

File created: 2-Aug-88 21:52:05 {ERIS}<TEST>GC>HAND>MAIKO-GC-TESTS.;7

changes to: (FNS MAIN-GC-TEST LIST-MANIPULATION-TEST CODE-RECLAIM-TEST)

previous date: 23-Jun-88 16:06:34 {ERIS}<TEST>GC>HAND>MAIKO-GC-TESTS.;6

Read Table: XCL

Package: INTERLISP

Format: XCCS

; Copyright (c) 1988 by John Sybalsky & Xerox Corporation. All rights reserved.

### (RPAQQ MAIKO-GC-TESTSCOMS

```
((FILES DANCEROBJ GCHAX)
 (ADDVARS (DISPLAYFONTDIRECTORIES "{ERIS}<TEST>GC>HAND-AUX>" "{ERIS}<LISPCORE>XEROXPRIVATE>FONTS>")
 (INTERPRESSFONTDIRECTORIES "{ERIS}<TEST>GC>HAND-AUX>" "{ERIS}<LISPCORE>XEROXPRIVATE>FONTS>"))
 (P (SETQ DISPLAYFONTEXTENSIONS '(DISPLAYFONT AC STRIKE)))
 (FNS MAIN-GC-TEST)
 (FNS ITEMS-ON-STACK-TEST MANY-BIGNUM-MAKER MANY-FIXP-MAKER MANY-FLOAT-MAKER BOUNDARY-TESTS
 ARRAY-STRING-TEST VARIOUS-TYPES-TEST)
 (FNS TEDIT-CRUNCH-TEST LIST-MANIPULATION-TEST)
 (FNS ATOM-FULL-TEST STORAGE-FULL-TEST)
 (COMS (FNS DATATYPE-TEST)
 (RECORDS GC-TEST-TYPE)
 ;; DATATYPE TESTS
 )
 )
```

(COMS ;; CODE RECLAMATION TESTS

(FNS CODE-RECLAIM-TEST)

;; The function that is repeatedly compiled to test that code-block constants inside code blocks are reclaimed.

```
(VARS (CODE-RECLAIM-TEST-TEMP-FN
 ' (DEFINEQ (CODE-RECLAIM-TEST-TEMP-FN
 (ASDF)
 (LET (I)
 (FOR I FROM 1 TO 10 COLLECT (SQRT 4.5))
 (ERSETQ (DATE))
 (NLSETQ (DATE))
 (ERSETQ (DATE))
 (NLSETQ (DATE))
 (ERSETQ (DATE))
 (CL:FLET ((TEMP (ARG)
 (SETQ ARG (FLOAT ARG))
 (EXPT (SQRT I)
 (SQRT (COS (/ I 180)))))))
 (CL:UNWIND-PROTECT
 (FOR I FROM 1 TO 1000 COLLECT (TEMP I))
 (SETQ I NIL)))))))))
```

(FILESLOAD DANCEROBJ GCHAX)

(ADDTOVAR **DISPLAYFONTDIRECTORIES** "{ERIS}<TEST>GC>HAND-AUX>" "{ERIS}<LISPCORE>XEROXPRIVATE>FONTS>")

(ADDTOVAR **INTERPRESSFONTDIRECTORIES** "{ERIS}<TEST>GC>HAND-AUX>" "{ERIS}<LISPCORE>XEROXPRIVATE>FONTS>")

(SETQ DISPLAYFONTEXTENSIONS '(DISPLAYFONT AC STRIKE))

(DEFINEQ

### (MAIN-GC-TEST

```
(LAMBDA (LIMIT DRIBBLE-FILE STACK-COUNT BIGNUM-COUNT FIXP-COUNT FLOAT-COUNT TEDIT-COUNT LIST-COUNT CODE-COUNT
TYPE-COUNT LIST-LEN-LIMIT) ; Edited 23-Jun-88 13:30 by jds
```

(DRIBBLE (OR DRIBBLE-FILE "{LPT}"))

(PRINTOUT T ";;; \*\*\*\*\*" T ";;; MAIKO GARBAGE COLLECTOR TESTS" T ";;; Run on " (DATE)

T ";;; Dribble to " (OR DRIBBLE-FILE "{LPT}"))

T T)

(|for| I |from| 1 |to| (OR LIMIT 10) |do| (PRINTOUT T "Starting Maiko GC tests, pass " I T)

(ITEMS-ON-STACK-TEST (OR STACK-COUNT 100))

