```
unit cmmnds1;
   interface
      uses

MemTypes,QuickDraw,OSIntf,ToolIntf,PackIntf,PrintTraps,PasLibIntf
,
            globals,help,text;



       procedure deleteTheorem;
       procedure deleteTempTheorem(i:longint);
       procedure excludeRule;
       procedure resetTable;
       procedure reSetRemove;
         procedure printBound;
         procedure printRuleStack;
         procedure
printCommand(act:strng;actr:strng2;numact:real);
         procedure printHeader(copy,table:longint);
         procedure printTable(table1,table2:longint);
         procedure getAnswer(flag:longint);
         procedure printTheoremText;
         procedure theoremsWith(full:longint);
         procedure versusIth;
         procedure recallIth;
         procedure stepForward;
         procedure stepBackward;
         procedure
printUserCommands(act:strng;biaction:strng3;actr:strng2;numa:real
;id:longint);
         procedure userCommands;
         procedure printTempTheorem(i:longint);

   implementation


procedure printCommand(act:strng;actr:strng2;numact:real);
(*********************************************)
(*                                           *)
(*      print the command issued for a table.   *)
(*                                           *)
(*********************************************)
var i,j:longint;
begin
  write(actr:2);
  if act = parameter[spectr] then
     if numact < 0 then
        begin
          i:=trunk(-numact/10000000.0);
          rlb:=i;
```

```
          rhb:=(-numact-rlb*10000000.0)/1000.0;
          rlb:=rlb/1000.0;
          write(rlb:6:2,rhb:7:2);
         end
       else
         begin
           rlb:=numact;
           rlb:=rlb/1000.0;
           write(rlb:8:3,'     ');
         end
      else
        if numact < 0 then
            begin
              i:=trunk(-numact/10000.0);
              j:=trunk(-numact)-i*10000;
              write(i:5,j:6,'  ');
            end
          else
            begin
              i:=trunk(numact);
              write(i:5,'        ');
            end;
end;

procedure printHeader(copy,table:longint);
(*********************************)
(*                               *)
(*   prints header for table     *)
(*                               *)
(*********************************)
var numact:real;
    act:strng;
    actr:strng2;
    biact:strng3;
begin
  numact:=numactioncopies[copy];
  act:=actioncopies[copy];
  actr:=actionrelcopies[copy];
  biact:=biactioncopies[copy];
  if table > 0 then write('TABLE',table:3,': ');
  write(act:6);
  if biact = '   ' then printCommand(act,actr,numact)
     else
       if biact = 'act' then write('  activated    ')
                        else write(' =',biact:4,'         ');
end;

procedure resetTable;
(*********************************************)
(*                                           *)
(*      resets system to its previous status.  *)
(*                                           *)
```

```
(*      temporary theorem(s) are re-executed   *)
(*      in case their effects were lost or     *)
(*      changed.                               *)
(*                                             *)
(***********************************************)
label 99;
var k,i,z,l:longint;
begin
  z:=1;
  l:=copyptr+1;
  if x <> remove then
      begin
        moveCurToCopyI(copyptr);
        moveForward;
        if errcode = 0 then numtables:=numtables+1;
      end
    else
      begin
        if nextc <= numc then
             if buffer[nextc] in digit then
                     begin
                       readNum(z);
                       if errcode <> 0 then goto 99;
                       if (z < 1) or (z > mxncopies) then z:=1;
                     end;
        i:=(copyptr-z+mxncopies) mod mxncopies;
        if i = mxncopiesm1 then k:=0
                           else k:=i+1;
        if (not undoable[k]) and (xtracopies > 0) then
                begin
                  noescap:=false;
                  error(17);
                end
            else
              if (not undoable[k]) or (z > ncopies) or
                 ((z = ncopies) and (ncopies < numtables)) then
                   begin
                     if z > 1 then
                          writeln(sysm:1,' Cannot remove',z:3,'
tables.');
                     if ncopies < numtables then z:=ncopies-1
                                            else z:=ncopies;
                     write(sysm:1,' There');
                     if z = 0 then write(' are no copies')
                       else if z = 1 then
                                write(' is only one copy')
                          else write(' are only',z:3,' copies');
                     writeln(' available for removal.');
                     error(100);
                   end
                 else
                   begin
```

```
                    copyptr:=i;
                    savez:=z;
                    ncopies:=ncopies-z;
                    ruletop:=rtopcopy[copyptr];
                    moveCopyIToCur(copyptr);
                  end;
       end;
  if errcode = 0 then
   begin
     for k:=1 to z do
       begin
         if not trace then
                 write(sysm:1,' Table',numtables:3,' removed:
');
         numtables:=numtables-1;
         if l = 0 then l:=mxncopiesm1
                  else l:=l-1;
         i:=-idnumcopies[l];
         if i >= 0 then
            if i < 10000 then
               begin
                 if not trace then write('Theorem',i:4,'
