          (IL:CREATE RECORD-TEST-NAME
               IL:COPYING RECORD-TEST-RECORD))
     (AND (EQUAL (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                 (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2))
          (NOT (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                   (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2)))))

(DO-TEST |smashing-typerecord|
    (SETQ RECORD-TEST-RECORD4 (IL:CREATE RECORD-TEST-NAME
         IL:SMASHING RECORD-TEST-RECORD2))
    (AND (NULL (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD2))
         (EQ  RECORD-TEST-RECORD2 RECORD-TEST-RECORD4)))


;record type proprecord

(DO-TEST |setup-proprecord|
    (IL:PROPRECORD RECORD-TEST-NAME (ALPHA BRAVO GAMMA)
        (IL:SYNONYM ALPHA A)
        (IL:TYPE? (EVENP (LENGTH IL:DATUM)))))

(DO-TEST |create-proprecord|
   (SETQ RECORD-TEST-RECORD
      (IL:|create| RECORD-TEST-NAME
           ALPHA IL:_ ALFA BRAVO IL:_ S)))


(DO-TEST |type?-proprecord|
    (IL:|type?| RECORD-TEST-NAME RECORD-TEST-RECORD))

(DO-TEST SYNONYM-proprecord
    (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) ALFA))

(DO-TEST |fetch-proprecord|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) ALFA)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST |replace-proprecord|
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                         IL:WITH S) S)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                         IL:WITH ALFA) ALFA)))

(DO-TEST |refetch-proprecord|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) S)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) ALFA)
```

```
         (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST rereplace-proprecord
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                         IL:WITH ALFA) ALFA)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                          IL:WITH S) S)))

(DO-TEST |typeglobalvariable-proprecord|
    (EQ (SYMBOL-PACKAGE (IL:\\TYPEGLOBALVARIABLE
                              (QUOTE RECORD-TEST-NAME)))
        (FIND-PACKAGE "XCL-TEST")))



(DO-TEST |using-proprecord|
    (SETQ RECORD-TEST-RECORD3
       (IL:CREATE RECORD-TEST-NAME IL:USING RECORD-TEST-RECORD
        GAMMA IL:_ S))
   (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
            (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
        (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                  (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))
       (EQ (IL:FETCH GAMMA IL:OF RECORD-TEST-RECORD3) S)))

(DO-TEST |reusing-proprecord|
    (SETQ RECORD-TEST-RECORD3
        (IL:CREATE RECORD-TEST-NAME
           IL:REUSING RECORD-TEST-RECORD))
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
             (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
             (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))))

(DO-TEST |copying-proprecord|
    (SETQ RECORD-TEST-RECORD2
        (IL:CREATE RECORD-TEST-NAME
            IL:COPYING RECORD-TEST-RECORD))
    (AND (EQUAL (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2))
         (NOT (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                  (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2)))))

(DO-TEST |smashing-proprecord|
    (SETQ RECORD-TEST-RECORD4 (IL:CREATE RECORD-TEST-NAME
         IL:SMASHING RECORD-TEST-RECORD2))
    (AND (NULL (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD2))
         (EQ RECORD-TEST-RECORD2 RECORD-TEST-RECORD4)))


;record type datatype

(DO-TEST |setup-datatype|
    (IL:DATATYPE RECORD-TEST-NAME (ALPHA BRAVO GAMMA)
        (IL:SYNONYM ALPHA A)))

(DO-TEST |create-datatype|
    (SETQ RECORD-TEST-RECORD
      (IL:|create| RECORD-TEST-NAME
          ALPHA IL:_ ALFA BRAVO IL:_ S)))


(DO-TEST |type?-datatype|
    (IL:|type?| RECORD-TEST-NAME RECORD-TEST-RECORD))
```

```
(DO-TEST SYNONYM-datatype
    (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) ALFA))

(DO-TEST |fetch-datatype|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) ALFA)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST |replace-datatype|
     (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                          IL:WITH S) S)
          (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                           IL:WITH ALFA) ALFA)))

(DO-TEST |refetch-datatype|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) S)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) ALFA)
         (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) S)))


