```
unit help;

  interface
     uses
MemTypes,QuickDraw,OSIntf,ToolIntf,PackIntf,PrintTraps,PasLibIntf
,
             globals;

     procedure helpp;

implementation


(*******************************)
(*                             *)
(*   pauses but 'escape' is    *)
(*   trapped.                  *)
(*                             *)
(*******************************)
          procedure slow(var i:longint);
             var op:char;
          begin
            pause2(op);
            if op = escp then i:=-1;
            space(3);
          end;



procedure helpNC(var i:longint);
var nnn:longint;
begin
    i:=0;
    writeln('                 Adding a New Command');
    writeln;
    writeln(' A new command, say "newcom", is added to the system
as follows.');
    writeln;
    nnn:=ninst-1;
    writeln('    a)  In the "const" section, decrement');
    writeln('        the constant "ninst" by one (note that it is
a negative');
    writeln('        longint) to "',nnn:3,'". Then add a new line
of');
    writeln;
    writeln('                 newcom:=',nnn:3,';');
    writeln;
    writeln('    b)  In "procedure startup", add');
    writeln;
    writeln('                 parameter[newcom]:=''newcom'';');
```

```
    writeln;
    writeln('    c)  Add a procedure, say "pnew", to execute the
new command');
    writeln('        (anywhere prior to "procedure input".)');
    writeln;
    writeln('    d)  In "procedure input", add a new case to the
case');
    writeln('        statement - "case x of .... end".');
    writeln('        The new case should be');
    writeln;
    writeln('                  new: pnew;');
    pause;
  end;



procedure helpNT(var i:longint);
label 99;
var nnn :longint;
begin
    i:=0;
    writeln('              Adding a New Theorem');
    writeln;
    writeln(' Each theorem "iii" requires a procedure named
"riii". The');
    writeln('constant "cnumrules" (found in the "const" section)
is the');
    writeln('number of theorems the system is designed to hold.
The last');
    writeln('few of these may be just empty shells. The
variable');
    writeln('"actualnumrules" (found in procedure "startup") is
the');
    writeln('number of theorems actually implemented at this
time. If');
    writeln('there are no empty shells (actualnumrules=cnumrules)
you will');
    writeln('have to create one in step (a) below. The number
assigned to');
    writeln('your new theorem will be nnn (=actualnumrules+1).');
    writeln;
    write('  At this time cnumrules=',cnumrules:4,' and
actualnumrules=');
    writeln(actualnumrules:4,',');
    nnn:=cnumrules-actualnumrules;
    write('so you will ');
    if nnn > 0 then write('not ');
    writeln('have to add an empty shell in (a) below.');
    nnn:=actualnumrules+1;
    writeln;
    writeln('  (a) First, if necessary, increase "cnumrules" by
```

```
one. It');
    writeln('      is in the "const" section. Also, if necessary,
you must');
    writeln('      add a new empty shell for the procedure for
your new');
    writeln('      rule (right before "procedure delthm"). The
form of the');
    writeln('      new empty shell is:');
    writeln;
    slow(i);
     if i < 0 then goto 99;
    writeln('        procedure r',nnn:3,';');
    writeln('        (*************************)');
    writeln('        (*                       *)');
    writeln('        (*   documentation       *)');
    writeln('        (*                       *)');
    writeln('        (*************************)');
    writeln('        begin');
    writeln('          if activerule[',nnn:3,'] then');
    writeln('            begin');
    writeln('              rule:=''',nnn:3,'/ '';');
    writeln('                .');
    writeln('               code');
    writeln('                .');
    writeln('            end;');
    writeln('         end;');
    writeln;
    writeln;
    writeln('  (b) In "procedure eval;", do the following:');
    writeln('            if your new theorem uses(in procedure
r',nnn:3,')');
    writeln('            invariants x, y, etc., add to each case
x, case y,');
    writeln('            etc.  a statement -   "r',nnn:3,';"');
    slow(i);
     if i < 0 then goto 99;
    space(6);
    writeln('  (c) In "procedure executerule(i:longint);", add a
new case');
    writeln('      to the case statement:');
    writeln('               ',nnn:3,': r',nnn:3,';');
    writeln;
    writeln('  (d) In "procedure ruletext", near the end, add a
write');
    writeln('      statement to print the text of the new
theorem. Note');
    writeln('      if the description only reguires one line
then');
    writeln;
    writeln('             ',nnn:3,': write('' .....  '');');
    writeln;
    writeln('         is sufficient. Otherwise you must check to
```

```
see if a');
    writeln('         "short" version is wanted (trace = 1) or a
longer more');
    writeln('         complete version is desired.');
