USERINIT.LSP IS A SYSTEM FILE WHICH LOADS

1.TIME2.F4S

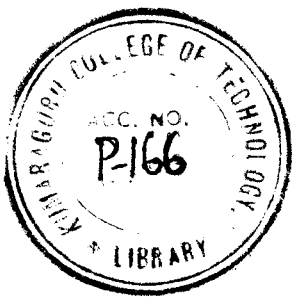
2.GCLTIME.F4S

3.BIGHOOKS.F4S

4.BIGARITH.F4S

5.BIGLOG.F4S

6.REPLACE.F4S



```
;;; Change Log:
;;;
;;; 12/06/89  Phil    File created.

;;; End.

(LOAD"D:/GCL40/LISPLIB/CH25/TIME2.F4S")
(LOAD"D:/GCL40/LISPLIB/CH25/GCLTIME.F4S")
(LOAD"D:/GCL40/LISPLIB/CH12/BIGHOOKS.F4S")
(LOAD"D:/GCL40/LISPLIB/CH12/BIGARITH.F4S")
(LOAD"D:/GCL40/LISPLIB/CH12/BIGLOG.F4S")
(LOAD"D:/GCL40/LISPLIB/CH14/REPLACE.F4S")
```

LOAD.LSP:-LOADS THE FILES REQUIRED FOR DIAGNOSIS

```
(defun load-files()
     (load"d:/pro/start.lsp")
     (load"d:/pro/main.lsp")
     (load"d:/pro/quest.lsp")
     (load"d:/pro/disease.lsp")
     (load"d:/pro/decide.lsp"))


    (load-files)
```

```
(defun first()
     (send *terminal-io* :clear-screen)
     (skip 4)
     (print"          NEPHROTIC DISEASE DIGNOSTIC EXPERT SYSTEM")
     (skip 2)
     (print"                                          BY          ")
     (skip 2)
     (print"                                       SUMANT.M       ")
     (skip 1)
     (print"                                       BIJU PAUL  &   ")
     (skip 1)
     (print"                                    ALEXANDER MATHEW  ")
     (skip 3)
     (sleep 10)
     (send *terminal-io* :clear-screen))
(defun skip(x)
     (dotimes (y x t)
        (terpri)))


(first)
```

```
(defun main-module()
   (send *terminal-io* :clear-screen)
      (print"     WELCOME TO THE WORLD OF AUTO DISEASE DIAGNOSIS")
(terpri) (terpri)
(print"                PLEASE ENTER YOUR NAME:")
   (setf *name* (read))(terpri)
(print"                          ENTER YOUR AGE:")
   (setf *age* (read))(terpri)
(print"                   ENTER YOUR SEX(M/F):")
 (setf *sex* (read))
 (setf *entry-time* (get-universal-time))
(send *terminal-io* :clear-screen))


(main-module)
```

MUSIC.LSP GIVES A BACKGROUND MUSIC WHEN SOFTWARE STARTS UP INITIALLY.

```
(in-package 'user)


(DEFVAR *MUSIC-OCTAVE* 2)

(DEFVAR *MUSIC-SCALE* '(:C 494 :CS 466 :D 440 :DS 415 :E 392 :F 370
                        :FS 349 :G 330 :GS 311 :A 294 :AS 277 :B 262

(DEFVAR *MUSIC-TIME* .1)

(DEFCONSTANT SPEAKER-CONTROL-PORT #X61)

(DEFCONSTANT TIMER-SELECT-PORT #X43)

(DEFCONSTANT FREQUENCY-SET-PORT #X42)


(defun octavemove (action)
  (case action
        (:ou
         (decf *MUSIC-OCTAVE*))
        (:od
         (incf *MUSIC-OCTAVE*))))

(defun sethertz (hertz)
  (SYS:%IOPORT TIMER-SELECT-PORT #X0B6 nil)
  (SYS:%IOPORT FREQUENCY-SET-PORT (logand hertz #X0FF) nil)
  (SYS:%IOPORT FREQUENCY-SET-PORT (lsh hertz -8) nil))

(defun speaker (switch
                &aux (val (SYS:%IOPORT SPEAKER-CONTROL-PORT nil  nil)
  (case switch
        (:on
         (SYS:%IOPORT SPEAKER-CONTROL-PORT (logior val 3)  nil))
        (:off
         (SYS:%IOPORT SPEAKER-CONTROL-PORT (logand val #X0FC) nil))))

(defun my-beep (&optional (tone 1330) (time .15))
  (sethertz tone)
  (speaker :on)
  (sleep time)
  (speaker :off))

(defun play (music &optional (time *MUSIC-TIME*))
  (if (numberp music)
      (setq *MUSIC-TIME* music)
      (case music
            (:r  (sleep time))
            (:ou (octavemove :ou))
            (:od (octavemove :od))                    ;octave UP
            (otherwise                                ;octave DOWN
             (let ((freq (getf *MUSIC-SCALE* music)))
               (when (null freq)
                 (error "Unknown frequency: ~s" freq))
```

```
                    (my-beep (lsh freq *MUSIC-OCTAVE*)
                             *MUSIC-TIME*))))))

(defun playlist (notelist)
  (dolist (note notelist) (play note))
  (speaker :off))

(defun music ()
  (playlist '(.1 :g .05 :r .1 :g .05 :r .1 :g .05 :r .3 :ds
                 .5 :r .1 :f .05 :r .1 :f .05 :r .1 :f .05 :r .5

(SEND *TERMINAL-IO* :CLEAR-SCREEN)
```