(MANY-BIGNUM-MAKER (OR BIGNUM-COUNT 1000))

(MANY-FIXP-MAKER (OR FIXP-COUNT 1000))

(MANY-FLOAT-MAKER (OR FLOAT-COUNT 1000))

(TEDIT-CRUNCH-TEST (OR TEDIT-COUNT 5))

(ARRAY-STRING-TEST 3)

(LIST-MANIPULATION-TEST (OR LIST-COUNT 5)

LIST-LEN-LIMIT)

(BOUNDARY-TESTS)

(CODE-RECLAIM-TEST (OR CODE-COUNT 20))

(VARIOUS-TYPES-TEST (OR TYPE-COUNT 10))

(FRPTQ 100 (RECLAIM))

(STORAGE))

(ATOM-FULL-TEST)

(STORAGE-FULL-TEST)

(DRIBBLE NIL))

)

(DEFINEQ

**(ITEMS-ON-STACK-TEST**

```
(LAMBDA (LIMIT) ; Edited 25-May-88 11:56 by jds
  (PRINTOUT T " Starting ITEMS-ON-STACK test for " LIMIT " iterations." T)
  (FOR I FROM 1 TO LIMIT DO (LET ((X (CREATE CHARLOOKS))
    (Y (EXPT 1234.5 (RAND 3 7))))
    (ERSETQ (FRPTQ 5 (RECLAIM))
      (COND
        ((\ISONFREELIST X)
          (HELP "X is free, but pointer is on stack.))
        ((\ISONFREELIST Y)
          (HELP "Y is free, but pointer is on stack.)))))
```

**(MANY-BIGNUM-MAKER**

```
(LAMBDA (LIMIT) ; Edited 25-May-88 11:54 by jds
  (PRINTOUT T " Starting MANY-BIGNUM-MAKER test for " LIMIT " iterations." T)
  (LET (X Y Z W)
    (FOR I FROM 1 TO LIMIT DO (SETQ X (CL:* 12345678901234567890 (RAND 1 I)))
      (SETQ Y (IQUOTIENT X 3))
      (SETQ Z (IPLUS Y X X 34 2 9 (IMOD X 7)
        (IREMAINDER Y 3)
        (CL:FLOOR Y 2)
        (CL:CEILING X 8)))
      (SETQ W (/ Z Y))))
```

**(MANY-FIXP-MAKER**

```
(LAMBDA (LIMIT) ; Edited 25-May-88 11:54 by jds
  (PRINTOUT T " Starting MANY-FIXP-MAKER test for " LIMIT " iterations." T)
  (LET (X Y Z W)
    (FOR I FROM 1 TO LIMIT DO (SETQ X (CL:* 543 (RAND 1 I)))
      (SETQ Y (IQUOTIENT X 3))
      (SETQ Z (IPLUS Y X X 34 2 9 (IMOD X 7)
        (IREMAINDER Y 3)
        (CL:FLOOR Y 2)
        (CL:CEILING X 8)))
      (SETQ W (/ Z Y))))
```

**(MANY-FLOAT-MAKER**

```
(LAMBDA (LIMIT) ; Edited 25-May-88 11:55 by jds
  (PRINTOUT T " Starting MANY-FLOAT-MAKER test for " LIMIT " iterations." T)
  (LET (X Y Z W)
    (FOR I FROM 1 TO LIMIT DO (SETQ X (FTIMES 1.0 (RAND 0 1)))
      (SETQ Y (+ (SQRT I)
        (EXPT (SQRT (SQRT I))
          3.4)))
      (SETQ Z (LOG Y))))
```