(DO-TEST |typeglobalvariable-datatype|
    (EQ (SYMBOL-PACKAGE (IL:\\TYPEGLOBALVARIABLE
                         (QUOTE RECORD-TEST-NAME)))
        (FIND-PACKAGE "XCL-TEST")))


(DO-TEST fetchfield-datatype
    (AND (EQ (IL:FETCHFIELD (CAR (IL:GETDESCRIPTORS
             (QUOTE RECORD-TEST-NAME))) RECORD-TEST-RECORD) S)
         (EQ (IL:FETCHFIELD (CADR (IL:GETDESCRIPTORS
             (QUOTE RECORD-TEST-NAME))) RECORD-TEST-RECORD)
            ALFA)))

(DO-TEST replacefield-datatype
    (AND (EQ (IL:REPLACEFIELD (CAR (IL:GETDESCRIPTORS
                (QUOTE RECORD-TEST-NAME)))
               RECORD-TEST-RECORD ALFA) ALFA)
         (EQ (IL:REPLACEFIELD (CADR (IL:GETDESCRIPTORS
                (QUOTE RECORD-TEST-NAME)))
                RECORD-TEST-RECORD S) S)))

(DO-TEST refetchfield-datatype
     (AND (EQ (IL:FETCHFIELD (CAR (IL:GETDESCRIPTORS
                 (QUOTE RECORD-TEST-NAME)))
                 RECORD-TEST-RECORD) ALFA)
          (EQ (IL:FETCHFIELD (CADR (IL:GETDESCRIPTORS
                 (QUOTE RECORD-TEST-NAME)))
                 RECORD-TEST-RECORD) S)))

(DO-TEST getfieldspecs-datatype
     (EQ (CAR (IL:GETFIELDSPECS (QUOTE RECORD-TEST-NAME)))
         (CADDAR (IL:GETDESCRIPTORS (QUOTE RECORD-TEST-NAME)))))

(DO-TEST IL:typename-datatype
    (EQ (IL:TYPENAME RECORD-TEST-RECORD)
        (QUOTE RECORD-TEST-NAME)))

(DO-TEST typenamep-datatype
    (IL:TYPENAMEP RECORD-TEST-RECORD (QUOTE RECORD-TEST-NAME)))

(DO-TEST |using-datatype|
    (SETQ RECORD-TEST-RECORD3
        (IL:CREATE RECORD-TEST-NAME
            IL:USING RECORD-TEST-RECORD GAMMA IL:_ S))
```

```
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
             (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                  (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))
         (EQ (IL:FETCH GAMMA IL:OF RECORD-TEST-RECORD3) S)))


(DO-TEST |reusing-datatype|
    (SETQ RECORD-TEST-RECORD3
        (IL:CREATE RECORD-TEST-NAME
           IL:REUSING RECORD-TEST-RECORD))
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
             (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
             (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))))


(DO-TEST |copying-datatype|
    (SETQ RECORD-TEST-RECORD2
        (IL:CREATE RECORD-TEST-NAME
           IL:COPYING RECORD-TEST-RECORD))
    (AND (EQUAL (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2))
         (NOT (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                  (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2)))))

(DO-TEST |smashing-datatype|
    (SETQ RECORD-TEST-RECORD4 (IL:CREATE RECORD-TEST-NAME
        IL:SMASHING RECORD-TEST-RECORD2))
    (AND (NULL (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD2))
         (EQ RECORD-TEST-RECORD2 RECORD-TEST-RECORD4)))


;record type arrayrecord

(DO-TEST |setup-arrayrecord|
    (IL:ARRAYRECORD RECORD-TEST-NAME (ALPHA BRAVO GAMMA)
        (IL:SYNONYM ALPHA A)
        (IL:TYPE? (COND (IL:DATUM T)))))

(DO-TEST |create-arrayrecord|
    (SETQ RECORD-TEST-RECORD
       (IL:|create| RECORD-TEST-NAME
          ALPHA IL:_ ALFA BRAVO IL:_ S)))


(DO-TEST |type?-arrayrecord|
    (IL:|type?| RECORD-TEST-NAME RECORD-TEST-RECORD))

(DO-TEST SYNONYM-typearary
    (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) ALFA))