DISEASE.LSP:-ASSIGNS SYMBOLS TO LISTS OF NAMES OF NEPHROTIC DISEASES, WHICH ARE ACCESSED LATER BY DECIDE.LSP.

```
(defun codes()
    (setf *AN* (list '1 'ACUTE 'NEPHRITIS))
     (setf *AE* (list '2 'ANGEONEUROTIC 'EDEMA))
(setf *PL* (list '3 'POLY 'ARTHIRITIS 'NODOSA 'AND 'LUPUOS 'AERYTHROMA
 (setf *AP* (list '4 'ACUTE 'PILITIS 'AND 'PILO 'NEPHRITIS))
 (setf *CN* (list '5 'CHRONIC 'NEPHRITIS ))
 (setf *MGN* (list '6 'MEMBRANEOUS 'GLOMERULO 'NEPHRITIS ))
  (setf *CCF* (list '7 'CONJUSTIVE 'CARDIAC 'FAILURE))
 (setf *NE* (list '8 'NUTRITIONAL 'EDEMA ))
 (setf *HE* (list '9 'HEPATIC 'EDEMA ))
 (setf *ADOTK* (list '10 'AMYLOID 'DISEASE 'OF 'THE 'KIDNEY ))
 (setf *ARF* (list '11 'ACUTE 'RENAL 'FAILURE ))
 (setf *CRF* (list '12 'CHRONIC 'RENAL 'FAILURE ))
 (setf *HT* (list '13 'HYPERTENSION ))
 (setf *CPD* (list '14 'CONGENITAL 'POLYCYSTIC 'DISEASE))
 (setf *CPN* (list '15 'CHRONIC 'PILO 'NEPHRITIS ))
 (setf *AA* (list '16 'ACUTE 'APPENDICITIS ))
 (setf *DP* (list '17 'DIAPHRAGMATIC 'PLEURICY ))
 (setf *PNA* (list '18 'PERI 'NEPHRIC 'ABSCESS ))
 (setf *GN* (list '19 'GLOMERULO 'NEPHRITIS.))
 (setf *RTD* (list '20 'RENAL 'TUBULE 'DEFECTS ))
 (setf *N* (list '21 'NEPHROLITHIASIS ))
 (setf *AFN* (list '22 'ACUTE 'FOCAL 'NEPHRITIS ))
-- More --


  (setf *NS* (list '23 'NEPHROTIC 'SYNDROME))
  (setf *RVT* (list '24 'RENAL 'VEIN 'THROMBOSIS ))
  (setf *PT* (list '25 'PERIPHERAL 'THROMBOSIS ))
  (setf *PE* (list '26 'PULMONARY 'EMBOLISM ))
  (setf *INP* (list '27 'INTESTICIAL 'NEPHRITIS 'DUE 'TO 'PHINACETIN))
(setf *INL* (list '28 'INTESTICIAL 'NEPHRITIS 'DUE 'TO 'LEAD))
(setf *ING* (list '29 'INTESTICIAL 'NEPHRITIS 'DUE 'TO 'GOUT 'DISEASE))
(setf *IN* (list '30 'IRRADIATION 'NEPHRITIS))
(setf *DGS* (list '31 'DIABETIC 'GLOMERULO 'SCLEROSIS))
(setf *AMY* (list '33 'RENAL 'AMYLOIDOSIS))
(setf *RTUBER* (list '34 'RENAL 'TUBERCULOSIS))
(setf *RTUMOR* (list '35 'RENAL 'TUMOR))
(setf *RA* (list '36 'RENAL 'ABSCESS))
  (send *terminal-io* :clear-screen))
(defun x()
    (if (equal *found* t) (load"d:/pro/output.lsp")
        (print"UNABLE")))

(codes)



D:\PRO>
```

```
;questionare module activates lab-tests
(defun questionare()
      (setf *found* nil)
        (send *terminal-io* :clear-screen)
        (print"PLEASE ANSWER THE QUESTIONS CAREFULLY!")
(t)
 (if (equal (y-or-n-p"DO YOU HAVE SOAR THROAT OR TONSILITIS?") t)
    (setf *infection* t)
                 (setf *infection* nil))
(t)
     (if (equal (y-or-n-p"DO YOU HAVE WOMITING?") t)
    (setf *womiting* t)
        (setf *womiting* nil))(t)
  (if (equal (y-or-n-p"DO YOU GET RED COLOURED URINE AT TIMES?") t)
        (setf *hematuria* t)
    (setf *hematuria* nil))(t)
(cond ((equal (y-or-n-p"DO YOU FEEL PUFFINESS ON THE BODY AFTER SLEEP?
    (setf *edema* t)
    (print"PLEASE SPECIFY WHERE EXACTLY YOU FEEL PUFFINESS")
    (print"                    FACE  ")
    (print"                    LEGS  ")
    (print"                    EYES  ")
    (print"                    BODY  ")
    (print"                  ABDOMEN ")
    (print"PLEASE ENTER HERE:")(setf *edema-organ* (read)))
          (t (setf *edema* nil)))
(t)
(if (equal (y-or-n-p"DID YOU EXPERIENCE ANY RENAL DISORDER?") t)
    (setf *renal-fn* t)
        (setf *renal-fn* nil))(t)
  (if (equal (y-or-n-p"DO YOU FEEL PUFFINESS ALL OVER BODY AT TIMES?")
          (setf *full-edema* t)
    (setf *full-edema* nil))(t)
(if (equal (y-or-n-p"DO YOU GET DIFFICULTY IN BREATHING NOWADAYS?") t)
          (setf *breath* t)
    (setf *breath* nil))(t)
(if (equal (y-or-n-p"DO YOU HAVE SWELLING OR BULGING NEAR LOWER
                      ABDOMEN(YOU CAN MAKE IT SURE BY KEEPING
                    YOUR LEFT HAND OVER ABDOMEN AND TAPPING ON IT
                       BY YOUR LEFT HAND'S FINGERS)?") t)
    (setf *ascetis* t)
    (setf *ascetis* nil))(t)
  (if (equal (y-or-n-p"DO YOU EXPERIENCE FITZ OR SIMILIAR PHENOMENA?") t
    (setf *fitz* t)
        (setf *fitz* nil))(t)
```

```
(if (equal (y-or-n-p"DO YOU FEEL A STATE OF COMA AT TIMES?") t)
                (setf *coma* t)
    (setf *coma* nil))(t)
   (if (equal (y-or-n-p"DID YOU EXPERIENCE MICTURATING AT SLEEP
                                        UNKNOWINGLY?") t)
    (setf *nocturia* t)