    writeln;
    slow(i);
      if i < 0 then goto 99;
    writeln('  (e) In "procedure startup", find the set
assignment');
    writeln('      statements- they are of the form (for theorem
iii)');
    writeln;
    writeln('              rulemx[iii]:=[a,b,...];');
    writeln('                rulemn[iii]:=[x,y,...];');
    writeln;
    writeln('              The MAX of invariants a, b,... and
the MIN of');
    writeln('            invariants x, y, ... are used in the
evaluation');
    writeln('            of theorem iii (i.e., in "procedure
riii;").');
    writeln;
    writeln('      You must add similar statements for the new
"',nnn:3,'".');
    writeln;
    writeln('      At the end of this group of statements
(probably');
    writeln('      where you just added) is the assignment for
the variable');
    writeln('      "actualnumrules". It must now be assigned
',nnn:3,'.');
    writeln;
    writeln('      Also, just following this, and if the text
description');
    writeln('      given in step (d) requires k > 1 lines, then
add the');
    writeln('      statement');
    writeln;
    writeln('                lsinthm[',nnn:3,']:=k;');
    pause;
 99: end;

procedure helpNI(var i:longint);
var nnn:longint;
begin
    i:=0;
    writeln('               Adding a New Invariant');
    writeln;
    writeln('  To add a new invariant name, say xxxxxx. The name
must be 1-6');
    writeln('alphanumeric characters with no embedded blanks and,
if it is k');
```

```
    writeln('characters in length, they must be different from
the first k');
    writeln('characters of any other name (invariant or
command).');
    writeln;
    nnn:=nparam+1;
    writeln('     (a) In the "const" section, increase the value
of the');
    writeln('         constant "nparam" by one to ',nnn:3,'. Then
add a new');
    writeln('         constant with value nnn as follows:');
    writeln;
    writeln('               xxxxxx=',nnn:3,';');
    writeln;
    writeln('     (b) In "procedure startup", add the
statement');
    writeln;
    writeln('              parameter[xxxxxx]:=''xxxxxx'';');
    writeln;
    writeln('         If xxxxxx is to be a binary invariant, add
its name');
    writeln('         to the binary set assignment statement:');
    writeln;
    writeln('                  bparam:=[xxxxxx,.....];');
    writeln;
    slow(i);
      if i = 0 then
        begin
    space(3);
    writeln('      (c) In "procedure eval", add a new case to the
case');
    writeln('          statement:');
    writeln;
    writeln('               xxxxxx: begin');
    writeln('                         riii;.....');
    writeln('                       end;');
    writeln;
    writeln('               where the listed pocedure calls riii,
etc., are');
    writeln('               (or will be) theorems which use
invariant xxxxxx');
    writeln;
    pause;
      end;
  end;



procedure helpBI(var i:longint);
 begin
  i:=0;
```

```
  writeln('               Basic Information');
  writeln;
  writeln('There are currently',actualnumrules:4,' theorems in
the system,');
  writeln(-ninst:4,' System Action Commands, and ',nparam:3,'
invariants.');
  writeln;
  writeln;
  writeln('The current value of "infinity" is ',infinity:10);
  writeln('The value may be changed to nnn, by typing "^ nnn".');
  writeln('If nnn is <= 0, the current value is printed and left
unchanged.');
  writeln;
  writeln('You may return to the command level by entering
"',escp:1,'".');
  writeln('To terminate a particular graph, type "end". To
terminate a');
  writeln('session, first terminate the graph then type
"end"(again).');
  writeln;
  slow(i);
  if i >= 0 then
    begin
  writeln('When in the command level, the system prompts with
"',prompt:1,'".');
  writeln('A valid user response is any of the invariant names');
  writeln('or commands. The value or range for invariants and
any');
  writeln('arguments for commands is expected on this same
line.');
  writeln('If not given on the same line, the system will
prompt');
  writeln('with "?".');
  writeln;
  writeln('   If a numeric invariant:');
  writeln('     Enter either "= value", ": low high", "< value",
or ');
  writeln('     "> value".');
  writeln('         (note: "<" and ">" are interpreted as "less
than or');
  writeln('          equal" and "greater than or equal",
respectively).');
  writeln;
  writeln('   If a binary invariant:  Enter "=y" or "=n".');
  writeln;
  writeln('When the given value or range does not reduce the
current');
  writeln('range the system will automatically return to the
command level.');
  pause;
    end;
end;
```

```
procedure helpIN(var i:longint);
 begin
   i:=0;
  writeln('              Invariant Names and Meanings.');
  writeln;
  writeln('arbor  :  point arboricity number.');
  writeln('bwidth :  bandwidth.');
  writeln('circ   :  length of the longest cycle.');
  writeln('chr    :  chromatic number.');