deactivated.');
                 activerule[i]:=false;
               end
             else
               begin
                 i:=i-10000;
                 if not trace then
                 write( 'Temporary Theorem',i:3,' deactivated.');
                 tempactive[i]:=false;
                end
              else
                if not trace then printHeader(l,0);
         if not trace then writeln;
       end;
     down:=zeros;
     stack:=-1;
   end;
99: end;


procedure printTempTheorem(i:longint);
(*******************************************************)
(*                                                     *)
(*     prints the infix form of the temp. theorem      *)
(*     which starts at position z in trules.           *)
(*     it is in postfix form in TRULES.                *)
(*                                                     *)
(*******************************************************)
var top,j,loc,temp,zmax:longint;
    buffer:array[1..60] of longint;
```

```
         procedure subprint(z:longint);
(*********************************************)
(*                                           *)
(*  (subprocedure of printTempTheorem)       *)
(*     prints a single theorem               *)
(*                                           *)
(*********************************************)
var i,j,infin,ii:longint;
begin
  top:=0;
  infin:=10000;
  buffer[1]:=trules[z];
  z:=z+1;
  j:=3;
  while trules[z] < 1004 do
     begin
       x:=trules[z];
       z:=z+1;
       if x < 1000 then
          begin
            buffer[j]:=infin;
            buffer[j+1]:=x;
            buffer[j+2]:=infin;
            buffer[j+3]:=infin;
            down[top+1]:=j;
            j:=j+4;
            down[top+2]:=j-1;
            top:=top+3;
            down[top]:=0;
          end
         else
           begin
             buffer[down[top-4]]:=x;
             if x < 1002 then
                   begin
                     loc:=down[top-4];
                     temp:=-down[top];
                     if (x=1001) and (temp > 0) then
                           if buffer[temp]=1000 then
buffer[temp]:=1001
                                                 else
buffer[temp]:=1000;
                     down[top-4]:=down[top-1];
                     top:=top-3;
                     down[top]:=-loc;
                    end
                  else
                    begin
                      for i:=1 to 2 do
                        begin
                          if (down[top] <> 0) and (down[top] < x)
```

```
then
                              begin
                                buffer[down[top-2]]:
=buffer[down[top-2]]-1;
                                buffer[down[top-1]-1]:
=buffer[down[top-1]-1]+1;
                              end;
                            top:=top-3;
                           end;
                       top:=top+3;
                       down[top]:=x;
                       down[top-1]:=down[top+2];
                     end;
             end;
    end;
buffer[2]:=trules[z];
write(parameter[buffer[1]]:6);
j:=j-2;
for i:=2 to j do
        begin
          x:=buffer[i];
          if x > 9000 then
               begin
                 if x < infin then for ii:=1 to infin-x do
write('(')
                                 else for ii:=1 to x-infin do
write(')');
               end
             else
               if x > 999 then
                case x of
                    1000: write('+');
                    1001: write('-');
                    1002: write('*');
                    1003: write('/');
                    1004: write('= ');
                    1005: write('<= ');
                    1006: write('>= ');
                 end
              else
                if x > 0 then if parameter[x][4]=blk then
write(parameter[x]:3)
                  else if parameter[x][5]=blk then
write(parameter[x]:4)
                   else if parameter[x][6]=blk then
write(parameter[x]:5)
                     else write(parameter[x]:6)
                       else
                         begin
                           x:=-x;
                           if x < 10 then write(x:1)
                              else if x < 100 then write(x:2)
```

```
                                    else if x < 1000 then
write(x:3)
                                       else write(x:4);
                             end;
         end;
 end;

(*****************************************)
(*                                       *)
(*   body of printTempTheorem            *)
(*                                       *)
(*****************************************)
begin
  if all then zmax:=ntt
         else zmax:=i;
  for j:= i to zmax do
        begin
          if perror then write('         ')
            else if not trace then write(sysm:1,'TT-',j:2,'   ');
          z:=tspoint[tsp[j]];
          subprint(z);
          if not tempactive[j] then write('       (inactive)');
          if j < zmax then writeln;
        end;
  down:=zeros;
end;


procedure pTempThm;
(*****************************************)
(*                                       *)
(*   print temp. theorem(s) (in infix)   *)
(*                                       *)
(*****************************************)
begin
  op:=buffer[nextc];
  if op ='a' then
      begin
        nextc:=nextc+1;
        z:=1;
        all:=true;
        if ntt < 1 then write(sysm:1,' No Temporary Theorems to
print.')