        (setf *nocturia* nil))(t)
 (if (equal (y-or-n-p"DO YOU UNINTENTIONALLY VOID URINE AT TIMES?") t
    (setf *enurisis* t)
        (setf *enurisis* nil))(t)
 (if (equal (y-or-n-p"DO YOU HISTORY OF KIDNEY-STONE REMOVAL?") t)
        (setf *stone* t)
    (setf *stone* nil))(t)
(if (equal (y-or-n-p"DO YOU FEEL LIKE MICTURATING MORE NUMBER OF TIME
                                        NOWADAYS THAN BEFORE?") t)
        (setf *frequency* t)
    (setf *frequency* nil))(t)
 (if (equal (y-or-n-p"DO YOU FEEL SUDDEN OR STRONG DESIRE TO
                                MICURATE NOWDAYS THAN USUAL?") t)
    (setf *urgency* t)
        (setf *urgency* nil))(t)
    (if (equal (y-or-n-p"DO YOU HAVE INFLAMMATION NEAR URINARY TRACT
                                        AND PENIS?") t)
    (setf *pyuria* t)
        (setf *pyuria* nil))(t)
     (if (equal (y-or-n-p"DO YOU HAVE PAIN ON TESTIS AND FLANK?") t)
       (setf *testis* t)
    (setf *testis* nil))(t)
(if (equal (y-or-n-p"ARE YOU EXPERIENCING RESTLESSNESS AT TIMES ?") t)
          (setf *rest* t)
    (setf *rest* nil))(t)
   (if (equal (y-or-n-p"DO YOU SWEAT ENORMOUSLY?") t)
        (setf *sweat* t)
    (setf *sweat* nil))(t)
(if (equal (y-or-n-p"DO YOU HAVE FEVER?") t)
        (setf *fever* t)
    (setf *fever* nil))(t)
     (if (equal (y-or-n-p"DO YOU HAVE ABDOMINAL PAIN?") t)
            (setf *abdomen-pain* t)
        (setf *abdomen-pain* nil))(t)
  (if (equal (y-or-n-p"     HOW IS YOUR CONSTIPATION?
                        DO YOU FEEL DIFFICUTY OR
                    IS CONSTIPATION RARE FOR PAST FEW DAYS ") t)
        (setf *constipation* t)
    (setf *constipation* nil))(t)
   (if (equal (y-or-n-p"DO YOU HAVE CHEST PAIN") t)
                        (setf *chest-pain* t)
    (setf *chest-pain* nil))(t)
```

```
(if (equal (y-or-n-p"WERE YOU OR ARE YOU COUGHING HEAVILY?") t)
    (setf *cough* t)
      (setf *cough* nil))(t)
 (if (equal (y-or-n-p"CAN YOU FIND BOILS OR SIMILIAR MARKS ON
                      YOUR BODY?") t)

     (setf *boils* t)
   (setf *boils* nil))(t)
(if (equal (y-or-n-p"DO YOU HAVE PAIN IN THE LOIN REGION?") t)
    (setf *loin* t)
      (setf *loin* nil))(t)
(if (equal (y-or-n-p"DO YOU HAVE PAIN IN THE RENAL REGION?") t)
    (setf *renal-pain* t)
      (setf *renal-pain* nil))(t)
(if (equal (y-or-n-p"DO YOU HAVE PAIN IN THE CALF MUSCLE?") t)
     (setf *calf* t)
    (setf *calf* nil))(t)
(if (equal (y-or-n-p"DO YOU SPIT BLOOD NOW A DAYS?") t)
     (setf *spit* t)
    (setf *spit* nil))(t)
(if (equal (y-or-n-p"WILL YOU GET MIGRANE OR HEADACHE?") t)
     (setf *migrane* t)
    (setf *migrane* nil))(t)
 (if (equal (y-or-n-p"WERE YOU CONSUMING EXCESS ASPIRIN OR
                      PHINACETIN FOR PAST FEW DAYS
                    FOR VOIDING PAINS AND HEADACHES") t)

          (setf *aspirin* t)
    (setf *aspirin* nil))(t)
 (if (equal (y-or-n-p"DID YOU ACCIDENTLY CONSUME OR INTAKE
                      PAINT OR SIMILIAR ITEMS") t)

          (setf *paint* t)
    (setf *paint* nil)) (t)
 (if (equal (y-or-n-p"DO YOU HAVE HISTORY OF CANCER TREATMENT
                      OR OVARY/TESTIS TREATMENTS BY MEANS
                        OF RADIATION THERAPIES?") t)

          (setf *radiation* t)
    (setf *radiation* nil))(t)
 (cond ((equal (y-or-n-p"DO YOU HAVE DIABETICS/HISTORY OF
                      DIABETICS TREATMENTS UNDERGONE?") t)

          (setf *diabetics* t)(t)
       (if (equal (y-or-n-p"DO YOU HAVE DIABETIC EYES?") t)
          (setf *eye* t) (setf *eye* nil)))
      (t (setf *diabetics* nil)))
 (t)
 (if (equal (y-or-n-p"DO YOU HAVE HISTORY OF HEART FAILURE?") t)
     (setf *heart* t)
                (setf *heart* nil))

 (t)
 (if (equal (y-or-n-p"DID YOU EXPERIENCE ANY MENTAL DISTURBANCE
```

```
                        OF ANY KIND LIKE FRIGHT AND LACK OF
                          INTREST IN DOING THINGS ?") t)

    (setf *mental* t)
          (setf *mental* nil))
(t)
 (if (equal (y-or-n-p"DO YOU HAVE HISTORY OF URINARY INFECTION?") t)
    (setf *urinary* t)
          (setf *urinary* nil))
  (if (equal (y-or-n-p"IS YOUR SKIN REDDENED ANY WHERE?") t)
        (setf *red* t)
     (setf *red* nil))
   (if (equal (y-or-n-p"DO YOU HAVE ANY SKIN INFECTION?") t)
