```
unit main;
  interface
     uses
MemTypes,QuickDraw,OSIntf,ToolIntf,PackIntf,PrintTraps,PasLibIntf
,

globals,help,moretext,txt400,text,cmmnds1,pusherr,pushStack,

      ruleAtoF,rules050,rules100,rules150,rules200,rules250,rules3
00,rules350,rules400,rules450,rules500,
                 evaluate,cmmnds2,traceExec;



            procedure mainblock;


implementation
(*****************************************************************
**)
(*****************************************************************
**)
(*
*)
(*
*)
(*                        -INGRID-
*)
(*
*)
(*             INteractive GRaph Invariant Delimiter
*)
(*             --          --    -         -
*)
(*
*)
(*          A software tool for facilitating extremal
*)
(*          graph theory research and an aid to graph
*)
(*          theorists in general.
*)
(*
*)
(*          designed by:
*)
(*
*)
(*             R. Dutton and R. Brigham  (1981)
*)
```

```
(*              Computer Science Department
*)
(*              University of Central Florida
*)
(*              Orlando, Florida 32816
*)
(*
*)
(*****************************************************************
**)
(*****************************************************************
**)
(*
*)
(*
*)
(*       To add a new rule..
*)
(*
*)
(*    (1)     In the CONST declaration section, increase
*)
(*            the value of 'cnumrules' by 1.
*)
(*
*)
(*    (2)     Create a procedure named rxxx and add
*)
(*            the code for the rule, where xxx is equal
*)
(*            to 'cnumrules'(as changed above).
*)
(*
*)
(*               procedure rxxx;
*)
(*                begin
*)
(*                  if activerule[xxx] then
*)
(*                     begin
*)
(*                       rule:='xxx/ ';
*)
(*                       ---
*)
(*                       body
*)
(*                       ---
*)
(*                      end;
*)
```

```
(*                  end;
*)
(*
*)
(*
*)
(*      (3)   In procedure EXECUTERULE append, to the case
*)
(*            statement,
*)
(*                      xxx:rxxx;
*)
(*
*)
(*      (4)   In procedure EVAL, add rxxx; to each invariant
*)
(*            involved in rxxx.
*)
(*
*)
(*      (5)   In procedure RULETEXT, add another entry into
*)
(*            the case statement, to print the text of the
*)
(*            newly added theorem.
*)
(*                xxx:write('............');
*)
(*
*)
(*            if the description is too long for one line,
*)
(*               xxx: if trace = 1 then write('..short version.')
*)
(*                     else
*)
(*                       begin
*)
(*                         writeln('  long version...');
*)
(*                         if trace<>0 then write('         ');
*)
(*                         writeln('  long version cont.  ');
*)
(*                           .
*)
(*                           .
*)
(*                         write('  long version cont.  ');
*)
(*                       end;
*)
```

```
(*
*)
(*              if description took the second form above,
*)
(*              add
*)
(*                     lsinthm[xxx]:=n;
*)
(*
*)
(*                 where n lines of output will be produced
*)
(*                 in the above case.
*)
(*
*)
(*        (6)  In procedure INITVALUE, add
*)
(*                     rulemx[xxx]:=[n1,n2,..];
*)
(*                     rulemn[xxx]:=[m1,m2,..];
*)
(*                        where
*)
(*                           ni's are invariants whose maximum
*)
(*                           values are used in computing the
*)
(*                           new rule.
*)
(*                           mi's are invariants whose minimum
*)
(*                           values are used.
*)
(*                              note: these sets need not be
*)
(*                                     disjoint.
*)
(*                    also increase 'actualnumrules' by 1.
*)
(*
*)
(*       (7)   Then recompile.
*)
(*
*)
(*
*)
(*
*)
(*****************************************************************
**)
(*****************************************************************
**)
```

```
(*
*)
(*
*)
(*      To add a new invariant:
*)
(*
*)
(*           In the CONST declaration section, increase
*)
(*              the value of 'nparam' by 1.
*)
(*              Let the new name be xxxxxx (any 1-6 alphanumeric
*)
(*              characters- no imbedded blanks and,if it is of
*)
(*              length k, it must be unique from the first k
*)
(*              characters of any other name or command,
*)
(*              e.g., LIST and NODES are currently a command
*)
(*              and an invariant name,  no new name of l,li,lis,
*)
(*              list,n,no,nod,node, or nodes can be added to
*)
(*              the system.)
*)
(*
*)
(*              insert the statement
*)
(*                       xxxxxx=nnn
*)
(*              Normally nnn is the new value of nparam.