                   else printTempTheorem(z);
        all:=false;
      end
    else
      if op in digit then
          begin
            readNum(z);
            if (z < 0) or (z > ntt) then error(13);
          end
```

```
        else error(13);
 end;


procedure printBound;
(*****************************************************)
(*                                                   *)
(*         prints individual bounds                  *)
(*                                                   *)
(*****************************************************)
var i:longint;
begin
  while (nextc <= numc) and (errcode = 0) do
    begin
      readName;
      if errcode = 0 then validName(1,i);
      if errcode = 0 then
        if (i < 1) or (i > nparam) then error(13)
        else
          begin
            write('  ',name:6,'      ');
            if i=spectr then
              if lammin=lammax then
                begin
                  rWriteUdt(lammin);
                  writeln;
                end
              else
                begin
                  if lammin < infinity then
write('(',lammin:9:3)
                                       else write('(',udt:7,'
');
                  if lammax < infinity then
writeln(lammax:9:3,')')
                                       else writeln(udt:7,')');
                end
              else
                if i in bparam then
                 if min[i]=1 then writeln('yes')
                    else
                      if max[i]=0 then writeln('no')
                                  else writeln('(not
determined)')
                else
                  if min[i]=max[i] then
                     if min[i] < infinity then writeln(min[i]:8)
                                          else writeln(udt:4)
                    else
                      begin
                        if min[i] < infinity then
write('(',min[i]:8,' ')
```

```
                                                  else
write('(',udt:5,' ');
                          if max[i] < infinity then
writeln(max[i]:8,')')
                                                  else
writeln(udt:6,')');
                        end;
          end;
       end;
 end;


procedure printRuleStack;
(*****************************************)
(*                                       *)
(*       prints the rules that have been *)
(*       used and in the order in which  *)
(*       they were used.                 *)
(*                                       *)
(*****************************************)
 var i,j,k,l,n,r:longint;
 begin
   if ruletop = 0 then error(5)
     else
       begin
         writeln;
         writeln(sysm:1,'     Rule stack(',ruletop:3,' items)');
         writeln;
         j:=0;
         i:=1;
         n:=0;
         while n < ruletop do
            begin
              j:=j+1;
              if j > rpercol then
                      begin
                        pause;
                        if errcode = 0 then
                           if traceop = escp then
                             begin
                               j:=0;
                               n:=ruletop;
                             end
                           else
                             begin
                               writeln;
                               j:=1;
                               i:=n+1;
                             end;
                      end;
              if (j > 0) and (errcode = 0) then
                begin
```

```
                    k:=i-rpercol;
                    for l:=1 to 4 do
                      begin
                        k:=k+rpercol;
                        if k <= ruletop then
                          begin
                            n:=n+1;
                            if rulename[k] > 0 then
                              begin
                                name:=parameter[rulename[k]];
                                if rulestk[k]=blk5 then
                                   begin
                                     if count[k] > 0 then
                                        write('***
',name:6,'(',count[k]:3,')**')
                                      else write('***   ',name:6,'
***');
                                     if change[k] = 1 then write('
')
                                                      else
write('>');
                                     if change[k] > 0 then
write('<')
                                                      else write('
');
                                   end
                                 else
                                   begin
                                     if change[k]=1 then cart:=lte
                                                    else cart:=gte;
write(k:3,'-',rulestk[k]:5,'(',name:6,') ',cart:1);
                                   end;
                              end
                            else
                              begin
                                r:=-rulename[k];
                                if r >= 10000 then
                                  begin