     (setf *skin* t) (setf *skin* nil))
(t)
(send *terminal-io* :clear-screen)
   (print"YOU WILL HAVE TO UNDERGO THE FOLLOWING LAB TESTS")
   (print"                1.BLOOD TEST")
   (print"                2.URINE MICROSCOPIC TEST")
   (print"                3.URINE VOLUMETRIC TEST")
   (print"
  (cond ((equal (y-or-n-p"HAVE YOU ALREADY DONE WITH THE TESTS") t)
        (lab-tests))
        (t (later-lab))))


 (defun later-lab()
    (print"PLEASE UNDERGO ABOVE TESTS AND COME AT A LATER TIME")
    (print"                        GOOD BYE!!!"))

 (defun lab-tests()
     (print"PLEASE ENTER VALUES FROM YOUR BLOOD TEST REPORT")
 (cond ((equal (y-or-n-p"CHOLESTROL?") t)(t)
        (print"ENTER CHOLESTROL VALUE(%):")
          (setf *cholestrol* (read)))(t (setf *cholestrol* 175)))

 (terpri)
 (cond ((equal (y-or-n-p"UREA?") t)(t)
        (print"ENTER UREA VALUE(%):")
          (setf *urea* (read)))(t (setf *urea* 15)))
 (cond ((equal (y-or-n-p"PROTIENS?") t)(t)
        (print"ENTER PROTIENS VALUE(g/day):")
          (setf *protiens* (read)))(t (setf *protiens* 0.1)))
 (if (equal (y-or-n-p"NITROGEN?") t)
                (setf *nitrogen* t)(setf *nitrogen* nil))(t)
 (if (equal (y-or-n-p"IS LOSS OF BLOOD OF UREA INDICATED IN
                                BLOOD TEST REPORT?") t)
      (setf *loss* t)(setf *loss* nil))(t)
 (cond ((equal (y-or-n-p"URIC ACID?") t)(t)
        (print"ENTER URIC ACID VALUE(%):")
          (setf *uric-acid* (read)))(t (setf *uric-acid* 5)))
```

```
    (cond ((equal (y-or-n-p"CREATININ?") t) (t)
           (print"ENTER CREATININ VALUE(meq/litre):")
           (setf *creatinin* (read))) (t (setf *creatinin* 1.0)))
    (cond ((equal (y-or-n-p"POTASSIUM?") t) (t)
           (print"ENTER POTASSIUM VALUE(meq/litre):")
      (setf *k* (read))) (t (setf *k* 4)))
    (cond ((equal (y-or-n-p"CALCIUM?") t) (t)
           (print"ENTER CALCIUM VALUE(meq/litre):")
      (setf *ca* (read))) (t (setf *ca* 5)))
    (cond ((equal (y-or-n-p"PHOSPHATES?") t) (t)
           (print"ENTER PHOSPATES VALUE:")
      (setf *phosphates* (read))) (t (setf *phosphates* 0.5)))
    (cond ((equal (y-or-n-p"PHOSPHOLIPIDS") t) (t)
           (print"ENTER PHOSPHOLIPIDS VALUE(%):")
      (setf *lipids* (read))) (t (setf *lipids* 200)))
    (send *terminal-io* :clear-screen)
     (print"NOW ENTER THE FOLLOWING FROM THE URINE MICROSCOPY TEST SHEET")
    (cond ((equal (y-or-n-p"IS URINE TEST INDICATED NEGATIVE") nil)
                  (setf *urine-test* t) (continue))
           (t (setf *urine-test* nil) (urine-volume)))


    (defun continue()
    (print"ENTER SPECIFIC GRAVITY:")
    -- More --


      (setf *gravity* (read))(t)
    (print"ENTER PROTIENURIA VALUE(g/day):")
       (setf *protienuria* (read)) (t)
      (if (equal (y-or-n-p"LUCOSITES?") t)(setf *lucosites* t)
    (t)          (setf *lucosites* nil))
     (if (equal (y-or-n-p"RBC?") t)
    (setf *rbc* t)

                  (setf *rbc* nil))(t)
     (if (equal (y-or-n-p"EPC?") t)(setf *epc* t)
          (setf *epc* nil))(t)
     (if (equal (y-or-n-p"WBC?") t)
    (setf *wbc* t)
          (setf *wbc* nil))(t)
     (if (equal (y-or-n-p"PUS CELLS?") t)
                              (setf *pus* t)
          (setf *pus* nil))(t)
    (cond ((equal (y-or-n-p"SODIUM CONTENT?") t)
              (print"ENTER VALUE IN meq/litre:")
              (setf *sodium* (read)))
         (t (setf *sodium* 30)))(t)
    (if (equal (y-or-n-p"ALBUMIN?") t)
          (setf *albumin* t)
          (setf *albumin* nil))(t)
-- More --
```

```
 (if (equal (y-or-n-p"GLYCOSUREA?") t)(setf *gly* t)
                                      (setf *gly* nil))(t)
 (cond ((equal (y-or-n-p"TUBERCLE BASILLI?") t)
(setf *bas* t)(urine-volume))
       (t (setf *bas* nil) (urine-volume))))

(defun urine-volume()
 (send *terminal-io* :clear-screen)
(print"PLEASE ENTER FROM URINE VOLUMETRY TEST")
(t)
(print"ENTER THE VALUE OF VOUME OF URINE(ml/day):")
(setf *uv* (read))
 (cond ((equal (< *uv* 101) t) (setf *anuria* t) (setf *oliguria* t))
               (t (cond ((equal (< *uv* 401) t) (setf *oliguria* t))
                        (t (setf *oliguria* nil))))))


(defun t()
    (dotimes (count 3 t)
          (terpri)))
```

