File created: 20-Oct-88 17:19:09 **{POOH/N}<POOH>VANMELLE>LISP>XCLOPCODETESTS;2**

changes to: (VARS XCLOPCODETESTSCOMS)

previous date: 26-Sep-88 14:11:23 {POOH/N}<POOH>VANMELLE>LISP>XCLOPCODETESTS;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ XCLOPCODETESTSCOMS
       [(COMS                                                    ; FINDKEY
              (FNS FINDKEYTESTER DOFINDKEYTEST DOFINDKEYTEST1)
              (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS FINDKEYCHECK)))
        (COMS                                                    ; RESTLIST
              (FNS \RESTLIST.SPLICE.FRAME RESTLISTTESTER DORESTLISTTEST GETRESTARGREFCNTS DORESTLISTTEST1)
              (INITVARS (RESTLISTCOUNTER 0))
              (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS RESTLISTCHECK \COMPUTED.FORM)
                     (RECORDS MDSTYPEWORD)
                     (GLOBALVARS RESTLISTCOUNTER)))
        (COMS                                                    ; UNWIND
              (FNS UNWINDTESTER UNWINDMAINTEST UNWINDMAINTEST.RECURSE UNWINDCHECK1 UNWINDCHECK2 UNWINDCODE)
              (DECLARE%: EVAL@COMPILE DONTCOPY (RECORDS BINDMARKSLOT))
              (FNS UW2.TEST UW2.RECURSE UW2.TEST.MAIN UW2.CHECK UW2.IDENTITY))
        (COMS                                                    ; Closure tests
              (FNS CLOSURETESTER CLOSUREMAINTEST CLOSUREMAINTEST.RECURSE CLOSUREFNCHECK CLOSUREFNCHECK2
                   CLOSUREFN1 CLOSUREFN1VALUE CLOSUREFN2 CLOSUREFN2VALUE CLOSUREFN4CODE CLOSUREFN4VALUE)
              (INITVARS (CLOSURETEST.DEPTH 50)
                     (CLOSURETEST.ENVIRONMENT "Closure Environment"))
              (GLOBALVARS CLOSURETEST.DEPTH CLOSURETEST.ENVIRONMENT))
        (COMS (FNS CHECKSTACKSPACE))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA DORESTLISTTEST DOFINDKEYTEST]
)

;; FINDKEY

(DEFINEQ

(FINDKEYTESTER
  [LAMBDA NIL                                                    (* bvm%: "14-Jul-86 17:54")

;;; Test the opcode FINDKEY

    (DOFINDKEYTEST 'KEYA 'VALA 'KEYB 'VALB 'KEYC 'VALC])

(DOFINDKEYTEST
  [LAMBDA KEYARGS                                                (* bvm%: "21-Jul-86 16:37")
    (DECLARE (SPECVARS KEYARGS))
    (AND (FINDKEYCHECK 1 KEYA)
         (FINDKEYCHECK 2 KEYA)
         (FINDKEYCHECK 3 KEYA)
         (FINDKEYCHECK 4 KEYA)
         (FINDKEYCHECK 5 KEYA)
         (FINDKEYCHECK 6 KEYA)
         (FINDKEYCHECK 7 KEYA)
         (FINDKEYCHECK 8 KEYA)
         (FINDKEYCHECK 1 KEYB)
         (FINDKEYCHECK 2 KEYB)
         (FINDKEYCHECK 3 KEYB)
         (FINDKEYCHECK 4 KEYB)
         (FINDKEYCHECK 5 KEYB)
         (FINDKEYCHECK 6 KEYB)
         (FINDKEYCHECK 7 KEYB)
         (FINDKEYCHECK 8 KEYB)
         (FINDKEYCHECK 1 KEYC)
         (FINDKEYCHECK 2 KEYC)
         (FINDKEYCHECK 3 KEYC)
         (FINDKEYCHECK 4 KEYC)
         (FINDKEYCHECK 5 KEYC)
         (FINDKEYCHECK 6 KEYC)
         (FINDKEYCHECK 7 KEYC)
         (FINDKEYCHECK 8 KEYC])