*)
(*              (but, notice that the order in which invariants
*)
(*              are listed by the LIST statement depends upon
*)
(*              the order of the numbers assigned here.
*)
(*              Renumbering may be desirable).
*)
(*
*)
(*           In procedure INITVALUE:
*)
(*              add the statement
*)
(*                parameter[xxxxxx]:='xxxxxx';
*)
```

```
(*
*)
(*              Also,if this is a binary invariant,
*)
(*              insert xxxxxx into the set assignment
*)
(*              statement
*)
(*                   bparam:=[xxxxxx,.....
*)
(*
*)
(*           In procedure EVAL:
*)
(*               add a new case of
*)
(*                  xxxxxx: begin
*)
(*                          (names of rules which use xxxxxx   )
*)
(*                          end;
*)
(*
*)
(*****************************************************************
**)
(*****************************************************************
**)
(*
*)
(*
*)
(*         Portability Considerations.
*)
(*
*)
(*               possible nonstandard features
*)
(*                 -System function:   CLOCK
*)
(*                 -System subroutines: DATE and TIME
*)
(*                 -There are a few GOTO statements, but every
*)
(*                      attempt has been made to keep them to a
*)
(*                      minimum.
*)
(*
*)
(*          To compile on the VAX, use the -J option to allow
*)
```

```
(*         long jumps.
*)
(*          i.e.       pc -J ingrid.p
*)
(*****************************************************************
**)
(*****************************************************************
**)


procedure binaryParm;
(*****************************)
(*                           *)
(*  binary invariant. next   *)
(*  input must be            *)
(*        =Y                 *)
(*  or    =N                 *)
(*                           *)
(*****************************)
var lb:longint;
begin
  if op <> eql then
          begin
            if op <> semicol then nextc:=nextc-1;
            op:=eql;
          end;
  while (op=eql) and (errcode = 0) do
        begin
          if nextc > numc then op:=yes
                  else
                    begin
                      nextChar(op);
                      if (op=blk) and (errcode = 0) then
nextChar(op);
                    end;
          if errcode = 0 then
            if op=yes then lb:=1
                else
                  if op=no then lb:=0
                      else
                        begin
                          numc:=0;
                          op:=eql;
                        end;
        end;
  if errcode = 0 then
       if min[x]=max[x] then
              begin
                write(sysm:1,name:7,': is already ');
                if min[x]=1 then writeln(' true. No action
taken.')
                            else writeln(' false. No action
```

```
taken.');
              end
              else
                begin
                  rule:=blk5;
                  if lb=1 then cart:=blk
                          else cart:=lte;
                  savesw:=true;
                  push(x);
                  if errcode = 0 then
                    begin
                      spec[x]:=numtables;
                      action:=parameter[x];
                      actionrel:='  ';
                      if op = yes then biaction:='yes'
                                  else biaction:='no ';
                      actionrel:='  ';
                      min[x]:=lb;
                      max[x]:=lb;
                      glb1:=lb;
                      ghb1:=lb;
                    end;
                end;
 end;

procedure realSpectr;
(*****************************)
(*                           *)
(*   real parmaeter, Lambda.   *)
(*                           *)
(*****************************)
var ilb,ihb:longint;
    dactionrel:strng2;
    dnumaction:real;
  begin
    readReal(rlb);
    ilb:=trunk(rlb*1000.0);
    case op of
          colon:  begin
                    readReal(rhb);
                    ihb:=trunk(rhb*1000.0);
                    dactionrel:=': ';
                    dnumaction:=-(ilb*10000000.0+ihb);
                  end;
          eql  :  begin
                    rhb:=rlb;
                    dactionrel:='= ';
                    dnumaction:=ilb;
                  end;
          lte  :  begin
                    rhb:=rlb;
                    rlb:=lammin;
```

```
                    dactionrel:='<=';
                    dnumaction:=ilb;
                  end;
          gte   :  begin
                    rhb:=lammax;
                    dactionrel:='>=';
                    dnumaction:=ilb;
                  end;
     end;
    if rlb < lammin then rlb:=lammin;
    if rhb > lammax then rhb:=lammax;
    if (rhb < rlb) or ((rlb = lammin) and (rhb = lammax)) then
              begin
                write(sysm:1,' Current range for
',name:6,':(',lammin:8:3);
                rWriteUdt(lammax);
                writeln('). No action taken.');
              end
            else
              begin
                rule:=blk5;
                if (rlb > lammin) and (rhb < lammax) then cart:
=eql
                   else if rlb > lammin then cart:=blk
                      else cart:=lte;
                savesw:=true;
                push(spectr);
                if errcode = 0 then
                   begin
                     spec[spectr]:=numtables;
                     action:=parameter[spectr];
                     actionrel:=dactionrel;
                     numaction:=dnumaction;
                     biaction:='   ';
                     lammin:=rlb;
                     lammax:=rhb;
                     grlb1:=rlb;
                     grhb1:=rhb;
                   end;
               end;
end;

procedure numericParm;
(*********************************)
(*                               *)
(*  numerical invariant. next    *)
(*  input must be either of      *)
(*      :low high                *)
(*      =value                   *)
(*      >value                   *)
(*      <value                   *)
(* ( > and < mean >= and <= )    *)
```

```
(*                                  *)