-- More --

```
(questionare)
```

-- More --

DECIDE.LSP:-THIS CONFIRMS
THE 90% EXISTENCE OF A
MAJOR SYNDROME
AND THEN IF NOT WILL
TRANSFER THE CONTROL
TO THE CORRESPONDING
DIFFERENTIAL DIAGNOSIS
OF THE DISEASE
AND SHIFTS THE CONTROL
TO OUTPUT.LSP.

```
;;; module confirming 90% existence of major sydromes.


(defun an-confirm()
  (cond ((equal (and (and (and (and (and (and (and
   (and (and (and (and (and (and (and
    (equal *infection* t)
    (equal *womiting* t))
     (equal *hematuria* t))
      (equal *edema* t))
       (equal *renal-fn* t))
        (equal *oliguria* t))
      .   (and (< *cholestrol* 266) (> *cholestrol* 153)))
           (< *uv* 401))
            (or (equal *gravity* 1.01) (and (> *gravity* 1.01)
             (< *gravity* 1.1))))
             (equal *urine-test* t))
              (equal *lucosites* t))
               (equal *rbc* t))
                (equal *epc* t))
                 (> *urea* 20))
                  (> *protienuria* 4)) t) (check1))
   (t (an-differ))))
-- More --


(defun check1()
(cond ((equal (y-or-n-p"IS ANY THING INDICATED IN YOUR BLOOD TEST THAT
                        HEMATURIA DUE TO OTHER ORGANS OTHER THAN
                         KIDNEY?") t) (setf *disease* *pl*)
                                (setf *found* t) (x))
                      (t (setf *disease* *an*)
 (setf *found* t) (x) )))



(defun mgn-confirm()
  (cond ((equal (and (and (and (and (and (and (and
                (and (and (and (and
   (equal *infection* nil)
      (equal *breath* nil))
        (equal *hematuria* nil))
         (equal *edema* t))
          (equal *edema-organ* 'body))
           (equal *renal-fn* nil))
             (> *cholestrol* 299))
               (equal *urine-test* t))
                 (equal *rbc* nil))
                  (equal *epc* t))
                   (< *protiens* 5))
-- More --
```

```
                    (> *protienuria* 25)) t) (check2))
                      (t (mgn-differ))))


(defun check2()
  (cond ((equal (y-or-n-p"HAVE YOU TAKEN AN X-RAY OF THE ABDOMEN?") t)
    (if (equal (y-or-n-p"IS A LARGE SPLEEN INDICATED?") t)
                   (setf *spleen* t) (setf *spleen* nil))))
 (cond ((equal (y-or-n-p"HAVE YOU UNDERGONE CONGO-RED TEST?") t)
    (if (equal (y-or-n-p"IS RESULT POSITIVE?") t)
                   (setf *congo-red* t) (setf *congo-red* nil))))
    (cond ((equal (and
                   (equal *spleen* t)
                    (equal *congo-red* t)) t)
                     (setf *disease* *amy*)
                       (setf *found* t) (x)))
    (cond ((equal *ascetis* t)
                     (setf *disease* *he*)
                      (setf *found* t) (x)))
    (cond ((equal (and (equal *ascetis* nil) (equal *found* nil)) t)
                     (setf *disease* *ne*)
                            (setf *found* t) (x)))
    (cond ((equal *found* nil) (setf *found* t)))
           (if (equal *found* t) (setf *diseease* *mgn*))
-- More --

             (x))


(defun arf-confirm()
(cond ((equal (and (and (and (and (and (and (and
                 (and (and (and (and (and (and
                 (and (and (and (and
       (equal *womiting* t)
        (= *sodium* 40))
         (equal *edema* t))
           (equal *oliguria* t))
             (equal *anuria* t))
             (< *uv* 101))
               (equal *urine-test* t))
              (equal *lucosites* t))
               (equal *rbc* t))
                (equal *epc* t))
                (< *gravity* 0.101))
                    (equal *wbc* t))
                  (equal *loss* t))
                 (> *urea* 20))
                  (> *protienuria* 10))
                   (> *uric-acid* 8))
                    (> *creatinin* 1.5))
-- More --
```

```
                    (> *k* 5))
                      (equal *nitrogen* t)) t)
            (setf *disease* *arf*)
             (setf *found* t) (x))))


(defun crf-confirm()
  (cond ((equal (y-or-n-p"HAVE YOU YOUR BLOOD PRESSURE REPORT") t)
        (print"ENTER CYCTOLIC PRESSURE:")
        (setf *cyst* (read))
        (print"ENTER DIASTOLIC PRESSURE:")
        (setf *diast* (read))))
        (if (equal (and (> *cyst* 120) (> *diast* 80)) t)
    (setf *hyper* t))
  (cond ((equal (y-or-n-p"HAVE YOU TAKEN AN X-RAY OF KIDNEY") t)
         (print"ENTER KIDNEY SIZE")
         (print"   LARGE")
         (print"   SMALL")
         (print"ENTER:")
         (setf *kidney* (read))))
 (cond ((equal (y-or-n-p"IS YOUR KIDNEY X-RAY INDICATING AUSTEO-MALIYSIA")
        (setf *austeo* t))
       (t (setf *austeo* nil)))
  (cond ((equal (and (and (and (and (and (and (and
-- More --

               (and (and (and (and (and (and (and
                 (and (and (and
                       (equal *hyper* t)
                         (equal *renal-fn* t))
                           (equal *loss* t))
                            (equal *fitz* t))
                             (equal *coma* t))
                              (equal *austeo* t))
                                (> *uv* 3000))
                                 (equal *urine-test* t))
                                   (equal *rbc* t))
                                     (equal *epc* t))
                                      (equal *gravity* 1.010))
                                       (> *urea* 20))
                                        (equal *kidney* 'SMALL))
                                         (> *phosphates* 0.63))
                                          (> *lipids* 250))
                                           (> *creatinin* 1.5))
                                             (< *ca* 4))
                                              (equal *nitrogen* t)) t)
                                               (setf *disease* *crf*)
                                                (setf *found* t) (x))
    (t (crf-differ))))

-- More --
```

```
(defun rtd-confirm()
   (if (equal (y-or-n-p"IS X-RAY REPORT INDICATING OSTEODYSTROHY") t)
    (setf *osteodystrophy* t)
    (setf *osteodystrophy* nil))
   (cond ((equal (y-or-n-p"HAVE YOU TAKEN AN X-RAY OF KIDNEY") t)
        (print"ENTER KIDNEY SIZE")
        (print"   LARGE")
        (print"   SMALL")
        (print"ENTER:")
        (setf *kidney* (read))))
   (cond ((equal (and (and (and (and (and (and (and
              (and (and (and (and
              (and (and
     . (equal *anuria* t)
        (equal *oliguria* t))
          (equal *nocturia* t))
            (equal *enurisis* t))
             (equal *hematuria* t))
              (< *uv* 100))
               (equal *urine-test* t))
                (equal *rbc* t))
                 (equal *epc* t))
                   (= *gravity* 0.1010))
-- More --

                     (equal *loss* t))
                      (> *urea* 20))
                       (equal *kidney* 'LARGE))
                        (equal *osteodystrophy* t)) t)
                        (setf *disease* *rtd*)(setf *found* t)

(x))))


(defun nephro-confirm()
   (cond ((equal (and (and (and (and
              (and (and (and
              (and (and (and
       (equal *urgency* t)
         (equal *frequency* t))
           (equal *stone* t))
             (equal *rest* t))
    (equal *sweat* t))
    (equal *hematuria* t))
                          (equal *urine-test* t))
                           (equal *rbc* t))
               (equal *pus* t))
             (equal *pyuria* t))
                                  (equal *testis* t)) t)
            (setf *disease* *n*)(setf *found* t)(x))))