  writeln('clique :  order of maximal clique.');
  writeln('diam   :  maximum distance between any two nodes.');
  writeln('dom    :  domination number.');
  writeln('earbor :  edge arboricity.');
  writeln('edges  :  number of edges.');
  writeln('eccov  :  minimum number of cliques to cover the
edges.');
  writeln('echr   :  edge chromatic number.');
  writeln('econn  :  edge connectivity number.');
  writeln('ecov   :  minimum number of edges to cover the
nodes.');
  writeln('eind   :  maximum number of independent edges.');
  writeln('genus  :  genus.');
  writeln('girth  :  length of the shortest cycle.');
  writeln('maxdeg :  maximum degree.');
  writeln('mindeg :  minimum degree.');
  writeln('nccov  :  minimum number of cliques to cover the
nodes.');
  writeln;
  slow(i);
  if i >= 0 then
    begin
  writeln('ncomp  :  number of components.');
  writeln('nconn  :  node connectivity number.');
  writeln('ncov   :  minimum number of nodes to cover the
edges.');
  writeln('nind   :  maximum number of independent nodes.');
  writeln('nodes  :  number of nodes.');
  writeln('radius :  radius.');
  writeln('spectr :  spectral radius.');
  writeln('thick  :  thickness.');
  writeln('xnum   :  crossing number.');
  writeln;
  writeln('bipart :  bipartite graph      (binary, yes or no).');
  writeln('compl  :  complete graph               "');
  writeln('connct :  connected graph              "');
  writeln('cycle  :  cycle graph                  "');
  writeln('forest :  collection of trees          "');
  writeln('hamil  :  hamiltonian graph            "');
  writeln('planar :  planar graph                 "');
  writeln('reg    :  regular graph                "');
  writeln('tree   :  connected,acyclic graph      "');
```

```
  writeln;
  slow(i);
  if i >= 0 then
    begin
  writeln('   Names and commands may be abbreviated by
truncating');
  writeln('   to a unique set of characters. For example the');
  writeln('   number of nodes may be specified by using any
of:');
  writeln;
  writeln('         nodes   node  nod  no');
  writeln;
  pause;
    end;
    end;
end;


procedure helpSAC(var i,iy:longint;specific,extern:boolean);
label 98;
var op:char;
    ntries,code,ix:integer;
 begin
  ix:=iy;
  i:=0;
  if not specific then
      begin
  space(6);
  writeln('                   System Action Commands.');
  writeln;
  writeln('      The following is a list of valid SA command
names. If you');
  writeln('  would like to see the help entry for a specific one,
type its');
  writeln('  name. Otherwise type:');
  writeln('      +  to scroll through all commands,');
  writeln('      -  to return to the Help Menu, or');
  writeln('      ',escp:1,'  to exit Help.');
  writeln;

  writeln('        batch      bstep       bound      date');
  writeln('        dtt        end         exclud     fstep');
  writeln('        ftrace     ftwith      help       includ');
  writeln('        list       recall      remove     rules');
  writeln('        thmtext    time        trace      tt');
  writeln('        twith      ucomm       undo       versus');
  ntries:=0;
  code:=18;
  while (ntries < 3) and (code <> 0) do
    begin
      code:=0;
      write('       >');
```

```
      readLine;
      op:=buffer[1];
      if op=semicol then
        begin
          op:='+';
          numc:=1;
        end;
      if (numc > 0) or (op=escp) then
         if (op='+') or (op='-') or (op=escp) then
            begin
              specific:=false;
              numc:=0;
              if op='+' then space(5)
                        else
                          begin
                            if op=escp then i:=0
                                       else i:=-1;
                            goto 98;
                          end;
            end
           else
             begin
               readName;
               if name <> blank then
                  begin
                    validName(-1,iy);
                    if (errcode <> 0) or (ix < ninst) or (-1
< ix) then
                       begin
                         ntries:=ntries+1;
                         code:=18;
                         if ntries < 3 then write(sysm:1,' Try
again.');
                         errcode:=0;
                       end
                     else
                       begin
                         specific:=true;
                         space(2);
                       end;
                  end;
             end;
    end;
   if code <> 0 then error(code);
   if errcode <> 0 then goto 98;
 end;
 if ((not specific) or (ix = batch)) and (i >= 0) then
      begin
  writeln('batch   : In the default interactive mode long
responses are');
  writeln('          broken up into screen size pages by the
insertion');
```

```
  writeln('          of "pauses" which require the viewer to hit
the return');
  writeln('          key in order to proceed to the next page.