(************************************)
  var lb,hb:longint;
      dactionrel:strng2;
      dnumaction:real;
  begin
    if (op in digit) or ((op='u') and (x in
[radius,diam,girth,circ])) then
           begin
             op:=eql;
             nextc:=nextc-1;
           end;
    while not(op in [colon,eql,lte,gte]) do
           begin
             if op <> semicol then
                  begin
                    write(sysm:1,' Character read was
''',op:1,'''.');
                    writeln(' The remaining characters are
discarded.');
                    write(sysm:1,' Please enter =, <, >, or :
followed');
                    writeln(' by the approiate value or values
for');
                  end;
             write(sysm:1,name:7,' ?');
             numc:=0;
             nextChar(op);
             if (op=blk) and (errcode = 0) then nextChar(op);
            end;
    if errcode = 0 then
          if x=spectr then realSpectr
              else
                (*******************************)
                (*                             *)
                (*  longint valued invariant.  *)
                (*                             *)
                (*******************************)
                begin
                  readNum(lb);
                  if errcode = 0 then
                  case op of
                         colon:  begin
                                   readNum(hb);
                                   if errcode = 0 then
                                     begin
                                       dactionrel:=': ';
                                       dnumaction:=-(lb*10000
+hb);
                                     end;
                                 end;
                         eql  :  begin
```

```
                                    hb:=lb;
                                    dactionrel:='= ';
                                    dnumaction:=hb;
                                  end;
                           lte  :  begin
                                    hb:=lb;
                                    lb:=min[x];
                                    dactionrel:='<=';
                                    dnumaction:=hb;
                                   end;
                           gte   :  begin
                                     hb:=max[x];
                                     dactionrel:='>=';
                                     dnumaction:=lb;
                                   end;
                       end;
                   if errcode = 0 then
                     begin
                       if lb < min[x] then lb:=min[x];
                       if hb > max[x] then hb:=max[x];
                       if (hb < lb) or ((lb = min[x]) and (hb =
max[x])) then
                         begin
                           write(sysm:1,' The current range for
',name:6,' is :(',min[x]:4);
                           iWriteUdt(max[x],5);
                           writeln('). No action taken.');
                         end
                        else
                         begin
                           rule:=blk5;
                           if (lb > min[x]) and (hb < max[x])
then cart:=eql
                              else if lb > min[x] then cart:
=blk
                                  else cart:=lte;
                           savesw:=true;
                           push(x);
                           if errcode = 0 then
                              begin
                                spec[x]:=numtables;
                                action:=parameter[x];
                                biaction:='   ';
                                actionrel:=dactionrel;
                                numaction:=dnumaction;
                                min[x]:=lb;
                                max[x]:=hb;
                                glb1:=lb;
                                ghb1:=hb;
                              end;
                         end;
                     end;
```

```
              end;
end;


procedure  pinPut;
(************************************************)
(*                                              *)
(*                                              *)
(*    Inputs and parses instructions.           *)
(*                                              *)
(*                                              *)
(************************************************)
begin
  x:=-1;
  while x <> 0 do
       begin
         numc:=0;
         errcode:=0;
         name:=blank;
         while (name = blank) and (errcode = 0) do
            begin
              if not(btch) then write(prompt:1);
              op:=blk;
              numc:=0;
              readName;
              if (errcode = 0) and (op=escp) then error(10);
            end;
         if errcode = 0 then
          begin
            oldcpuClock:=cpuClock;
            nthms:=0;
            validName(0,x);
            if errcode = 0 then
              if x < 0 then
                  case x of
                      batch:  begin
                                btch:=not(btch);
                                writeln;
                              end;
                      bound:  printBound;
                      bstep:  stepBackward;
                      dtt:    deleteTheorem;
                      endd:   x:=0;
                      exclud: excludeRule;
                      fstep:  stepForward;
                      ftrace: traceIt(2);
                      ftwith: theoremsWith(1);
                      helpc:   helpp;
                      includ: includeRule;
                      list:   printTable(numtables,numtables-1);
```

```
                    recall: recallIth;
                     remove: resetTable;
                    rules:  printRuleStack;
                    tdate:  dayTime;
                    thmtxt: printTheoremText;
                    trase:  traceIt(1);
                    tt:     postfix;
                    twith:  theoremsWith(0);
                    tymes:  begin
                             timeon:=not(timeon);
                             if timeon then
                                     writeln(sysm:1,' Timing
activated.')