-- More --
```

```
(defun ns-confirm()
                    (cond ((equal (and (and (and (and
               (and (and (and
               (and (and
      (equal *fever* t)
        (equal *abdomen-pain* t))
          (equal *edema* t))
            (equal *constipation* t))
      (equal *chest-pain* t))
    (equal *sweat* t))
               (equal *boils* t))
                    (equal *cough* t))
          (> *protienuria* 4))
      (> *cholestrol* 300)) t) (check3))))

(defun check3()
(cond ((equal *spit* t) (setf *disease* *pe*) (setf *found* t)(x))
      ((equal *calf* t) (setf *disease* *pt*) (setf *found* t)(x))
      ((equal (and (< *uv* 100) (equal *loin* t)) t) (setf *disease* *rvt*)
                        (setf *found* t)(x))
        (t (setf *disease* *ns*) (setf *found* t) (x))))

-- More --


(defun in-phinacetin-confirm()
        (cond ((equal (and
           (equal *migrane* t)
      (equal *aspirin* t)) t)
       (setf *disease* *inp*) (setf *found* t)(x))))


(defun in-lead-confirm()
      (cond ((equal (and (and
         (< *age* 11)
        (equal *paint* t))
     (equal *kidney* 'SMALL)) t)
     (setf *disease* *inl*) (setf *found* t) (x))))


(defun in-gout-confirm()
     (cond ((equal (and (and (and (and (and
       (equal *hematuria* t)
       (equal *loin* t))
    (equal *renal-pain* t))
    (> *protienuria* 4))
   (> *uric-acid* 8))
  (equal *loss* t)) t)
-- More --
```

```
   (setf *disease* *ing*)(setf *found* t)(x))))


(defun irradiation-confirm()
 (cond ((equal (and (and (and
     (equal *albumin* t)
   (equal *edema* t))
  (equal *renal-fn* t))
(equal *radiation* t)) t)
   (setf *disease* *in*)(setf *found* t)(x))))


(defun dgs-confirm()
(cond ((equal (and (and (and (and
                (and (and (and
                (and (and
       (equal *diabetics* t)
         (equal *gly* t))
           (equal *edema* t))
             (equal *albumin* t))
       (equal *renal-fn* t))
     (equal *heart* t))
                (equal *mental* t))
                      (equal *eye* t))
-- More --

           (equal *urinary* t))
        (< *protiens* 5)) t)
(setf *disease* *dgs*)(setf *found* t)(x))))


(defun con-polycystic-confirm()
 (cond ((equal (y-or-n-p"HAVE YOU UNDERGONE INTRAVENOUS PILOGRAPHY?") T)
       (if (equal (y-or-n-p"BAT WING SHADOW INDICATED?") T)
           (setf *bat* t) (setf *bat* nil))))
       (cond ((equal
              (and (and
               (and
                (equal *kidney* 'LARGE)
                 (equal *history* t))
                 (equal *albumin* t))
                  (equal *bat* t)) t)
                   (setf *disease* *cpd*)(setf *found* t)(x))))


(defun renal-amy-confirm()
(if (equal(y-or-n-p"IS YOUR ABDOMEN X-RAY INDICATING SPLEEN ENLARGEMENT?")
     (setf *spleen* t) (setf *spleen* nil))
(cond ((equal (y-or-n-p"HAVE YOU UNDERGONE THE CONGO-RED TEST") t)
      (if (equal (y-or-n-p"IS THE TEST INDICATED POSITIVE") t)
-- More --
```

```
          (setf *congo-red* t) (setf *congo-red* nil))))
     (cond ((equal (and (and
         (> *protienuria* 40)
           (equal *spleen* t))
            (equal *congo-red* t)) t)
             (setf *disease* *amy*)(setf *found* t)(x))))


(defun renal-abscess-confirm()
   (if (equal (y-or-n-p"IS YOUR BLOOD CULTRE POSITIVE?") t)
     (setf *blood-cultre* t) (setf *blood-cultre* nil))
       (cond ((equal
              (and (and (and
              (and (and (and
                (equal *fever* t)
                  (equal *skin* t))
                    (equal *red* t))
                     (equal *abdomen-pain* t))
                      (equal *edema* t))
                        (equal *loin* t))
                         (equal *blood-culture* t))t)
                          (setf *disease* *ra*) (setf *found* t)
(x))))

-- More --


(defun renal-tuber-confirm()
     (cond ((equal (and (and (and (and (and (and (and
      (if (equal (y-or-n-p"IS X-RAY ABDOMEN INDICATING CALCIFICATION?") t)
        (setf *calcification* t) (setf *calcification* nil))
       (equal *frequency* t)
       (equal *urgency* t))
       (equal *epc* t))
     (equal *hematuria* t))
     (equal *loin* t))
      (equal *calcification* t))
      (equal *bas* t))
      (> *protienuria* 0.15)) t)
      (setf *disease* *rtuber*)(setf *found* t)(x))))


(defun renal-tumor-confirm()
     (cond ((equal (and (and (and
       (equal *loin* t)
     (equal *renal-pain* t))
    (equal *hematuria* t))
    (equal *albumin* t)) t)
    (setf *disease* *rtumor*)(setf *found* t)(x))))

-- More --
```

```
(defun pilitis-confirm()

      (cond ((equal (and (and (and (and (and (and (and
               (and  (and (and (and (and (and (and (and
   (equal *infection* t)
    (equal *womiting* t))
     (equal *hematuria* t))
      (equal *edema* t))
       (equal *renal-fn* t))
        (equal *oliguria* t))
        (equal *strangury* t))
         (and (< *cholestrol* 266) (> *cholestrol* 153)))
          (< *uv* 401))
           (or (equal *gravity* 1.01) (and (> *gravity* 1.01)
            (< *gravity* 1.1))))
            (equal *urine-test* t))
             (equal *lucosites* t))
              (equal *pus* t))
              (equal *rbc* t))
               (equal *epc* t))
                (> *urea* 20))
                 (> *protienuria* 4)) t)
  (if (equal *albumin* t) (setf *disease* *gn*)
-- More --
                      (setf *disease* *ap*))
               (setf *found* t) (x))
 (t (pilitis-differ))))


 (defun afn-confirm()
    (cond ((equal (and (and (and (and (and (and (and
                (and (and (and
                (and (and (and (and
   (equal *infection* t)
    (equal *womiting* t))

  (equal *hematuria* t))
      (equal *edema* nil))
       (equal *renal-fn* nil))
        (equal *oliguria* t))
         (and (< *cholestrol* 266) (> *cholestrol* 153)))
          (< *uv* 401))

   (or (equal *gravity* 1.01) (and (> *gravity* 1.01)
                                    (< *gravity* 1.1))))
             (equal *urine-test* t))
              (equal *lucosites* t))

-- More --
```

```
   (equal *rbc* t))
                (equal *epc* t))
                 (> *urea* 20))
                  (> *protienuria* 4)) t)
               (setf *disease* *afn*) (setf *found* t) (x))))

(defun an-differ()
 (cond ((equal (and (and (and (and (and (and (and
                 (and (and (and
   (equal *infection* t)
    (equal *womiting* t))
     (equal *hematuria* t))
      (equal *edema* t))
       (equal *renal-fn* t))
        (equal *oliguria* t))
         (and (< *cholestrol* 266) (> *cholestrol* 153)))
          (< *uv* 401))
           (or (equal *gravity* 1.01) (and (> *gravity* 1.01)
                                           (< *gravity* 1.1))))
            (equal *urine-test* nil))
                          (> *urea* 20)) t)
          (setf *disease* *ae*)(setf *found* t)(x)))

   (cond ((equal (and (and (and (and (and (and (and (and
-- More --

                (and (and (and (and (and (and (and (and
    (equal *infection* t)
     (equal *womiting* t))
      (equal *hematuria* t))
        (equal *edema* nil))
         (equal *anuria* t))
          (equal *renal-fn* t))
            (equal *oliguria* t))
 (equal (and (< *cholestrol* 266) (> *cholestrol* 153)) t))
                 (equal (< *uv* 101) t))
     (equal (or (equal *gravity* 1.01) (and (> *gravity* 1.01)
                                            (< *gravity* 1.1))) t))
              (equal *urine-test* t))