                                    r:=r-10000;
                                    if count[k] > 0 then write('***
TT-',r:2)
                                                    else write('***
TT-',r:2);
                                  end
                                else
                                  if count[k] > 0 then write('*
Thm-',r:3)
                                                  else write('***
Thm-',r:3);
                               if count[k] > 0 then write('
(',count[k]:3,') ***')
```

```
                                                  else write('   ***
');
                              end;
                           end;
                 if (l <> 4) and (errcode = 0) then write(' ');
              end;
           if errcode = 0 then
               begin
                 writeln;
                 i:=i+1;
               end;
        end;
        end;
     end;
    stars;
    page:=j;
 end;

procedure printTable(table1,table2:longint);
(*************************************************)
(*                                               *)
(*    LIST command.                              *)
(*                                               *)
(*       prints the bounds of all invariants     *)
(*       as well as their values at the pre-     *)
(*       vious execution of this command(if      *)
(*       different).                             *)
(*                                               *)
(*************************************************)
label 99;
var i,j,y,hb,pcopy,bcopy,linecnt:longint;
    initbkup,prnt:boolean;
 begin
   initbkup:=true;
   page:=20;
   if (table2 <= numtables-mxncopies) or
      ((table2 <= 0) and (numtables =ncopies+xtracopies)) then
initBackup
     else
       begin
         bcopy:=(copyptr-numtables+table2+mxncopies) mod
mxncopies;
         moveCopyIToBk(bcopy);
         initbkup:=false;
       end;
   pcopy:=(copyptr-numtables+table1+mxncopies) mod mxncopies;
   if table1 <> primary then
      begin
        primary:=table1;
        moveCopyIToCur(pcopy);
      end;
   if (nextc <= numc) and (x = list) then nextChar(op)
```

```
                                      else op:=blk;
   if errcode <> 0 then goto 99;
   if op='a' then sdisp:=alphadisp
             else sdisp:=dispord;
   writeln;
   write('**********');
   stars;
   if numtables = 0 then
            writeln('*                          BASE TABLE
*')
    else
      begin
        write('* ');
        printHeader(pcopy,table1);
        if initbkup then writeln('    -vs-     [ BASE TABLE ]
*')
         else
          begin
           write('  -vs- [ ');
           printHeader(bcopy,table2);
           writeln('] *');
          end;
       end;
   write('**********');
   stars;
   hb:=nparam div 2;
   linecnt:=0;
   for i:= 1 to hb do
       begin
         linecnt:=linecnt+1;
         if linecnt > page then
               begin
                 linecnt:=1;
                 pause;
                 if errcode = 0 then goto 99;
                 write('**********');
                 stars;
               end;
          for j:= 0 to 1 do
               begin
                 if j = 0 then write('*');
                 y:=sdisp[i+j*hb];
                 if (y <= nparam) and (y <> spectr) then
                    begin
                      if spec[y]>0 then write('(',spec[y]:2,')')
                                   else write('    ');
                      write(parameter[y]:6,'   ');
                      if y in bparam then
                            begin
                              if min[y]=1 then write('   yes
')
                                    else
```

```
                                        if max[y]=0 then write('
no    ')
                                                    else write('
undet   ');
                                  if (ssmin[y]<min[y]) or (ssmax[y]>
max[y]) then
                                                write('    [undet]
')
                                           else write('
');
                                    if j = 1 then writeln(' *');
                                   end
                                else
                                  begin
                                    if min[y]=max[y] then
                                         begin
                                           wrtudf(min[y]);
                                           write('     ');
                                         end
                                       else
                                         begin
                                           write('(');
                                           iWriteUdt(min[y],4);
                                           iWriteUdt(max[y],5);
                                           write(')');
                                          end;
                                    if (ssmin[y] < min[y]) or
(ssmax[y] > max[y])
                                           then
                                             begin
                                               write('[');