(DO-TEST |fetch-arrayrecord|
'     (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) ALFA)
           (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST |replace-arrayrecord|
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                         IL:WITH S) S)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                         IL:WITH ALFA) ALFA)))

(DO-TEST |refetch-arrayrecord|
```

```
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) S)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) ALFA)
         (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST rereplace-arrayrecord
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                         IL:WITH ALFA) ALFA)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                          IL:WITH S) S)))

(DO-TEST |typeglobalvariable-arrayrecord|
    (EQ (SYMBOL-PACKAGE (IL:\\TYPEGLOBALVARIABLE
                         (QUOTE RECORD-TEST-NAME)))
        (FIND-PACKAGE "XCL-TEST")))

(DO-TEST getfieldspecs-arrayrecord
    (EQ (CAR (IL:GETFIELDSPECS (QUOTE RECORD-TEST-NAME)))
        (CADDAR (IL:GETDESCRIPTORS (QUOTE RECORD-TEST-NAME)))))

(DO-TEST IL:typename-arrayrecord
    (EQ (IL:TYPENAME RECORD-TEST-RECORD)
        (QUOTE il:arrayp)))

(DO-TEST typenamep-arrayrecord
     (IL:TYPENAMEP RECORD-TEST-RECORD (QUOTE il:arrayp)))


(DO-TEST |using-arrayrecord|
    (SETQ RECORD-TEST-RECORD3
          (IL:CREATE RECORD-TEST-NAME
              IL:USING RECORD-TEST-RECORD GAMMA IL:_ S))
   (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
            (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
        (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                 (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))
        (EQ (IL:FETCH GAMMA IL:OF RECORD-TEST-RECORD3) S)))


(DO-TEST |reusing-arrayrecord|
    (SETQ RECORD-TEST-RECORD3
         (IL:CREATE RECORD-TEST-NAME
             IL:REUSING RECORD-TEST-RECORD))
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
             (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
             (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))))


(DO-TEST |copying-arrayrecord|
    (SETQ RECORD-TEST-RECORD2
         (IL:CREATE RECORD-TEST-NAME
             IL:COPYING RECORD-TEST-RECORD))
    (AND (EQUAL (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2))
         (NOT (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                  (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2)))))


(DO-TEST |smashing-arrayrecord|
    (SETQ RECORD-TEST-RECORD4 (IL:CREATE RECORD-TEST-NAME
       IL:SMASHING RECORD-TEST-RECORD2))
    (AND (NULL (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD2))
         (EQ RECORD-TEST-RECORD2 RECORD-TEST-RECORD4)))
```

```
;record type assocrecord

(DO-TEST |setup-assocrecord|
    (IL:ASSOCRECORD RECORD-TEST-NAME (ALPHA BRAVO GAMMA)
       (IL:SYNONYM ALPHA A)
       (IL:TYPE? (NOT (IL:ATOM (CAR IL:DATUM))))))


(DO-TEST |create-assocrecord|
   (SETQ RECORD-TEST-RECORD (IL:|create| RECORD-TEST-NAME
         ALPHA IL:_ ALFA BRAVO IL:_ S)))


(DO-TEST |type?-assocrecord|
    (IL:|type?| RECORD-TEST-NAME RECORD-TEST-RECORD))

(DO-TEST synonym-assocrecord
    (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) ALFA))

(DO-TEST |fetch-assocrecord|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) ALFA)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST |replace-assocrecord|
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                         IL:WITH S) S)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                         IL:WITH ALFA) ALFA)))

(DO-TEST |refetch-assocrecord|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) S)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) ALFA)
         (EQ (IL:FETCH A IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST rereplace-assocrecord
    (AND (EQ (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                         IL:WITH ALFA) ALFA)
         (EQ (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                         IL:WITH S) S)))

(DO-TEST |typeglobalvariable-assocrecord|
    (EQ (SYMBOL-PACKAGE (IL:\\TYPEGLOBALVARIABLE
                        (QUOTE RECORD-TEST-NAME)))
        (FIND-PACKAGE "XCL-TEST")))