               (equal *lucosites* t))
                (equal *rbc* t))
                 (equal *epc* t))
                  (> *urea* 20))
                   (> *protienuria* 4)) t)
                     (setf *disease* *cn*) (setf *found* t) (x) )))


 (defun mgn-differ()

 (cond ((equal (and (and (and (and (and (and (and
 -- More --
```

```
        (equal *coma* t))
          (equal *austeo* t))
           (> *uv* 3000))
            (equal *urine-test* t))
             (equal *rbc* t))
              (equal *epc* t))
               (= *gravity* 1.010))
                (> *urea* 20))
                 (equal *kidney* 'LARGE))
                  (> *phosphates* 0.63))
                   (> *lipids* 250))
                    (> *creatinin* 1.5))
                     (equal *ca* t))
                      (equal *nitrogen* t))

   t)
    (setf *disease* *cpd*)
    (setf *found* t)(x)))

 (cond ((equal (and (and (and (and (and (and (and
                (and (and (and (and (and (and (and
                 (and (and (and
    (equal *hyper* nil)
     (equal *renal-fn* t))
       (equal *loss* t))
-- More --

         (equal *fitz* t))
           (equal *coma* t))
             (equal *austeo* t))
              (> *uv* 3000))
                (equal *urine-test* t))
                 (equal *rbc* t))
                  (equal *epc* t))
                   (= *gravity* 1.010))
                    (> *urea* 20))
                     (equal *kidney* 'SMALL))
                      (> *phosphates* 0.63))
                       (> *lipids* 250))
                        (> *creatinin* 1.5))
                         (equal *ca* t))
                          (equal *nitrogen* t))

     t)
     (setf *disease* *cpn*)
     (setf *found* t)(x)))
  )


  (defun pilitis-differ()
       (cond ((equal (and (and (and (and (and (and (and
  -- More --
```

```
              (and (and (and
  (equal *infection* nil)
  (equal *breath* t))
       (equal *hematuria* nil))
     (equal *edema* t))
      (equal *renal-fn* nil))
            (> *cholestrol* 299)))
              (equal *urine-test* t))
                (equal *rbc* nil))
              (equal *epc* t))
               (< *protiens* 5)) t)
                (setf *disease* *ccf*) (setf *found* t) (x))))


(defun crf-differ()
    (cond ((equal (and (and (and (and (and (and (and
                (and (and (and (and (and (and
                  (and (and (and
   (equal *hyper* nil)
    (equal *renal-fn* t))
      (equal *loss* t))
        (equal *fitz* t))
          (equal *coma* t))
            (equal *austeo* t))
-- More --

              (> *uv* 3000))
                (equal *urine-test* t))
                  (equal *rbc* t))
                   (equal *epc* t))
                     (= *gravity* 1.010))
                      (> *urea* 20))
                       (equal *kidney* 'LARGE))
                        (> *phosphates* 0.63))
                         (> *lipids* 250))
                          (> *creatinin* 1.5))
                           (equal *ca* t))
                            (equal *nitrogen* t)) t)
      (setf *disease* *ht*)
      (setf *found* t)
      (x)))


     (cond ((equal (and (and (and (and (and (and (and
                 (and (and (and (and (and (and
                   (and (and (and
     (equal *hyper* t)
      (equal *renal-fn* t))
        (equal *loss* t))
          (equal *fitz* t))
  -- More --
```

```
          (and (and (and (and (and (and (and (and (and
(equal *infection* t)
 (equal *womiting* t))
  (equal *hematuria* t))
   (equal *edema* t))
    (equal *renal-fn* t))
     (equal *oliguria* t))
     (equal *strangury* t))
      (and (< *cholestrol* 266) (> *cholestrol* 153)))
       (< *uv* 401))
        (or (equal *gravity* 1.01) (and (> *gravity* 1.01)
                                        (< *gravity* 1.1))))

          (equal *urine-test* t))
      (equal *lucosites* t))
            (equal *rbc* t))
                  (equal *epc* t))
              (> *urea* 20))
     (equal *pus* nil))
               (> *protienuria* 4)) t)
(setf *disease* *aa*)(setf *found* t)(x)))

 (cond ((equal (and (and (and (and (and.(and (and
               (and (and (and

-- More --

    (equal *infection* t)
     (equal *womiting* t))
      (equal *hematuria* t))
       (equal *edema* t))
        (equal *renal-fn* t))
         (equal *oliguria* t))
         (equal *strangury* t))
          (and (< *cholestrol* 266) (> *cholestrol* 153)))
           (< *uv* 401))
                    (equal *urine-test* nil))
                        (> *urea* 20)) t)
  (setf *disease* *dp*)(setf *found* t)(x)))


       (cond ((equal (and (and (and (and (and (and (and
                  (and  (and (and (and (and     (equal *infection* t)
      (equal *womiting* t))
       (equal *hematuria* t))
        (equal *edema* t))
         (equal *renal-fn* t))
           (equal *oliguria* t))
           (equal *strangury* t))
            (and (< *cholestrol* 266) (> *cholestrol* 153)))
             (< *uv* 401))

   -- More --
```

```
        (or (equal *gravity* 1.01) (and (> *gravity* 1.01)
                                        (< *gravity* 1.1))))
          (equal *urine-test* nil))
        (equal *pus* nil))
                              (> *urea* 20)) t)
(setf *disease* *pna*)(setf *found* t)(x))))
```

```
(defun confirm()
       (if (equal (and (equal *paint* t) (< *age* 10)) t)
           (test))
       (if (equal *found* nil) (in-lead-confirm))
        (if (equal *found* nil) (an-confirm))
         (if (equal *found* nil) (mgn-confirm))
          (if (equal *found* nil) (arf-confirm))
           (if (equal *found* nil) (crf-confirm))
            (if (equal *found* nil) (rtd-confirm))
             (if (equal *found* nil) (nephro-confirm))
              (if (equal *found* nil) (ns-confirm))
               (if (equal *found* nil) (in-phinacetin-confirm))
       (if (equal *found* nil) (in-gout-confirm))
        (if (equal *found* nil) (in-gout-confirm))
-- More --
        (if (equal *found* nil) (in-gout-confirm))
-- More --
                      (if (equal *found* nil) (irradiation-confirm))
                       (if (equal *found* nil) (dgs-confirm))
                        (if (equal *found* nil) (con-polycystic-confirm))
  (if (equal *found* nil) (renal-abscess-confirm))
   (if (equal *found* nil) (renal-tuber-confirm))
    (if (equal *found* nil) (renal-tumor-confirm))
     (if (equal *found* nil) (pilitis-confirm))
      (if (equal *found* nil) (print"UNABLE")))
  (defun test()
       (print"ENTER YOU KIDNEY SIZE FROM X-RAY REPORT")
        (print"              LARGE")
         (print"              SMALL")
          (print"             NORMAL")
           (print"ENTER HERE:")
            (setf *kidney* (read)))


   (confirm)
```

```
defun derived()
(setf *exit-time* (get-universal-time))
(setf *diff* (time-difference *exit-time* *entry-time*))
(send *terminal-io* :clear-screen)
(print"~~~~~~~~~~~~~~~~~~~~~~~~~CURRENT OUTPATIENT REPORT SHEET~~~~~~~~~~~~~~~~~~~~~
~~~")
(t)                          NAME OF PATIENT :~a"*name*)
(format t"
(t)                                        AGE :~d"*age*)
(format t"
(t)                                        SEX :~a"*sex*)
(format t"
(t)                          DETECTED DISEASE :~s"(cdr *disease*))
(format t"
(t)                            ENTRY DATE/TIME :")
(format t"                                         (print-universal-time *entry-time*)