                                               iWriteUdt(ssmin[y],4);
                                               iWriteUdt(ssmax[y],5);
                                               write(']  ');
                                             end
                                           else write('             ');
                                     if j = 1 then writeln('*');
                                    end;
                           end;
                      end;
          end;
   if (numtables > ncopies) and (x = list) then prnt:=true
                                           else prnt:=false;
   if spec[spectr]>0 then write('*(',spec[spectr]:2,')')
                        else write('*    ');
   write(parameter[spectr]:6,'   ');
   if lammin=lammax then
                  begin
                    write('   ');
                    rWriteUdt(lammin);
                    write('        ');
                  end
```

```
                else
                  begin
                    write('(');
                    rWriteUdt(lammin);
                    rWriteUdt(lammax);
                    write(')');
                  end;
   if(sslammin < lammin) or (sslammax > lammax) then
                  begin
                    write('[');
                    rWriteUdt(sslammin);
                    rWriteUdt(sslammax);
                    write(']');
                  end
                 else
                   write('                    ');
   if prnt then writeln('   {No. backups =',ncopies-1:2,'} *')
           else writeln('                      *');
   write('**********');
   stars;
 99: end;

procedure getAnswer(flag:longint);
(*********************************)
(*                               *)
(*                               *)
(*********************************)
var op:char;
begin
  write(sysm:1,' Type "y", else return. ?');
  read(op);
  if op <> blk then
          begin
            readln;
            if op=yes then
                if flag=0 then printTable(numtables,numtables-1)
                          else printRuleStack
               else
                 if (op='K') or (op = escp) then
                      begin
                        if op='K' then
                            begin
                              trmax:=0;
                              ntemptt:=0;
                              ntt:=0;
                            end;
                        resetTable;
                        if not noescap then error(100);
                      end;
            end;
   end;
```

```
procedure printTheoremText;
(*********************************)
(*                               *)
(*  prints the text of theorems  *)
(*                               *)
(*********************************)
label 99;
var  spfix,i,i1,i2,page,is:longint;
begin
  spfix:=pfix;
  pfix:=1;
  page:=0;
  is:=0;
  if nextc > numc then
     begin
       write(sysm:1,' Enter Theorem number--?');
       readLine;
       writeln(sysm:1);
     end;
  while nextc <= numc do
      begin
        i1:=nextc;
        while (i1 <= numc) and (buffer[i1] = blk) do i1:=i1+1;
        if i1 <= numc then
          if buffer[i1] = 't' then
                begin
                  i1:=i1+1;
                  if buffer[i1] ='t' then i1:=i1+1;
                  while (i1 < numc) and (buffer[i1] = blk) do i1:
=i1+1;
                  nextc:=i1;
                  pTempThm;
                  writeln;
                end
           else
            begin
              readNum(i1);
              if errcode <> 0 then goto 99;
              if chc='-' then
                      begin
                        readNum(i2);
                        if errcode <> 0 then goto 99;
                        if (i2 > actualnumrules) and (i1 <=
actualnumrules) then
                                    begin
                                      is:=i2;
                                      i2:=actualnumrules;
                                    end;
                       end
                     else i2:=i1;
              if (i1 < 1) or (i2 > actualnumrules) or (i1 > i2)
then
```

```
                begin
                  write(sysm:1,' Invalid rule number(s)
',i1:3);
                  if i1 <> i2 then writeln('-',i2:3)
                              else writeln;
                end
               else
                 for i:=i1 to i2 do
                     begin
                       if page+lsinthm[i] > 20 then
                           begin
                             pause;
                             if errcode <> 0 then goto 99;
                             page:=lsinthm[i];
                           end
                         else page:=page+lsinthm[i];
                       if perror then write('        ')
                                 else write(i:4,':   ');
                       ruletext(i,0);
                       writeln;
                     end;
            end;
      end;
    pfix:=spfix;
    if is > 0 then
       begin
         writeln;
         writeln(sysm:1,' There are only ',i2:3,' theorems.