(DO-TEST |using-assocrecord|
    (SETQ RECORD-TEST-RECORD3
       (IL:CREATE RECORD-TEST-NAME
          IL:USING RECORD-TEST-RECORD GAMMA IL:_ S))
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
             (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                  (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))
         (EQ (IL:FETCH GAMMA IL:OF RECORD-TEST-RECORD3) S)))


(DO-TEST |reusing-assocrecord|
    (SETQ RECORD-TEST-RECORD3
        (IL:CREATE RECORD-TEST-NAME
           IL:REUSING RECORD-TEST-RECORD))
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD)
             (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD3))
```

```
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
             (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD3))))


(DO-TEST |copying-assocrecord|
    (SETQ RECORD-TEST-RECORD2
        (IL:CREATE RECORD-TEST-NAME
            IL:COPYING RECORD-TEST-RECORD))
   (AND (EQUAL (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
               (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2))
        (NOT (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD)
                 (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD2)))))

(DO-TEST |smashing-assocrecord|
    (SETQ RECORD-TEST-RECORD4 (IL:CREATE RECORD-TEST-NAME
       IL:SMASHING RECORD-TEST-RECORD2))
    (AND (NULL (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD2))
         (EQ RECORD-TEST-RECORD2 RECORD-TEST-RECORD4)))

;record type accessfns

(DO-TEST setup-accessfns
    (IL:ACCESSFNS RECORD-TEST-NAME
           ((ALPHA (CAR IL:DATUM)
                   (SETQ IL:DATUM(CONS IL:NEWVALUE
                                       (CDR IL:DATUM))))
            (BRAVO (CADR IL:DATUM)
                   (SETQ IL:DATUM (CONS (CAR IL:DATUM)
                                        (CONS IL:NEWVALUE
                                              (CDDR IL:DATUM)))))
            (GAMMA (CADDR IL:DATUM)
                   (SETQ IL:DATUM (LIST (CAR IL:DATUM)
                                        (CADR IL:DATUM)
                                        IL:NEWVALUE))))
          (IL:CREATE (LIST ALFA S NIL))
          (IL:TYPE? (ODDP (LENGTH IL:DATUM)))))

(DO-TEST create-accessfns
    (SETQ RECORD-TEST-RECORD
        (IL:create RECORD-TEST-NAME)))

(DO-TEST |type?|
    (IL:|type?| RECORD-TEST-NAME RECORD-TEST-RECORD))


(DO-TEST |fetch-accessfns|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) ALFA)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST |replace-accessfns|
    (AND (IL:REPLACE ALPHA IL:OF RECORD-TEST-RECORD
                     IL:WITH S)
         (IL:REPLACE BRAVO IL:OF RECORD-TEST-RECORD
                     IL:WITH ALFA)))

(DO-TEST |refetch-accessfns|
    (AND (EQ (IL:FETCH ALPHA IL:OF RECORD-TEST-RECORD) S)
         (EQ (IL:FETCH BRAVO IL:OF RECORD-TEST-RECORD) ALFA) ))

(DO-TEST |typeglobalvariable-accessfns|
    (EQ (SYMBOL-PACKAGE (IL:\\TYPEGLOBALVARIABLE
                         (QUOTE RECORD-TEST-NAME)))
        (FIND-PACKAGE "XCL-TEST")))
```

```
; blockrecords

(DO-TEST setup-blockrecord
    (IL:DATATYPE RECORD-TEST-NAME1
           ((ALPHA IL:POINTER)) ALPHA IL:_ S)
    (IL:BLOCKRECORD RECORD-TEST-NAME2
           ((BRAVO IL:WORD) (GAMMA IL:WORD)))
    (SETQ RECORD-TEST-RECORD (IL:CREATE RECORD-TEST-NAME1)))

(DO-TEST TEST-FETCH-BLOCKRECORD
    (AND (EQ (IL:FETCH (RECORD-TEST-NAME1 ALPHA)
                       IL:OF RECORD-TEST-RECORD)
             (IL:\\VAG2 (IL:FETCH (RECORD-TEST-NAME2 BRAVO)
                                  IL:OF RECORD-TEST-RECORD)
                        (IL:FETCH (RECORD-TEST-NAME2 GAMMA)
                                  IL:OF RECORD-TEST-RECORD)))
          (EQ (IL:FETCH (RECORD-TEST-NAME1 ALPHA)
                        IL:OF RECORD-TEST-RECORD) S)))