(DOFINDKEYTEST1
  [LAMBDA (RESULT N KEY)                                         (* bvm%: "21-Jul-86 16:37")
    (DECLARE (USEDFREE KEYARGS))
    (LET [(ANSWER (for I from N by 2 to KEYARGS when (EQ KEY (ARG KEYARGS I)) do (RETURN (ADD1 I]
```

```
         (COND
            ((NEQ ANSWER RESULT)
             (HELP (CONCAT "FINDKEY." N " returned " RESULT " instead of " ANSWER " for ")
                KEY))
            (T T])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS FINDKEYCHECK DMACRO (DEFMACRO (N KEY) `(DOFINDKEYTEST1 ((OPCODES FINDKEY ,N)
                                                                  ',KEY)
                                                                 ,N
                                                                 ',KEY)))
)
)
```

;; RESTLIST

```
(DEFINEQ
```

(\RESTLIST.SPLICE.FRAME

```
  [LAMBDA NIL                                                    (* bvm%: "21-Jul-86 17:13")
```

;;; If caller is fast, so its BF is contiguous with its caller's FX, then adjust pointers so that its first ivar goes back on it's caller's fx, and back up pc

```
    (UNINTERRUPTABLY
        (LET ((CALLER (\MYALINK))
              CALLER2 IVAR BF)
             (COND
                ([AND (fetch (FX FASTP) of CALLER)
                      (EQ [SETQ IVAR (fetch (BF IVAR) of (SETQ BF (fetch (FX DUMMYBF) of CALLER]
                          (fetch (FX NEXTBLOCK) of (SETQ CALLER2 (fetch (FX ALINK) of CALLER]
                 (replace (BF IVAR) of BF with (add IVAR WORDSPERCELL))
                 (replace (FX NEXTBLOCK) of CALLER2 with IVAR)
                 (add (fetch (FX PC) of CALLER2)
                      -2)
                 T)))])
```

(RESTLISTTESTER

```
  [LAMBDA NIL                                                    (* bvm%: "21-Jul-86 17:28")
```

;;; Test the opcode RESTLIST

```
    (AND (DORESTLISTTEST 'KEYA 'VALA 'KEYB 'VALB 'KEYC 'VALC)
         (DORESTLISTTEST '(KEYA)
                '(VALA)
                '(KEYB)
                'VALB
                '(KEYC)
                '(VALC))
         (DORESTLISTTEST)
         (\COMPUTED.FORM (CONS 'DORESTLISTTEST (for I from 1 to 200 collect `',(LIST I])
```

(DORESTLISTTEST

```
  [LAMBDA KEYARGS                                                (* bvm%: "21-Jul-86 16:39")
    (DECLARE (SPECVARS KEYARGS))
    (AND (RESTLISTCHECK 1)
         (RESTLISTCHECK 2)
         (RESTLISTCHECK 3)
         (RESTLISTCHECK 4)
         (RESTLISTCHECK 5)
         (RESTLISTCHECK 6)
         (RESTLISTCHECK 7)
         (RESTLISTCHECK 8])
```

(GETRESTARGREFCNTS

```
  [LAMBDA (N)
    (DECLARE (USEDFREE KEYARGS))                                 (* bvm%: "18-Jul-86 15:01")
    (for I from N to KEYARGS collect (\REFCNT (ARG KEYARGS I])
```

(DORESTLISTTEST1