These pauses');
  writeln('          are undesirable when output is routed to
other than');
  writeln('          the screen. Issuing "batch" disables the
pauses. To');
  writeln('          restore the pauses, simply issue batch
again.');
  writeln;
  writeln('               Examples:       batch       bat
ba');
      end;
  if ((not specific) or (ix = bstep)) and (i >= 0) then
      begin
  writeln;
  writeln('bstep   : Backward step. Displays up to',mxncopies:2,'
Tables,');
  writeln('          in reverse order of occurrence. After each
is displayed,');
  writeln('          the user may continue to the next by hitting
the return');
  writeln('          key or end the command by typing
"',escp:1,'".');
  writeln;
  writeln('               Examples:       bstep     bst
bs');
  writeln;
      end;
  if not specific then slow(i);
  if ((not specific) or (ix = bound)) and (i >= 0) then
      begin
  writeln('bound   : With invariant name argument(s) i1 i2 .....
The');
  writeln('          interval for each of i1 i2 ... is printed.
Usually');
  writeln('          used when a value is too large to be printed
in the');
  writeln('          normal Table display.');
  writeln;
  writeln('               Examples:     bound nodes chr      bo
sprectr');
  writeln;
      end;
 if (not specific) or (ix = tdate) then
      begin
  writeln('date    : Displays the current date and time');
  writeln;
  writeln('               Examples:     date       da');
  writeln;
      end;
```

```
  if ((not specific) or (ix = dtt)) and (i >= 0) then
      begin
  writeln('dtt     : Deletes the last temporary theorem. With
argument i,');
  writeln('          deletes the i\th temporary theorem. With
argument a,');
  writeln('          all temporary theorems are deleted.');
  writeln;
  writeln('               Examples:     dtt       dtt3     dtt 10
dtt a');
  writeln;
      end;
  if ((not specific) or (ix = endd)) and (i >= 0) then
      begin
  writeln('end     : Terminates the current graph.');
  writeln;
  writeln('               Examples:    end       en');
  writeln;
      end;
  if ((not specific) or  (ix = exclud)) and (i >= 0) then
      begin
  writeln('exclud  : With argument i, causes permanent theorem
number i to');
  writeln('          be deactivated. With argument tt i,
temporary theorem');
  writeln('          i is deactivated.');
  writeln;
  writeln('               Examples:    exclud   54     ex 10 40 3
ex tt 4');
       end;
  if (not specific) and (i = 0) then slow(i);
  if ((not specific) or (ix = fstep)) and (i >= 0) then
      begin
  writeln('fstep   : Forward step. With argument i, tables i, i+
1,... are');
  writeln('          displayed. With no argument, the lowest
possible value is used.');
  writeln;
  writeln('               Examples:  fstep  4    fs  3    fs');
  writeln;
      end;
  if ((not specific) or (ix = ftrace)) and (i >= 0) then
      begin
  writeln('ftrace  : Full text trace. See "trace".');
  writeln;
  writeln;
       if specific then ix:=trase;
      end;
  if ((not specific) or (ix = ftwith)) and (i >= 0) then
      begin
  writeln('ftwith  : Full text of theorems with i1, i2, ... . See
"twith"');
```

```
  writeln;
  writeln;
       if specific then ix:=twith;
      end;
  if ((not specific) or (ix = helpc)) and (i >= 0) then
      begin
  writeln('help    : Provides review of basic information. With
argument "comm",');
  writeln('          a System Action Command, gives just the help
information');
  writeln('          for that command.');
  writeln;
  writeln('               Examples:   help      he     he exclud
he li');
  writeln;
      end;
  if ((not specific) or (ix = includ)) and (i >= 0) then
      begin
  writeln('includ  : With argument i1 i2 ... causes theorems i1
i2 ...');
  writeln('          to be reactivated (after they have been
"excluded").');
  writeln('          An argument may also be tti (or tta) to
reactivate');
  writeln('          one (or all) of the temporary theorems.');
  writeln;
  writeln('               Examples:   includ 5 30 19        in
17 tt3 29');
      end;
  if (not specific) and (i >= 0) then slow(i);
  if ((not specific) or (ix = list)) and (i >= 0) then
      begin
  writeln('list    : Displays the current table of invariant
values.');
  writeln('          With argument a, invariants are presented
in');
  writeln('          alphabetic order.');
  writeln;
  writeln('               Examples:    list     li       li a');
  writeln;
      end;
  if ((not specific) or (ix = recall)) and (i >= 0) then
      begin
  writeln('recall  : With argument i, displays the i\th table.');
  writeln;
  writeln('               Examples:  recall 4    rec 6');
  writeln;
      end;