**(BOUNDARY-TESTS**

```
(LAMBDA NIL ; Edited 26-May-88 11:54 by jds
  ;; Tests the transition into and out of big refcnts, and BIG refcnt's.
  (PRINTOUT T " Starting Refcnt-63 crossing test" T)
  (LET* ((ITEM (|create| FMTSPEC))
    (LIST (|for| I |from| 1 |to| 62 |collect| ITEM)))
    (|for| I |from| 1 |to| 1000 |do| (|for| J |from| (LENGTH LIST) |to| (+ 63 (RAND 1 10))
      |do| (SETQ LIST (CONS ITEM LIST)))
      (|for| J |from| (LENGTH LIST) |to| (- 63 (RAND 3 12)) |do| (|pop| LIST))
    (COND
      ((ZEROP (IMOD I 31))
        (RECLAIM)))
    (PRINTOUT T " Starting Refcount-500K <-> NIL test." T)
    (|for| LOOP |from| 1 |to| 10 |do| (|for| I |from| 1 |to| 500000 |do| (SETQ LIST (CONS ITEM LIST))
      (SETQ LIST NIL))
    (PRINTOUT T " Starting Refcount 1-2 boundary test." T)
    (LET ((ITEM (LIST (|create| FMTSPEC))))
      (|for| I |from| 1 |to| 5000 |do| (SETQ ITEM2 (CAR ITEM))
        (SETQ ITEM2 NIL))
    (PRINTOUT T " Starting Refcount 1 + stack boundary test." T)
    (LET ((ITEM (|create| FMTSPEC))
      ITEM2)
      (|for| I |from| 1 |to| 5000 |do| (SETQ ITEM2 (LIST ITEM))
        (RPLACA ITEM2 NIL))
    (PRINTOUT T " Starting Refcount 0-1 boundary test." T)
    (LET (ITEM)
      (|for| I |from| 1 |to| 5000 |do| (SETQ ITEM (LIST (|create| FMTSPEC))
        (RPLACA ITEM NIL))))))
```

**(ARRAY-STRING-TEST**

```
(LAMBDA (LIMIT REAL-STRESS) ; Edited 23-Jun-88 12:23 by jds
  ;; Try out array & string creation, and substringing on the GC.
```

```
(PRINTOUT T " Starting Array & String test." T)
(FOR I FROM 1 TO (OR LIMIT 10)
 DO (LET (STRINGS ARRAYS)
      (FOR ARRAY-COUNT FROM 1 TO 5000 COLLECT (CL:MAKE-ARRAY (RAND 10
                                                                (COND
                                                                 (REAL-STRESS 65000)
                                                                 (T (IMAX 100 (IQUOTIENT 65000
                                                                ARRAY-COUNT
                                                                ))))))))
      (FOR I FROM 1 TO 5000 COLLECT (BITMAPCREATE (RAND 1 512)
                                                    (RAND 1 512)))
      (SETQ STRINGS (FOR STRING-COUNT FROM 1 TO 5000
                    COLLECT (ALLOCSTRING (RAND 10 (COND
                                                  (REAL-STRESS 65000)
                                                  (T (IMAX 100 (IQUOTIENT 65000 STRING-COUNT))))
                    )))
      (FOR STRING IN STRINGS COLLECT (SUBSTRING STRING (RAND 1 (LRSH (NCHARS STRING)
                                                                    1))
                                                    (RAND (ADD1 (LRSH (NCHARS STRING)
                                                                    1))
                                                    (NCHARS STRING)))))))))
```

**(VARIOUS-TYPES-TEST**

(LAMBDA (LIMIT) ; Edited 23-Jun-88 12:04 by jds

;; Run thru creation and collection of various types that have caused trouble in the past.

```
(PRINTOUT T " Starting various type cases." T)
(FOR REPEAT-COUNT FROM 1 TO (OR LIMIT 10) DO (|for| TYPE IN ' (VMEMPAGE) AS CREATION-LIMIT
                                             IN ' (100)
                                             |do| (FOR I FROM 1 TO CREATION-LIMIT
                                                    COLLECT (NCREATE TYPE))
                                                    (DORECLAIM))))))
```

)

(DEFINEQ

**(TEDIT-CRUNCH-TEST**

(LAMBDA (LIMIT) ; Edited 27-May-88 13:06 by jds

;; GC Testing -- stressing the world.

;; Hardcopy a big TEdit file to a {CORE} file, copy that to disk, and delete everything.

```
(PRINTOUT T " Starting TEDIT-CRUNCH test for " LIMIT " iterations." T)
(FOR PASS FROM 1 TO LIMIT DO (PRINTOUT T " Round " PASS " started " (DATE)
                                         " ." T)
                              (LET ((TS (OPENTEXTSTREAM ' {|ERIS}<Test>GC>Hand-Aux>ADVICT-N-Z.TEDIT|))
                                      TLIST)
                                  (TEDIT.HARDCOPY TS '{CORE}FOO.IP T)
                                  (COPYFILE '{CORE}FOO.IP '{DSK}FOO.IP)
                                  (DELFILE '{DSK}FOO.IP)
                                  (DELFILE '{CORE}FOO.IP)
                                  (CLOSEF (FETCH (TEXTOBJ TXTFILE) OF (TEXTOBJ TS))))))
```