Not',is:4,'.');
       end;
99: end;

procedure theoremsWith(full:longint);
(*********************************************************)
(*                                                       *)
(*   print theorems with given invariants.               *)
(*   If full=1, then the full text, else just the numbers.*)
(*                                                       *)
(*********************************************************)
label 99;
var i,j,k,page,nitems:longint;
    a:array[1..15] of longint;
    keys:iset;
begin
  keys:=[];
  page:=0;
  j:=0;
  nitems:=0;
  while nextc <= numc do
     begin
       readName;
       if errcode = 0 then validName(1,i);
```

```
     if errcode <> 0 then goto 99;
     if (i < 1) or (i > nparam) then
         begin
           error(13);
           goto 99;
         end;
     keys:=keys+[i];
     j:=j+1;
     a[j]:=i;
   end;
k:=0;
if btch then
  begin
    writeln;
    write(sysm:1,' Theorems with  ');
    for i:=1 to j do write(parameter[a[i]]:7);
    writeln;
  end;
writeln;
for i:=1 to actualnumrules do
    if keys <= rulemx[i]+rulemn[i] then
      begin
        nitems:=nitems+1;
        if full = 1 then
           begin
             if page+lsinthm[i] > 20 then
                 begin
                   pause;
                   if errcode <> 0 then goto 99;
                   writeln;
                   page:=lsinthm[i];
                 end
              else page:=page+lsinthm[i];
             write(i:4,':  ');
             ruletext(i,0);
             writeln;
           end
         else
           begin
             k:=k+1;
             if k < 16 then a[k]:=i
                else
                  begin
                    for j:= 1 to 15 do write(a[j]:4);
                    writeln;
                    k:=1;
                    a[1]:=i;
                  end;
           end;
        end;
if k > 0 then
           begin
```

```
                for j:= 1 to k do write(a[j]:4);
                writeln;
              end;
  if nitems = 0 then writeln(sysm:1,' No matches found.')
     else
       begin
         writeln;
         writeln(sysm:1,' Number of matches found -',nitems:4);
       end;
99: end;

procedure versusIth;
(*************************************************)
(*                                               *)
(*     print out a table with the compnum\th table  *)
(*     as the backup table.                      *)
(*                                               *)
(*************************************************)
var low:longint;
begin
  readNum(compnum);
  if errcode = 0 then
    begin
      low:=numtables-(ncopies+xtracopies)+1;
      if (compnum >= low) and (compnum < numtables) then
           printTable(numtables,compnum)
        else
        begin
          write(sysm:1,' Given table number (',compnum:2,') is
outside');
          writeln(' of allowable range
[',low:2,numtables-1:3,'].');
        end;
     end;
end;

procedure recallIth;
(*******************************)
(*                             *)
(*       print the j\th table.  *)
(*                             *)
(*******************************)
var j,low:longint;
begin
  readNum(j);
  if errcode = 0 then
    begin
      low:=numtables-(ncopies+xtracopies)+1;
      if (low <= j) and (j <= numtables) then
        begin
          printTable(j,j-1);
          if errcode = 0 then
```

```
               begin
                 moveCopyIToCur(copyptr);
                 primary:=numtables;
               end;
        end
     else
        begin
          write(sysm:1,' Given table number (',j:2,') is
outside');
          writeln(' of allowable range
[',low:2,numtables:3,'].');
        end;
    end;
end;

procedure stepForward;
(*****************************************************)
(*                                                   *)
(*    print tables from either a supplied table      *)
(*    number or the lowest possible to the current   *)
(*    ( or until the user says to quit).             *)
(*                                                   *)
(*****************************************************)
label 99;
var i,low:longint;
    op:char;
begin
  low:=numtables-(ncopies+xtracopies)+1;
  recallnum:=low;
  i:=nextc;
  while i <= numc do
       if buffer[i] = blk then i:=i+1
             else
                begin