  if ((not specific) or (ix = remove)) and (i >= 0) then
      begin
  writeln('remove  : With argument i, deletes the last i
tables.');
```

```
  writeln;
  writeln('               Examples:   remove 3       rem 6');
  writeln;
      end;
  if ((not specific) or (ix = rules)) and (i >= 0) then
      begin
  writeln('rules   : Prints a list, in chronological order, of
theorem');
  writeln('          numbers. The theorem corresponding to a
number was');
  writeln('          invoked and altered the table which was
current at');
  writeln('          that time.');
  writeln;
  writeln('               Examples:   rules         ru');
  writeln;
      end;
  if (not specific) and (i >= 0) then slow(i);
  if ((not specific) or (ix = thmtxt)) and (i >= 0) then
      begin
  writeln('thmtxt  : With arguments i1 i2 ... , prints the text
of the');
  writeln('          theorems numbered i1 i2 ... .  Any argument
may be');
  writeln('          tti (or tta) and prints the ( or all)
temporary');
  writeln('          theorem(s) i.');
  writeln;
  writeln('               Examples:   thmtxt 3 56     thm 29 tt3
97');
  writeln;
      end;
  if ((not specific) or (ix = tymes)) and (i >= 0) then
      begin
  writeln('time    : Activates( and deactivates) an instruction
timing');
  writeln('          device. The default is no timing. When
activated');
  writeln('          the time in milliseconds of each command
issued is');
  writeln('          displayed after its execution.');
  writeln;
  writeln('               Examples:     time          ti');
  writeln;
      end;
  if ((not specific) or (ix = trase)) and (i >= 0) then
      begin
  writeln('trace   : With argument i> ( or i<) presents the
sequence of');
  writeln('          rules resulting in setting invariant i"s
lower(or');
  writeln('          upper) bound. A one line desciption of each
```

```
rule is');
  writeln('          printed. You may use "ftrace" to obtain a
more');
  writeln('          complete description.');
  writeln;
  writeln('               Examples:   trace nind<     tra
edges>');
  writeln('                           ftrace conn y   ftr conn
<');
      end;
  if (not specific) and (i >= 0) then slow(i);
  if ((not specific) or (ix = tt)) and (i >= 0) then
      begin
  writeln('tt      : Accepts a temporary theorem for inclusion in
the');
  writeln('          knowledge base. The temporary theorem
remains until');
  writeln('          either deleted by the command dtt or the
session ends.');
  writeln('          The user supplied theorem must be of the
following');
  writeln('          form:');
  writeln('               "invariant name" RELOP "arithmetic
expression"');
  writeln;
  writeln('          where RELOP may be any of');
  writeln('                    "<" (or "<=", these are treated
the same here)');
  writeln('                    ">" (or ">=",              "');
  writeln('                    "=" (or blank,             "');
  writeln('          The arithmetic expression must be in
"standard" infix');
  writeln('          form with the following restrictions: an
invariant');
  writeln('          name can occur only once and must be
different from');
  writeln('          that appearing before RELOP; the final value
of the');
  writeln('          expression must be monotonic in each
invariant; and');
  writeln('          the allowable operators are +(add), -
(subtract),');
  writeln('          /(longint divide), and *(multiply).');
  writeln;
  writeln('               Examples:  tt chr <= maxdeg/2');
  writeln('                          tt nconn econn');
  writeln('                          tt gi < ma+min');
      end;
  if (not specific) and (i >= 0) then slow(i);
  if ((not specific) or (ix = twith)) and (i >= 0) then
      begin
  writeln('twith   : With argument(s) i1 i2 ..., prints the
```

```
identification');
  writeln('          numbers of theorems which contain the
invariants i1,');
  writeln('          i2 ... .  May use "ftwith" to obtain the
numbers and');
  writeln('          the complete text of such theorems.');
  writeln;
  writeln('               Examples: twith edges maxdeg girth');
  writeln('                         tw chr ma nconn');
  writeln('                         ftwith ma mi chr');
  writeln('                         ftw reg ha');
  writeln;
      end;
  if ((not specific) or (ix = ucomm)) and (i >= 0) then
      begin
  writeln('ucomm   : prints the sequence of user given commands
(those');
  writeln('          altering the table).');
  writeln;
  writeln('               Examples:  ucomm            uc');
  writeln;
      end;
  if ((not specific) or (ix = undo)) and (i >= 0) then
      begin
  writeln('undo    : Removes the last command and its effects.