(t)
(format t"   CONSULTATION FINISH TIME :")
                                               (print-universal-time *exit-time*)

(t)                  TOTAL DIAGNOSIS TIME :~d  INTERNAL UNITS" *diff*)
(format t"
(sleep 30))

-- More --

(t)                          NAME OF PATIENT :~a"*name*)
(format t"
(t)                                        AGE :~d"*age*)
(format t"
(t)                                        SEX :~a"*sex*)
(format t"
(t)                          DETECTED DISEASE :~s"(cdr *disease*))
(format t"
(t)                            ENTRY DATE/TIME :")
(format t"                                         (print-universal-time *entry-time*)

(t)
(format t"   CONSULTATION FINISH TIME :")
                                               (print-universal-time *exit-time*)

(t)                  TOTAL DIAGNOSIS TIME :~d  INTERNAL UNITS" *diff*)
(format t"
(sleep 30))

-- More --


(derived)
```

```
;;; controls the flow of diagnosis to confirmation of diseases
(defun flow-control()
    (send *terminal-io* :clear-screen)(terpri) (terpri)
    (send *terminal-io* :clear-screen)(terpri) (terpri)
  (print"THINKING.....................")
   (cond ((equal (and (and
      (and (> *age* 10) (< *age* 20))
        (equal *edema-organ* 'face))
          (equal *infection* t)) t)
          (an-confirm)))

    (if (equal *edema-organ* 'body)
       (mgn-confirm))

    (if (equal *sodium* 40)
       (arf-confirm))

    (cond ((equal (and (equal *fitz* t) (equal *coma* t)) t)
            (send *terminal-io* :clear-screen) (crf-bp)

       (crf-ray)
       (if (equal *austio* t)
       (crf-confirm))))

       (cond ((equal (and
          (equal *dysuria* t)
    -- More --
            (equal *strangury* t)) t)
          (pilitis-confirm)))

         (cond ((equal (and
            (equal *nocturia* t)
           (equal *enurisis* t)) t)
         (rtd-confirm)))


        (if (equal *stone* t) (nephro-confirm))

        (cond ((equal (and
          (equal *boils* t)
           (equal *edema-organ* 'abdomen)) t)(ns-confirm)))

          (if (equal *migrane* t) (in-phinacetin-confirm))

         (cond ((equal (and
                (equal *paint* t)
              (and (> *age* 3) (< *age 10))) t)
            (in-lead-confirm)))

           (cond ((equal (and
                (equal *loss* t)
        -- More --
```

```
      (equal *loin* t)) t)
   (in-gout-confirm)))

(cond ((equal (and
     (equal *radiation* t)
    (equal *albumin* t)) t)
 (irradiation-confirm)))

 (cond ((equal (and
     (equal *diabetics* t)
      (equal *mental* t)) t)
     (dgs-confirm)))

  (cond ((equal (and
       (equal *congo-red* t)
        (equal *spleen* 'large)) t)
      (con-polycystic-confirm)))

   (cond ((equal (and
      (equal *skin* t)
        (equal *abdomen-pain* t)) t)
       (renal-abscess-confirm)))

    (cond ((equal (and (and
    -- More --

        (equal *frequency* t)
         (equal *urgency* t))
        (equal *calcification* t)) t)
       (renal-tuber-confirm)))

     (cond ((equal (and
       (equal *edema-organ* 'legs)
        (equal *swell* t)) t)
         (renal-tumor-confirm))))

     (defun crf-ray()
     (cond ((equal (y-or-n-p"HAVE YOU GOT AN X-RAY OF THE KIDNEY?") t)
            (cond ((equal (y-or-n-p"AUSTIOMALYSIA INDICATED?") t)
                (setf *austio* t)(setf *kidney* 'small))
              (t (setf *austio* nil) (setf *kidney* 'normal)))
        (t (print"PLEASE DO IT AND REDO CONSULTATION,GOOD BYE!(")))))


       (defun crf-bp()
         (print"ENTER YOUR BLOOD PRESSURE (RECENTLY TAKEN):")

              (print"ENTER CYSTOLIC PRESSURE:")
              (setf *cyst* (read))
              (print"ENTER DIASTOLIC PRESSURE:")
       -- More --
```

```
(cond ((equal (y-or-n-p"HAVE YOU GOT AN X-RAY OF THE KIDNEY?") t)
       (cond ((equal (y-or-n-p"AUSTIOMALYSIA INDICATED?") t)
              (setf *austio* t)(setf *kidney* 'small))
             (t (setf *austio* nil) (setf *kidney* 'normal))))
      (t (print"PLEASE DO IT AND REDO CONSULTATION,GOOD BYE!!"))))


(defun crf-bp()
  (print"ENTER YOUR BLOOD PRESSURE (RECENTLY TAKEN):")

       (print"ENTER CYSTOLIC PRESSURE:")
       (setf *cyst* (read))
       (print"ENTER DIASTOLIC PRESSURE:")
-- More --

       (setf *diast* (read))
    (if (equal (and (> *cyst* 120) (> *diast* 80)) t)
      (setf *hyper* t) (setf *hyper* nil)))

 (flow-control)



 D:\PRD>
```