**(LIST-MANIPULATION-TEST**

(LAMBDA (LIMIT LENGTH-LIMIT) ; Edited 23-Jun-88 14:03 by jds

;; Do lots of list creation, popping, and consing, to make sure the GC works.

```
(PRINTOUT T " Starting LIST-MANIPULATION test for " LIMIT " iterations." T)
(|for| PASS |from| 1 |to| LIMIT
 |do| (PRINTOUT T " Round " PASS " started " (DATE)
          " ." T)
      (LET ((TS (OPENTEXTSTREAM ' {|ERIS}<sybalsky>Top10-87>Dictionaries>ADVICT-A-M.TEDIT|))
              (LEN (RAND 0 (OR LENGTH-LIMIT 100000)))
              TLIST)
          (SETQ TLIST (|for| I |from| 1 |to| LEN |collect| TS))
          (|for| I |from| 1 |to| (RAND 1 (IMAX 1 (LRSH LEN 1))) |do| (|pop| TLIST))
          (|for| I |from| 1 |to| (RAND 1 100) |do| (SETQ TLIST (CONS TS TLIST)))
          (|for| I |from| 1 |to| (RAND 1 (IMAX 1 (LRSH (FLENGTH TLIST)
                                                    1)))
              |do| (|pop| TLIST))
          (|for| I |from| 1 |to| (RAND 1 2000) |do| (SETQ TLIST (CONS TS TLIST)))
          (|for| I |from| 1 |to| (RAND 1 (IMAX 1 (LRSH (FLENGTH TLIST)
                                                    1)))
              |do| (|pop| TLIST))
          (|for| \i |from| 1 |to| (RAND 1 1500)
              |do| (SETQ TLIST (NCONC TLIST (|for| J |from| 1 |to| (RAND 1 10)
                                                                    |join| (|for| K |from| 1 |to| 3 |collect| (CONS TS K))))))
          (|for| I |from| 1 |to| (RAND 1 (IMAX 1 (LRSH (FLENGTH TLIST)
                                                    1)))
              |do| (|pop| TLIST))
          (CLOSEF (|fetch| (TEXTOBJ TXTFILE) |of| (TEXTOBJ TS))))
      (LET ((GC-ITEM (NCREATE 'VMEMPAGE))
              (LEN (RAND 10 500))
```

```

      TLIST ELT)
      (SETQ TLIST (|for| I |from| 1 |to| LEN |collect| NIL))
      (|for| I |from| 1 |to| LEN |do| (SETQ ELT (CL:RANDOM LEN))
      (RPLACA (CL:NTHCDR ELT TLIST)
      GC-ITEM)
      (RPLACA (CL:NTHCDR (SUB1 I)
      TLIST)
      GC-ITEM))
      (|for| I |from| (SUB1 LEN) |to| 0 |by| -1 |do| (RPLACD (CL:NTHCDR I TLIST)
      GC-ITEM))))))

```

)

(DEFINEQ

**(ATOM-FULL-TEST**

; Edited 26-May-88 11:39 by jds

```

(LAMBDA NIL
  (PRINTOUT T " Starting ATOM-space full test.")
  (LET ((CUR-ATOM-COUNT |\\AtomFrLst|)
        (CL:UNWIND-PROTECT
         (PROGN (SETQ |\\AtomFrLst| 64000)
                 (FOR I FROM 64000 TO 70000 DO (GENSYM 'GC-TEST)))
                 (SETQ |\\AtomFrLst| CUR-ATOM-COUNT))))))

```

**(STORAGE-FULL-TEST**

; Edited 26-May-88 11:47 by jds

```

(LAMBDA NIL
  (PRINTOUT T " Starting Storage-full test." T)
  (ERSETQ (FOR I FROM 1 COLLECT (ARRAY 100))))))

```

)

(DEFINEQ

**(DATATYPE-TEST**

; Edited 26-May-88 11:26 by jds

```

(LAMBDA (LIMIT)
  (FOR I FROM 1 TO (OR LIMIT 10) DO (FOR L FROM 1 TO 100
  DO (FOR Y FROM 1 TO 20 COLLECT (CREATE GC-TEST-TYPE
  FIELD-1 _ T))
  (RECLAIM))))))

```

)