Note that');
  writeln('          undo followed by undo does nothing.');
  writeln;
  writeln('               Examples:   undo         un');
  writeln;
      end;
  if (not specific) and (i >= 0) then
                begin
                  slow(i);
                  if i = 0 then space(4);
                end;
  if ((not specific) or (ix = versus)) and (i >= 0) then
      begin
  writeln('versus  : With argument i, causes the current data
base to be');
  writeln('          compared with the i\th (the given argument)
and listed.');
  writeln;
  writeln('               Examples:  versus 3       ve 4');
      end;
  if (not specific) and (i >= 0) then
      begin
  space(6);
  writeln('        Names and commands may be abbreviated by
truncating');
  writeln('     to a unique set of characters. For example the');
  writeln('     command versus may be specified by using any
```

```
of:');
  writeln;
  writeln('           versus    versu   vers   ver   ve   v');
  writeln;
  pause;
      end;
98:end;



procedure helpp;
(***********************************************)
(*                                             *)
(*      prints allowable invariant names       *)
(*      and user options.                      *)
(*                                             *)
(***********************************************)
label 98,99;
var code,ntries,ix,i,j,noptions,nnn:longint;
    ch:char;
    specific,extern:boolean;



begin
  noptions:=6;
  noescap:=true;
  j:=0;
  i:=-1;
  specific:=false;
  extern:=false;
  ix:=0;
  if nextc <= numc then
       begin
         readName;
         if name <> blank then
               begin
                 i:=3;
                       extern:=true;
                 validName(-1,ix);
                       if (ix >= ninst) and (ix < 0) and (errcode
= 0)
                        then specific:=true
                        else errcode:=0;

               end;
       end;

98: while i <> 0 do
      begin
        while (i < 0) or (i > noptions) do
                if btch then
```

```
                              begin
                                j:=j+1;
                                if j > noptions then j:=0;
                                i:=j;
                              end
                               else
                                if not extern then
                                      begin
                                         space(2);
                                         writeln('       HELP MENU
OPTIONS');
                                         writeln;
                                         writeln('  Enter:    0 -
To Quit');
                                         writeln('            1 -
Basic Information');
                                         writeln('            2 -
Invariant Names and Meanings');
                                         writeln('            3 -
System Commands');
                                         writeln('            4 -
Adding a New Command to INGRID');
                                         writeln('            5 -
Adding a New Theorem to INGRID');
                                         writeln('            6 -
Adding a New Invariant to INGRID');
                                         write('           >');
                                         readln(ch);
                                         if ch='1' then i:=1
                                          else if ch='2' then i:=2
                                           else if ch='3' then i:=3
                                              else if ch='4' then
i:=4
                                               else if ch='5' then
i:=5
                                                else if ch='6' then
i:=6
                                                 else if (ch='0')
or (ch=escp) or (i = -3) then i:=0
                                             else
                                               begin
                                                  i:=i-1;
                                                  write(sysm:1,'
Please try again.');
                                                  if i = -3 then
writeln(' (last chance!)')
                                                         else
writeln;
                                               end;
                                       end;
                    if (i > 0) and (i <= noptions) then
                        begin
```

```
                    if not extern then space(6);
                    case i of
                             1:helpBI(i);
                             2:helpIN(i);
                             3:helpSAC(i,ix,specific,extern);
                             4:helpNC(i);
                             5:helpNT(i);
                             6:helpNI(i);
                             end;
                    if (errcode <> 0) or (extern) then goto 99;

                    if i <> 0 then goto 98;
               if specific then
                             begin
                               specific:=false;
                               i:=0;
                               write(sysm:1,'      Hit return');
                               readln;
                             end
                           else i:=-1;
                  end;
          end;
 writeln(sysm:1,' Exit HELP.');
 noescap:=false;
 errcode:=0;
99:end;

end.
```