```
  [LAMBDA (REFCNTS RESULT N)                                     (* bvm%: "21-Jul-86 17:22")
    (DECLARE (USEDFREE KEYARGS))
    [COND
       ([OR (NOT (EQLENGTH RESULT (IMAX (IDIFFERENCE KEYARGS (SUB1 N))
                                        0)))
            (for R in RESULT as I from N to KEYARGS thereis (NEQ R (ARG KEYARGS I]
        (HELP (CONCAT "RESTLIST." N " returned " RESULT " instead of " (for I from N to KEYARGS
                                                                          collect (ARG KEYARGS I]
    [for TAIL on RESULT as CNT in REFCNTS as I from 1
```

```
       do (COND
             [[AND (NEQ (\REFCNT (CAR TAIL))
                        (ADD1 CNT))
                   (NOT (fetch (MDSTYPEWORD NOREFCNT) of (\ADDBASE \MDSTypeTable (LRSH (fetch (POINTER PAGE#)
                                                                                          of (CAR TAIL))
                                                                                      1]
                    (HELP (CONCAT "Ref cnt of " I "th RESTLIST element was not incremented")
                          (CONCAT "Was " CNT ", now is " (\REFCNT (CAR TAIL]
                   ((NEQ (\REFCNT TAIL)
                         (COND
                            ((EQ TAIL RESULT)
                             0)
                            (T 1)))
                    (HELP (COND
                             ((EQ TAIL RESULT)
                              "Ref cnt of RESTLIST value is not zero")
                             (T "Ref cnt of RESTLIST tail is not one"))
                          (\REFCNT TAIL]
    T])

)

(RPAQ? RESTLISTCOUNTER 0)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS RESTLISTCHECK DMACRO (DEFMACRO (N) `(PROGN (RECLAIM)
                                                     (DORESTLISTTEST1 (GETRESTARGREFCNTS ,N)
                                                            ((OPCODES RESTLIST ,N)
                                                             NIL KEYARGS)
                                                            ,N))))

(PUTPROPS \COMPUTED.FORM MACRO [X (CONS 'PROGN (MAPCAR X (FUNCTION EVAL])
)

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD MDSTYPEWORD ((NOREFCNT FLAG)
                          (NIL BITS 15)))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS RESTLISTCOUNTER)
)
)

;; UNWIND

(DEFINEQ

(UNWINDTESTER
  [LAMBDA (DEPTH)                                                    ; Edited 26-Sep-88 14:11 by bvm
    [for D from 0 to (OR DEPTH CLOSURETEST.DEPTH) do (LET [(VALUE (UNWINDMAINTEST D (LOGAND D 7]
                                                          (COND
                                                             ((NEQ VALUE 'SUCCESS)
                                                              (HELP "UNWINDMAINTEST did not return correctly"
                                                                    VALUE]
    T])

(UNWINDMAINTEST
  [LAMBDA (DEPTH CODE)                                               ; Edited 26-Sep-88 14:10 by bvm
    (COND
       [(OR (NULL DEPTH)
            (LEQ DEPTH 0))
        (LET ((*B* 3)
              (*C* 2)
              (*D* DEPTH))
             (DECLARE (CL:SPECIAL *B* *C* *D*))
             (LIST (UNWINDCHECK1 DEPTH)
                   (LET ((*E* 10)
                         (*F* 11)
                         (*G* 12)
                         (*H* DEPTH))
                        (DECLARE (CL:SPECIAL *E* *F* *G* *H*))
                        ;; There are 8 pvar slots in this frame (for 7 pvars), so empty stack = 8+2 = 10.  Right now the stack depth is up to 13,
                        ;; because of two bind marks and the value returned from UNWINDCHECK1.
                        (UNWINDCHECKFAIL T *B* *C* *D* *E* *F* 'PREVPREV 'PREVIOUS-VALUE
                               [PROGN                                ; Unwind stack to depth 10 + {0,1,3,6} preserving tos when code
                                                                     ; is even
                                      (SELECTQ CODE
                                          (0                         ; Blow away whole stack
                                             ((OPCODES UNWIND 10 0)))
```