                                 else writeln(sysm:1,' Timing
deactivated.');
                            end;
                    ucomm:  userCommands;
                    undo:   resetRemove;
                    versus: versusIth;
                     end
               else
               (***********************************)
               (*                                 *)
               (*     A valid invariant name has  *)
               (*     been found.                 *)
               (*                                 *)
               (***********************************)
                        begin
                          if op=blk then nextChar(op);
                    if errcode = 0 then
                      begin
                         if x in bparam then binaryParm
                                        else numericParm;
                       if errcode = 0 then eval;
                       tnthms:=tnthms+nthms;
                      end;
            end;
           if timeon then
             begin
                 currentClock:=cpuClock;
                 rz:=(currentClock-oldcpuClock)/perSecond;
                 if rz >= 60 then
                      begin
                          i:=trunc(rz/60);
                           if i = 1 then write(sysm:1,i:4,'
minute and ')
                                    else write(sysm:1,i:4,'
minutes and ');
                           rz:=rz-i*60;
                      end;
                 write(rz:5:2,' secs.');
                 if nthms > 0 then writeln('   ',nthms:6,'
```

```
theorems executed.')
                                    else writeln;
             end;
        end;
   end;
end;



procedure startup;
(*****************************************************************
*)
(*
*)
(*        initialization procedure to set items which
*)
(*        do not need to be reset with each reinitialization.
*)
(*
*)
(*****************************************************************
*)
var i:integer;




procedure initMxMn001to200;
(*********************************)
(*                               *)
(*   initializes the first 200   *)
(*   elements of rulemx and      *)
(*   rulemn arrays.              *)
(*                               *)
(*********************************)
begin
rulemx[1]:=[nodes,nconn];
    rulemn[1]:=[edges];
rulemx[2]:=[clique,ncov];
    rulemn[2]:=[chr];
rulemx[3]:=[nodes,spectr];
    rulemn[3]:=[edges];
rulemx[4]:=[edges,ncov];
    rulemn[4]:=[spectr];
rulemx[5]:=[nodes,clique];
    rulemn[5]:=[edges];
rulemx[6]:=[maxdeg];
    rulemn[6]:=[spectr];
rulemx[7]:=[nodes,diam];
    rulemn[7]:=[mindeg,nconn,diam];
rulemx[8]:=[nodes];
```

```
    rulemn[8]:=[maxdeg,mindeg,ncomp];
rulemx[9]:=[nodes];
    rulemn[9]:=[eccov];
rulemx[10]:=[radius];
    rulemn[10]:=[diam];
rulemx[11]:=[nodes];
    rulemn[11]:=[eind];
rulemx[12]:=[maxdeg,eind];
    rulemn[12]:=[nodes];
rulemx[13]:=[girth,diam,reg];
    rulemn[13]:=[];
rulemx[14]:=[nodes,chr,spectr];
    rulemn[14]:=[];
rulemx[15]:=[edges,mindeg,ncomp];
    rulemn[15]:=[mindeg,girth,nodes];
rulemx[16]:=[nconn,econn];
    rulemn[16]:=[nconn,econn];
rulemx[17]:=[clique,eccov];
    rulemn[17]:=[edges];
rulemx[18]:=[genus];
    rulemn[18]:=[chr];
rulemx[19]:=[clique,nind,ncov];
    rulemn[19]:=[edges,maxdeg];