                  readNum(recallnum);
                  if errcode <> 0 then goto 99;
                  i:=numc+1;
                end;
  if (recallnum >= low) and (recallnum <= numtables) then
      begin
        op:=blk;
        while op <> escp do
            begin
              printTable(recallnum,recallnum-1);
              if errcode = 0 then
                begin
                  recallnum:=recallnum+1;
                  if recallnum <= numtables then pause2(op)
                                            else op:=escp;
                end;
              if errcode <> 0 then goto 99;
            end;
```

```
         moveCopyIToCur(copyptr);
         primary:=numtables;
       end
      else
        begin
          write(sysm:1,' Given table number (',recallnum:2,') is
outside');
          writeln(' of allowable range
[',low:2,numtables:3,'].');
       end;
99: end;

procedure stepBackward;
(*************************************************)
(*                                               *)
(*     print tables from the current back to     *)
(*     as far as possible ( or until the user    *)
(*     says to quit).                            *)
(*                                               *)
(*************************************************)
label 99;
var i,low:longint;
    op:char;
begin
  low:=numtables-(ncopies+xtracopies)+1;
  op:=blk;
  i:=numtables;
  while op <> escp do
       begin
         printTable(i,i-1);
         if errcode = 0 then
           begin
             i:=i-1;
             if i >= low then pause2(op)
                         else op:=escp;
           end;
         if errcode <> 0 then goto 99;
       end;
  moveCopyIToCur(copyptr);
  primary:=numtables;
99: end;

procedure printUserCommands

(act:strng;biaction:strng3;actr:strng2;numa:real;id:longint);
(*****************************************)
(*                                       *)
(*      prints for userCommands.         *)
(*                                       *)
(*****************************************)
begin
  write(act:6);
```

```
  if biaction = '   ' then printCommand(act,actr,numa)
        else
          if biaction = 'act' then write(' activated.')
                              else write(' ',biaction:3);
  if id > 0 then
              begin
                writeln;
                write('       ');
                if id > 10000 then printTempTheorem(id-10000)
                              else ruletext(id,0);
              end;
  writeln;
end;

procedure userCommands;
(*****************************************************)
(*                                                   *)
(*    prints the user commands (as many as possible)  *)
(*    which altered a table.                         *)
(*                                                   *)
(*****************************************************)
var i,low,did,j:longint;
    act:strng;
    arel:strng2;
    nact:real;
    biact: strng3;
begin
  low:=numtables-(ncopies+xtracopies)+1;
  for i:=low to numtables do
    begin
      write(i:2,':  ');
      j:=i mod mxncopies;
      act:=actioncopies[j];
      biact:=biactioncopies[j];
      arel:=actionrelcopies[j];
      nact:=numactioncopies[j];
      did:=-idnumcopies[j];
      printUserCommands(act,biact,arel,nact,did);
     end;
end;



procedure excludeRule;
(*************************************************)
(*                                               *)
(*   reads a 'rule number' and excludes it from  *)
(*   being invoked from this time until it is    *)
(*   reset by an include statement.              *)
(*                                               *)
(*************************************************)
var i:longint;
```

```
begin
  if (buffer[nextc]='t') or (buffer[nextc]='T') then
    (************************)
    (*  temporary theorem   *)
    (************************)
    begin
      nextc:=nextc+1;
      if (buffer[nextc]='t') or (buffer[nextc]='T') then nextc:
=nextc+1;
      if buffer[nextc]=blk then nextc:=nextc+1;
      if buffer[nextc]='a' then
        if ntt > 0 then
             begin
               for i:=1 to ntt do if tempactive[i] then
tempactive[i]:=false;
              writeln(sysm:1,' All Temporary Theorems have been
deactivated.');
             end
           else writeln(sysm:1,' No Temporary Theorems to
deactivate.')