```
                                        (1                          ; Same as 0 but keep tos
                                           ((OPCODES UNWIND 10 1)))
                                        (2                          ; Blow away second binding only
                                           ((OPCODES UNWIND 11 0)))
                                        (3                          ; Same as 2 but keep tos
                                           ((OPCODES UNWIND 11 1)))
                                        (4                          ; Don't touch the bindings, just get rid of some dynamic stuff
                                           ((OPCODES UNWIND 13 0)))
                                        (5                          ; Same as 4 but keep tos
                                           ((OPCODES UNWIND 13 1)))
                                        (6                          ; Don't touch the bindings, just get rid of some dynamic stuff
                                           ((OPCODES UNWIND 16 0)))
                                        ((OPCODES UNWIND 16 1]
                                  (PROGN                            ; Check that previous opcode left the stack in the right state
                                         (UNWINDCHECK2 CODE]
      (T                                                            ; Separate call so the compiler doesn't optimize out the recursion
         (UNWINDMAINTEST.RECURSE (SUB1 DEPTH)
                CODE])
```

(UNWINDMAINTEST.RECURSE

```
  [LAMBDA (DEPTH CODE)                                              ; Edited 26-Sep-88 14:08 by bvm
    (UNWINDMAINTEST DEPTH CODE])
```

(UNWINDCHECK1

```
  [LAMBDA NIL                                                       (* bvm%: "21-Jul-86 13:15")
                                                                    ; This just prevents compiler from merging specials

    NIL])
```

(UNWINDCHECK2

```
  [LAMBDA (CODE)                                                    ; Edited 26-Sep-88 14:10 by bvm
```

;;; Check that the UNWIND opcode executed prior to this did the right thing. TOS should be PREVIOUS-VALUE if the UNWIND said to preserve TOS.

```
    (LET* [(CALLER (\MYALINK))
           (EOS (fetch (FX NEXTBLOCK) of CALLER))
           (GOODEOS (+ (fetch (FX FIRSTPVAR) of CALLER)
                       (UNFOLD (+ 10 (LOGAND CODE 1)
                                  (SELECTQ (LRSH CODE 1)
                                           (0 0)
                                           (1 1)
                                           (2 3)
                                           6))
                                  WORDSPERCELL]
          (COND
             ((NEQ EOS GOODEOS)
              (HELP (CONCAT (UNWINDCODE CODE)
                            " unwound stack "
                            (COND
                               ((GREATERP GOODEOS EOS)
                                "too far")
                               (T "not far enough"))
                            " by "
                            (ABS (DIFFERENCE EOS GOODEOS))
                            " words")))
             ((AND (ODDP CODE)
                   (NEQ (\GETBASEPTR (ADDSTACKBASE (IDIFFERENCE EOS WORDSPERCELL))
                                     0)
                        'PREVIOUS-VALUE))                           ; Should have preserved tos
              (HELP (UNWINDCODE CODE)
                    " did not preserve top of stack")))
          [for V in '(*B* *C* *D* *E* *F* *G* *H*) bind SHOULDBEUNBOUNDP
             do (SETQ SHOULDBEUNBOUNDP (SELECTQ (LRSH CODE 1)
                                                (0 T)
                                                (1 (FMEMB V '(*E* *F* *G* *H*)))
                                                NIL))
                (COND
                   [(\FRAMESCAN CALLER (\ATOMVALINDEX V))
                    (COND
                       (SHOULDBEUNBOUNDP (HELP (CONCAT (UNWINDCODE CODE)
                                                        " left variable " V " bound but shouldn't have"]
                   ((NOT SHOULDBEUNBOUNDP)
                    (HELP (CONCAT (UNWINDCODE CODE)
                                  " left variable " V " unbound but shouldn't have"]
          (PROGN ;; Escape from UNWINDMAINTEST because the UNWIND there has ruined its stack
                 (RETFROM 'UNWINDMAINTEST 'SUCCESS])
```

(UNWINDCODE