(DO-TEST TEST-REPLACE-BLOCKRECORD
    (IL:REPLACE (RECORD-TEST-NAME1 ALPHA) IL:OF RECORD-TEST-RECORD IL:WITH
ALFA))

(DO-TEST  TEST-reFETCH-BLOCKRECORD
    (AND (EQ (IL:FETCH (RECORD-TEST-NAME1 ALPHA)
                       IL:OF RECORD-TEST-RECORD)
             (IL:\\VAG2 (IL:FETCH (RECORD-TEST-NAME2 BRAVO)
                                  IL:OF RECORD-TEST-RECORD)
                        (IL:FETCH (RECORD-TEST-NAME2 GAMMA)
                                  IL:OF RECORD-TEST-RECORD)))
          (EQ (IL:FETCH (RECORD-TEST-NAME1 ALPHA)
                        IL:OF RECORD-TEST-RECORD) ALFA)))

(DO-TEST "TEST THAT REPLACES THROUGH THE BLOCKRECORD STRUCTURE"
    (IL:REPLACE (RECORD-TEST-NAME2 BRAVO) IL:OF RECORD-TEST-RECORD
                IL:WITH (IL:\\HILOC S))
    (IL:REPLACE (RECORD-TEST-NAME2 GAMMA) IL:OF RECORD-TEST-RECORD
                IL:WITH (IL:\\LOLOC S)))

(DO-TEST "TEST REFETCHING AFTER REPLACING THROUGH THE BLOCKRECORD"
    (AND (EQ (IL:FETCH (RECORD-TEST-NAME1 ALPHA)
                       IL:OF RECORD-TEST-RECORD)
             (IL:\\VAG2 (IL:FETCH (RECORD-TEST-NAME2 BRAVO)
                                  IL:OF RECORD-TEST-RECORD)
                        (IL:FETCH (RECORD-TEST-NAME2 GAMMA)
                                  IL:OF RECORD-TEST-RECORD)))
          (EQ (IL:FETCH (RECORD-TEST-NAME1 ALPHA)
                        IL:OF RECORD-TEST-RECORD) S)))

(Do-test "look at floating point"
    (IL:DATATYPE flnum ((n IL:floating)))
    (setq num1 (IL:CREATE flnum))
    (setq num2 (IL:CREATE flnum))
    (IL:BLOCKRECORD fldisect
           ((sign IL:BITS 1) (exp IL:BITS 8) (mant IL:BITS 23)))
    (setq anynum (IL:RAND))
   (IL:REPLACE n IL:of num1 IL:with anynum)
   (IL:REPLACE n IL:of num2 IL:with (IL:times anynum 2))
   (eq (IL:add1 (IL:fetch exp IL:of num1))
       (IL:fetch exp IL:of num2)))
```

```
(Do-test "test blank fields and playing with integers"
     (IL:DATATYPE intnum ((int IL:integer)))
     (setq num (IL:CREATE intnum))
     (IL:BLOCKRECORD evenodd ((nil IL:bits 16)
                              (nil IL:BITS 15)
                             (lastbit IL:BITS 1)))
     (setq anynum (IL:RAND))
     (IL:REPLACE int IL:of num IL:with anynum)
     (if (evenp (IL:fetch int IL:of num))
            (progn (IL:replace lastbit IL:of num IL:with 1)
                   (oddp (IL:fetch int IL:of num)))
            (progn (IL:replace lastbit IL:of num IL:with 0)
                   (evenp (IL:fetch int IL:of num)))))

;Testing WITH

(Do-test "simple with using a datatype"
       (IL:with flnum num1
             (IL:setq n 0)
             (zerop n)))

(Do-test "compound with using two  datatypes"
    (IL:with flnum num1
          (IL:with intnum num
              (IL:setq n (il:times n 2))
               (IL:setq int 0)
               (and (equal (float int) n)
                    (zerop int)))))

) ;END OF DO-TEST-GROUP

STOP
```