          else
            if not(buffer[nextc] in digit) then error(6)
              else
                if errcode = 0 then
                begin
                  readNum(i);
                  if (i <= 0) or (i > ntt) then error(6)
                   else
                     if tempactive[i] then
                         begin
                           writeln(sysm:1,' TT-',i:2,':
deactivated.');
                           tempactive[i]:=false;
                         end
                       else writeln(sysm:1,' TT-',i:2,' is
already inactive.');
                end;
         end
       else
          (*******************************)
          (* regular theorem             *)
          (*******************************)
          while (nextc <= numc) and (errcode = 0) do
              begin
                readNum(i);
                if errcode = 0 then
                if (i < 0) or (i > cnumrules) then error(6)
                 else
                  if activerule[i] then
               begin
                 activerule[i]:=false;
                 writeln(sysm:1,' Theorem ',i:3,':
```

```
deactivated.');
               end
                  else
              writeln(sysm:1,' Theorem ',i:3,' is already
inactive.');
               end;
end;



procedure resetRemove;
(*********************************************)
(*                                           *)
(*    undoes the last set of (de)activations  *)
(*                                           *)
(*********************************************)
label 99;
var i,k,idnum:longint;
begin
  k:=copyptr;
  if savez < 0 then
    if undoable[k] then
       begin
         k:=k+1;
         numtables:=numtables+1;
       end
     else
       begin
         if xtracopies = 0 then
             begin
               writeln(sysm:1,' Cannot "undo" at this time.');
               error(100);
             end
           else
             begin
               noescap:=false;
               error(17);
             end;
          goto 99;
        end;
  copyptr:=(copyptr+savez+mxncopies) mod mxncopies;
  ruletop:=rtopcopy[copyptr];
  for i:=1 to abs(savez) do
        begin
          if savez > 0 then
                begin
                  if k = mxncopiesm1 then k:=0
                                     else k:=k+1;
                  numtables:=numtables+1;
                end
               else
                  begin
```

```
                if k = 0 then k:=mxncopiesm1
                         else k:=k-1;
                numtables:=numtables-1;
              end;
          if not trace then
            begin
              write(sysm:1,' Table',numtables:3);
              if savez > 0 then write(' restored:      ')
                           else write(' removed:       ');
            end;
          idnum:=-idnumcopies[k];
          if idnum < 0 then
              begin
                if not trace then
                       begin
                         printHeader(k,0);
                         writeln;
                       end;
              end
           else
            begin
              if idnum < 10000 then
                   begin
                     activerule[idnum]:=true;
                     if not trace then write('Theorem ',idnum:3);
                   end
                 else
                   begin
                     idnum:=idnum-10000;
                     tempactive[idnum]:=true;
                     if not trace then write('Temporary
Theorem',idnum:3);
                   end;
               if not trace then
                       if savez > 0 then writeln(' reactivated.')
                                    else writeln('
deactivated.');
            end;
        end;
  if savez < 0 then numtables:=numtables-1;
  ncopies:=ncopies+savez;
  savez:=-savez;
  moveCopyIToCur(copyptr);
  primary:=numtables;
99: end;



procedure deleteTempTheorem(i:longint);
(*********************************************************)
(*                                                       *)
(*      deletes temporary theorem i                      *)
```

```
(*                                                          *)
(************************************************************)
var j,k,l,m,mm:longint;
begin
  if (i < 1) or (i > ntt) then error(12)
       else
         if i < ntt then
               begin
                 k:=tspoint[tsp[i]];
                 l:=tspoint[tsp[i+1]];
                 m:=l-k;
                 for j:=l to trmax do
                       begin
                         trules[k]:=trules[j];
                         k:=k+1;
                       end;
                 k:=tsp[i];
                 l:=tsp[i+1];
                 mm:=l-k;
                 for j:=l to ntemptt do
                        begin
                          tspoint[k]:=tspoint[j]-m;
                          k:=k+1;
                        end;
                 for j:=i+1 to ntt do
                              begin
                                tempactive[j-1]:=tempactive[j];
                                tsp[j-1]:=tsp[j]-mm;
                               end;
                 trmax:=trmax-m;
                 ntemptt:=ntemptt-mm;
                 ntt:=ntt-1;
               end
             else
               begin
                 trmax:=tspoint[tsp[ntt]]-1;
                 ntemptt:=tsp[ntt]-1;
                 ntt:=ntt-1;
               end;
end;



procedure deleteTheorem;
(*********************************************************)
(*                                                       *)
(*       removes a temporary theorem.                    *)
(*                                                       *)
(*********************************************************)
var i:longint;
begin
```

```
  if errcode = 0 then
    if ntt < 1 then writeln(sysm:1,' No Temporary Theorems to
delete.')
      else
        if nextc > numc then deleteTempTheorem(ntt)
           else
             if buffer[nextc]='a' then
                begin
                  trmax:=0;
                  ntemptt:=0;
                  ntt:=0;
                end
              else
                if buffer[nextc] in digit then
                       begin
                         readNum(i);
                         deleteTempTheorem(i);
                      end
                   else error(11);
end;


end.
```