```
  [LAMBDA (CODE)                                                    (* bvm%: "21-Jul-86 15:34")
    (CONCAT "UNWIND." (PLUS 10 (LOGAND CODE 1))
            "."
            (LRSH CODE 1])
```

```
)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

[BLOCKRECORD BINDMARKSLOT ((BINDMARKP FLAG)
                           (NIL BITS 15))
       (BLOCKRECORD BINDMARKSLOT ((BINDNEGVALUES WORD)
                                  (BINDLASTPVAR WORD)))
       (ACCESSFNS BINDMARKSLOT ((BINDNVALUES (PROGN                (* Value stored in high half is one's complement of number of
                                                                     values bound)
                                                   (LOGXOR (fetch BINDNEGVALUES of DATUM)
                                                          65535]
)
)

(DEFINEQ
```

(UW2.TEST

```
  [LAMBDA (DEPTH)                                                   ; Edited 20-Oct-88 15:00 by vanmelle
    (COND
       ((OR (NULL DEPTH)
            (LEQ DEPTH 0))
        (UW2.TEST.MAIN))
       (T                                                           ; Separate call so the compiler doesn't optimize out the recursion
          (UW2.RECURSE (SUB1 DEPTH])
```

(UW2.RECURSE

```
  [LAMBDA (DEPTH)                                                   ; Edited 20-Oct-88 14:56 by vanmelle
                                                                    ; To foil compiler
    (UW2.TEST DEPTH])
```

(UW2.TEST.MAIN

```
  [LAMBDA (DEPTH)                                                   ; Edited 20-Oct-88 15:49 by bvm
    (LET ((*B* 3)
          (*C* 2.4)
          (*D* DEPTH))
         (DECLARE (CL:SPECIAL *B* *C* *D*))
         (LIST (UW2.IDENTITY 'TOS)
               (LET ((*E* 3.5))
                    (DECLARE (CL:SPECIAL *E*))
                    ;; There are 4 pvar slots in this frame, so empty stack = 4+2 = 6.  Right now the stack depth is up to 9, because of 2 bind
                    ;; marks and the value from NILL.
                    (UNWINDCHECKFAIL T NIL ((OPCODES UNWIND 9 0))
                           (UW2.CHECK])
```

(UW2.CHECK

```
  [LAMBDA NIL                                                       ; Edited 20-Oct-88 15:49 by bvm
```

;;; Check that the UNWIND opcode executed prior to this did the right thing.

```
    (LET* [(CALLER (\MYALINK))
           (EOS (fetch (FX NEXTBLOCK) of CALLER))
           (GOODEOS (+ (fetch (FX FIRSTPVAR) of CALLER)
                       (UNFOLD 9 WORDSPERCELL]
          [COND
             ((NEQ EOS GOODEOS)
              (HELP (CONCAT "Unwound stack " (COND
                                                ((GREATERP GOODEOS EOS)
                                                 "too far")
                                                (T "not far enough"))
                            " by "
                            (ABS (DIFFERENCE EOS GOODEOS))
                            " words"]
          [for V in '(*B* *C* *D* *E*) bind SHOULDBEUNBOUNDP
             do [SETQ SHOULDBEUNBOUNDP (AND NIL (EQ V '*E*]
                (COND
                   [(\FRAMESCAN CALLER (\ATOMVALINDEX V))
                    (COND
                       (SHOULDBEUNBOUNDP (HELP (CONCAT "UNWIND left variable " V " bound but shouldn't have"]
                   ((NOT SHOULDBEUNBOUNDP)
                    (HELP (CONCAT "UNWIND left variable " V " unbound but shouldn't have"]
          (PROGN ;; Escape from test because the UNWIND there has confused its stack
                 (RETFROM 'UW2.TEST.MAIN 'SUCCESS])
```

(UW2.IDENTITY

```
  [LAMBDA (X)                                                       ; Edited 20-Oct-88 15:19 by bvm
                                                                    ; Identity compiler doesn't know about
    X])
```