(DECLARE\ : EVAL@COMPILE

```

(DATATYPE GC-TEST-TYPE (FIELD-1 FIELD-2 FIELD-3 (FIELD-4 BYTE)
  (FIELD-5 FIXP)
  FIELD-6
  (FIELD-7 WORD)
  FIELD-8 FIELD-9 FIELD-10 FIELD-11 FIELD-12 FIELD-13 (FIELD-14 FIXP)
  FIELD-15
  (FIELD-16 XPOINTER)
  FIELD-17
  (FIELD-18 BYTE)
  (FIELD-19 FIXP)
  FIELD-20
  (FIELD-21 BYTE)
  FIELD-22 FIELD-23 FIELD-24 (FIELD-25 BYTE)
  FIELD-26
  (FIELD-27 BYTE)
  FIELD-28
  (FIELD-29 BYTE)
  FIELD-30
  (FIELD-31 WORD)
  FIELD-32
  (FIELD-33 XPOINTER)
  FIELD-34
  (FIELD-35 FIXP)
  FIELD-36 FIELD-37 FIELD-38 (FIELD-39 FLAG)
  FIELD-40
  (FIELD-41 FLAG)
  FIELD-42
  (FIELD-43 FIXP)
  (FIELD-44 FIXP)
  FIELD-45
  (FIELD-46 XPOINTER)
  FIELD-47 FIELD-48 FIELD-49 (FIELD-50 FLAG)
  (FIELD-51 BYTE)
  FIELD-52 FIELD-53 (FIELD-54 BYTE)
  FIELD-55 FIELD-56 (FIELD-57 BYTE)
  (FIELD-58 WORD)
  FIELD-59 FIELD-60 (FIELD-61 XPOINTER)
  FIELD-62 FIELD-63 (FIELD-64 XPOINTER)
  (FIELD-65 XPOINTER)
  FIELD-66 FIELD-67 FIELD-68 FIELD-69 (FIELD-70 FLAG)
  FIELD-71 FIELD-72 (FIELD-73 WORD)
  FIELD-74

```

(FIELD-75 FLAG)  
FIELD-76 FIELD-77 FIELD-78 FIELD-79 (FIELD-80 FIXP)  
(FIELD-81 FIXP)  
FIELD-82 FIELD-83 FIELD-84 FIELD-85 (FIELD-86 XPOINTER)  
(FIELD-87 BYTE)  
(FIELD-88 XPOINTER)  
FIELD-89  
(FIELD-90 BYTE)  
(FIELD-91 FLAG)  
(FIELD-92 FIXP)  
(FIELD-93 FIXP)  
(FIELD-94 FLAG)  
FIELD-95  
(FIELD-96 FLAG)  
FIELD-97  
(FIELD-98 FLAG)  
FIELD-99 FIELD-100 FIELD-101 FIELD-102 FIELD-103 (FIELD-104 XPOINTER)  
FIELD-105 FIELD-106 FIELD-107 FIELD-108 (FIELD-109 BYTE)  
FIELD-110  
(FIELD-111 WORD)  
FIELD-112  
(FIELD-113 XPOINTER)  
(FIELD-114 FLAG)  
(FIELD-115 FIXP)  
FIELD-116 FIELD-117 (FIELD-118 BYTE)  
FIELD-119 FIELD-120 FIELD-121 FIELD-122 FIELD-123 (FIELD-124 XPOINTER)  
(FIELD-125 BYTE)  
(FIELD-126 XPOINTER)  
FIELD-127 FIELD-128 (FIELD-129 FIXP)  
(FIELD-130 FLAG)  
FIELD-131 FIELD-132 FIELD-133 FIELD-134 (FIELD-135 WORD)  
(FIELD-136 FLAG)  
FIELD-137 FIELD-138 FIELD-139 (FIELD-140 WORD)  
(FIELD-141 FLAG)  
FIELD-142 FIELD-143 FIELD-144 (FIELD-145 FIXP)  
FIELD-146 FIELD-147 FIELD-148 FIELD-149 (FIELD-150 FLAG)  
FIELD-151 FIELD-152 FIELD-153 FIELD-154 (FIELD-155 FIXP)  
FIELD-156  
(FIELD-157 BYTE)  
FIELD-158  
(FIELD-159 FIXP)  
(FIELD-160 WORD)  
FIELD-161  
(FIELD-162 WORD)  
(FIELD-163 FIXP)  
FIELD-164  
(FIELD-165 FIXP)  
FIELD-166  
(FIELD-167 FLAG)  
(FIELD-168 BYTE)  
FIELD-169 FIELD-170 (FIELD-171 XPOINTER)  
(FIELD-172 BYTE)  
FIELD-173 FIELD-174 (FIELD-175 FLAG)  
(FIELD-176 BYTE)  
(FIELD-177 WORD)  
FIELD-178  
(FIELD-179 FIXP)  
FIELD-180 FIELD-181 (FIELD-182 BYTE)  
FIELD-183 FIELD-184 FIELD-185 FIELD-186 FIELD-187 (FIELD-188 BYTE)  
(FIELD-189 FIXP)  
FIELD-190 FIELD-191 FIELD-192 (FIELD-193 BYTE)  
FIELD-194  
(FIELD-195 WORD)  
FIELD-196 FIELD-197 FIELD-198 FIELD-199 (FIELD-200 WORD)  
FIELD-201  
(FIELD-202 FLAG)  
FIELD-203  
(FIELD-204 XPOINTER)  
FIELD-205 FIELD-206 FIELD-207 (FIELD-208 FLAG)  
FIELD-209  
(FIELD-210 WORD)  
(FIELD-211 BYTE)  
FIELD-212 FIELD-213 FIELD-214 (FIELD-215 FIXP)  
FIELD-216 FIELD-217 (FIELD-218 XPOINTER)  
FIELD-219  
(FIELD-220 FLAG)  
FIELD-221  
(FIELD-222 FLAG)  
(FIELD-223 WORD)  
(FIELD-224 FLAG)  
(FIELD-225 WORD)  
FIELD-226 FIELD-227 FIELD-228 FIELD-229 FIELD-230 (FIELD-231 XPOINTER)  
FIELD-232  
(FIELD-233 WORD)  
(FIELD-234 WORD)  
FIELD-235 FIELD-236 FIELD-237 FIELD-238 FIELD-239 FIELD-240 FIELD-241 (FIELD-242  
XPOINTER)



```
(PRINTOUT T " Starting code-block reclaim test" T)
(|for| I |from| 1 |to| LIMIT |do| (BKSYSBUF "ST
N
")
(COMPILE 'CODE-RECLAIM-TEST-TEMP-FN))
(PRINTOUT T " Starting MAPATOMS(GETD)" T)
(|for| I |from| 1 |to| LIMIT |do| (MAPATOMS (FUNCTION GETD)))
(PRINTOUT T " Starting MAPATOMS(MOVD to DUMMYFN)" T)
(FOR I FROM 1 TO LIMIT DO (MAPATOMS #'(LAMBDA (FN-NAME)
(AND (GETD FN-NAME)
(MOVD FN-NAME 'MAIKO-GC-TEST-DUMMY-FN))))))
```