```
)

;; Closure tests

(DEFINEQ

(CLOSURETESTER
  [LAMBDA (DEPTH)                                                    (* bvm%: "21-Jul-86 16:40")
    (for D from 0 to (OR DEPTH CLOSURETEST.DEPTH) always (CLOSUREMAINTEST D])

(CLOSUREMAINTEST
  [LAMBDA (DEPTH)                                                    (* bvm%: "21-Jul-86 16:40")
    (COND
       [(OR (NULL DEPTH)
            (LEQ DEPTH 0))
        (LET (VALUE)
             (PUTD 'CLOSUREFN4 (MAKE-COMPILED-CLOSURE (fetch (LITATOM DEFPOINTER) of 'CLOSUREFN4CODE)
                                    CLOSURETEST.ENVIRONMENT))
             (COND
                ([NOT (EQUAL (SETQ VALUE (FUNCALL (GETD 'CLOSUREFN1)
                                                  'A
                                                  'B
                                                  'C))
                             (CLOSUREFN1VALUE 'A 'B 'C]
                 (HELP "CLOSUREFN1 returned the wrong value" VALUE))
                ([NOT (EQUAL (SETQ VALUE (FUNCALL (MAKE-COMPILED-CLOSURE (fetch (LITATOM DEFPOINTER)
                                                                                 of 'CLOSUREFN2)
                                                                          CLOSURETEST.ENVIRONMENT)
                                                  'A
                                                  'B
                                                  'C))
                             (CLOSUREFN2VALUE 'A 'B 'C]
                 (HELP "CLOSUREFN2 returned the wrong value" VALUE))
                ((NOT (EQUAL (SETQ VALUE (CLOSUREFN4))
                             (CLOSUREFN4VALUE)))
                 (HELP "CLOSUREFN4 returned the wrong value" VALUE))
                (T T]
       (T                                                            ; Separate call so the compiler doesn't optimize out the recursion
          (CLOSUREMAINTEST.RECURSE (SUB1 DEPTH])

(CLOSUREMAINTEST.RECURSE
  [LAMBDA (DEPTH)                                                    (* bvm%: "18-Jul-86 14:07")
    (CLOSUREMAINTEST DEPTH])

(CLOSUREFNCHECK
  [LAMBDA (CLOSUREP FUNCALLP)                                        (* bvm%: "18-Jul-86 14:48")
    (LET* [(CALLER (\MYALINK))
           (PVAR0 (STACKADDBASE (fetch (FX FIRSTPVAR) of CALLER]
          (COND
             [CLOSUREP (COND
                          ((NEQ (\GETBASEPTR PVAR0 0)
                                CLOSURETEST.ENVIRONMENT)
                           (HELP (COND
                                    (FUNCALLP "FUNCALL of a full closure")
                                    (T "Call to symbol with Closure definition"))
                                 " did not store closure environment in pvar0"]
             ((fetch (PVARSLOT BOUND) of PVAR0)
              (HELP "FUNCALL of a null closure stored something into pvar0"])

(CLOSUREFNCHECK2
  [LAMBDA NIL                                                        (* bvm%: "18-Jul-86 14:51")
                                                                     ; Nothing really to check for now
    NIL])

(CLOSUREFN1
  [LAMBDA (ARG1 ARG2 ARG3 ARG4)                                      (* bvm%: "18-Jul-86 15:30")
                                                                     ; Vanilla closure called via FUNCALL
    (CLOSUREFNCHECK NIL)
    (LET ((DUMMY1 T)
          (DUMMY2 NIL))
         (DECLARE (SPECVARS DUMMY1 DUMMY2))
         (CLOSUREFNCHECK2)
         (CLOSUREFN1VALUE ARG1 ARG2 ARG3 ARG4])

(CLOSUREFN1VALUE
  [LAMBDA (ARG1 ARG2 ARG3 ARG4)                                      (* bvm%: "18-Jul-86 15:30")
    (LIST ARG1 ARG2 ARG3 ARG4])

(CLOSUREFN2
  [LAMBDA (ARG1 ARG2 ARG3 ARG4)                                      (* bvm%: "18-Jul-86 15:37")
```

```
                                                                    ; Vanilla closure called via FUNCALL
    (CLOSUREFNCHECK T T)
    (LET ((DUMMY1 T)
          (DUMMY2 NIL))
         (DECLARE (SPECVARS DUMMY1 DUMMY2))
         (CLOSUREFNCHECK2)
         (CLOSUREFN2VALUE ARG1 ARG2 ARG3 ARG4])
```

(CLOSUREFN2VALUE

```
  [LAMBDA (ARG1 ARG2 ARG3 ARG4)                                     (* bvm%: "18-Jul-86 15:37")
    (LIST ARG4 ARG3 ARG2 ARG1])
```

(CLOSUREFN4CODE

```
  [LAMBDA (ARG1 ARG2 ARG3)                                          (* bvm%: "18-Jul-86 15:53")
                                                                    ; closure called via FNx
    (CLOSUREFNCHECK T NIL)
    (LET ((DUMMY1 T)
          (DUMMY2 NIL))
         (DECLARE (SPECVARS DUMMY1 DUMMY2))
         (CLOSUREFNCHECK2)
         (CLOSUREFN4VALUE ARG1 ARG2 ARG3])
```

(CLOSUREFN4VALUE

```
  [LAMBDA (ARG1 ARG2 ARG3)                                          (* bvm%: "18-Jul-86 15:38")
    (LIST ARG2 ARG3 ARG1])

)

(RPAQ? CLOSURETEST.DEPTH 50)

(RPAQ? CLOSURETEST.ENVIRONMENT "Closure Environment")

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS CLOSURETEST.DEPTH CLOSURETEST.ENVIRONMENT)
)

(DEFINEQ
```

(CHECKSTACKSPACE

```
  [LAMBDA (N START)                                                 (* bvm%: "18-Jul-86 16:23")
    (PROG ((SCANPTR (fetch StackBase of \InterfacePage))
           (EASP (fetch EndOfStack of \InterfacePage)))
      SCAN
          (SELECTC (fetch (STK FLAGS) of SCANPTR)
              (\STK.FSB (add SCANPTR (fetch (FSB SIZE) of SCANPTR)))
              (\STK.GUARD (COND
                             ((EQ SCANPTR EASP)                     ; Guard block not at end of stack, treat as a free block
                              (RETURN T)))
                          (add SCANPTR (fetch (FSB SIZE) of SCANPTR))
                                                                    ; reached end

                          )
              (\STK.FX                                              ; frame extension
                       (CHECK (fetch (FX CHECKED) of SCANPTR))
                       (SETQ SCANPTR (fetch (FX NEXTBLOCK) of SCANPTR)))
              (LET ((ORIG SCANPTR))                                 ; must be a basic frame
                   (until (type? BF SCANPTR) do (CHECK (EQ (fetch (STK FLAGS) of SCANPTR)
                                                           \STK.NOTFLAG))
                                                (add SCANPTR WORDSPERCELL))
                   [CHECK (COND
                             ((fetch (BF RESIDUAL) of SCANPTR)
                              (EQ SCANPTR ORIG))
                             (T (AND (fetch (BF CHECKED) of SCANPTR)
                                     (EQ ORIG (fetch (BF IVAR) of SCANPTR]
                   (add SCANPTR WORDSPERCELL)))
      NEXT
          (CHECK (ILEQ SCANPTR EASP))
          (GO SCAN])

)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA DORESTLISTTEST DOFINDKEYTEST)
)

(PUTPROPS XCLOPCODETESTS COPYRIGHT ("Xerox Corporation" 1988))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

RECORD INDEX