)  
 ;; The function that is repeatedly compiled to test that code-block constants inside code blocks are reclaimed.

```
(RPAQQ CODE-RECLAIM-TEST-TEMP-FN
(DEFINEQ (CODE-RECLAIM-TEST-TEMP-FN (ASDF)
(LET (I)
(FOR I FROM 1 TO 10 COLLECT (SQRT 4.5))
(ERSETQ (DATE))
(NLSETQ (DATE))
(ERSETQ (DATE))
(NLSETQ (DATE))
(ERSETQ (DATE))
(CL:FLET ((TEMP (ARG)
(SETQ ARG (FLOAT ARG))
(EXPT (SQRT I)
(SQRT (COS (/ I 180))))))
(CL:UNWIND-PROTECT
(FOR I FROM 1 TO 1000 COLLECT (TEMP I))
(SETQ I NIL))))))
(PUTPROPS MAIKO-GC-TESTS COPYRIGHT ("John Sybalsky & Xerox Corporation" 1988))
```

---

**FUNCTION INDEX**

ARRAY-STRING-TEST .....2    DATATYPE-TEST .....4    MANY-BIGNUM-MAKER .....2    TEDIT-CRUNCH-TEST .....3  
ATOM-FULL-TEST .....4    ITEMS-ON-STACK-TEST .....2    MANY-FIXP-MAKER .....2    VARIOUS-TYPES-TEST .....3  
BOUNDARY-TESTS .....2    LIST-MANIPULATION-TEST ..3    MANY-FLOAT-MAKER .....2  
CODE-RECLAIM-TEST .....6    MAIN-GC-TEST .....1    STORAGE-FULL-TEST .....4

---

**VARIABLE INDEX**

CODE-RECLAIM-TEST-TEMP-FN .....7    DISPLAYFONTDIRECTORIES .....1    INTERPRESSFONTDIRECTORIES .....1

---

**RECORD INDEX**

GC-TEST-TYPE .....4

---