rulemx[20]:=[maxdeg,chr,nind,eind,nccov,ncov,girth,circ];
    rulemn[20]:=[compl,nccov,ncov,echr,girth,circ,nodes];
rulemx[21]:=[nodes];
    rulemn[21]:=[genus];
rulemx[22]:=[maxdeg,eind,echr];
    rulemn[22]:=[edges,maxdeg];
rulemx[23]:=[eind,ecov];
    rulemn[23]:=[eccov];
rulemx[24]:=[nodes,girth];
    rulemn[24]:=[mindeg,girth];
rulemx[25]:=[ecov];
    rulemn[25]:=[nodes];
rulemx[26]:=[nodes];
    rulemn[26]:=[edges,dom];
rulemx[27]:=[nodes,maxdeg];
    rulemn[27]:=[ecov];
rulemx[28]:=[nodes,diam];
    rulemn[28]:=[maxdeg,diam,nconn];
rulemx[29]:=[edges];
    rulemn[29]:=[clique,eccov];
rulemx[30]:=[nodes,edges];
    rulemn[30]:=[nodes,edges,radius,connct];
rulemx[31]:=[nodes];
    rulemn[31]:=[chr,nccov];
rulemx[32]:=[chr,nccov];
    rulemn[32]:=[nodes];
rulemx[33]:=[nodes];
    rulemn[33]:=[edges,dom];
rulemx[34]:=[diam,tree];
```

```
    rulemn[34]:=[nconn,girth];
rulemx[35]:=[nodes,clique,nind,plnar];
    rulemn[35]:=[plnar,nodes,ncov];
rulemx[36]:=[nodes,edges,maxdeg,eind,ncov,ecov,bwidth,plnar];
    rulemn[36]:=[plnar];
rulemx[37]:=[nodes];
    rulemn[37]:=[edges,ncomp];
rulemx[38]:=[maxdeg,dom];
    rulemn[38]:=[nodes];
rulemx[39]:=[clique,girth];
    rulemn[39]:=rulemx[39];
rulemx[40]:=[nodes,mindeg,nind,nccov,eccov,compl];
    rulemn[40]:=rulemx[40]+[diam];
rulemx[41]:=[chr,bipart];
    rulemn[41]:=rulemx[41];
rulemx[42]:=[nodes,maxdeg,radius];
    rulemn[42]:=rulemx[42];
rulemx[43]:=[connct,forest,tree];
    rulemn[43]:=rulemx[43];
rulemx[44]:=[nodes,radius,diam,nconn,ncomp,connct];
    rulemn[44]:=rulemx[44]-[nodes];
rulemx[45]:=[maxdeg,mindeg,connct,cycle];
    rulemn[45]:=rulemx[45];
rulemx[46]:=[maxdeg,mindeg,reg];
    rulemn[46]:=rulemx[46];
rulemx[47]:=[genus,thick,plnar];
    rulemn[47]:=rulemx[47];
rulemx[48]:=[forest,plnar];
    rulemn[48]:=[forest,chr,mindeg];
rulemx[49]:
=[nodes,edges,chr,clique,nind,eind,nccov,ncov,ecov,radius,

nconn,echr,girth,circ,xnum,arbor,bwidth,forest,cycle,reg,plnar];
    rulemn[49]:=rulemx[49];
rulemx[50]:=[nodes,edges,girth,circ,ncomp,arbor,forest];
    rulemn[50]:=rulemx[50];
rulemx[51]:=[arbor];
    rulemn[51]:=[chr];
rulemx[52]:=[nodes,maxdeg,clique,spectr,compl,cycle];
    rulemn[52]:=[arbor,connct,nodes,maxdeg,reg];
rulemx[53]:=[nodes,chr];
    rulemn[53]:=[girth,arbor];
rulemx[54]:=[plnar];
    rulemn[54]:=[mindeg,clique,xnum,arbor,plnar];
rulemx[55]:=[edges];
    rulemn[55]:=[nodes,maxdeg,mindeg];
rulemx[56]:=[nodes,maxdeg,mindeg];
    rulemn[56]:=[edges];
rulemx[57]:=[nodes,mindeg,reg];
    rulemn[57]:=[nodes,mindeg,reg];
rulemx[58]:=[nodes,clique];
    rulemn[58]:=[spectr];
```

```
rulemx[59]:=[nodes];
    rulemn[59]:=[nodes,xnum];
rulemx[60]:=[nodes,genus,girth];
    rulemn[60]:=[girth,nconn,mindeg,edges,ncomp];
rulemx[61]:=[nind,ncov,genus];
    rulemn[61]:=[ncov,nind,eccov];
rulemx[62]:=[nodes,nconn];
    rulemn[62]:=[mindeg,nind];
rulemx[63]:=[nodes,diam];
    rulemn[63]:=[connct,edges];
rulemx[64]:=[nind,nconn,hamil];
    rulemn[64]:=[hamil,nodes,nconn,nind];
rulemx[65]:=[nodes,hamil];
    rulemn[65]:=[hamil,edges];
rulemx[66]:=[hamil];
    rulemn[66]:=[hamil,nconn,plnar];
rulemx[67]:=[nodes,maxdeg,chr,compl,cycle];
    rulemn[67]:=[maxdeg,nodes,cycle,connct,chr];
rulemx[68]:=[nodes,echr,compl,reg];
    rulemn[68]:=[nodes,compl,echr];
rulemx[69]:=[edges,chr];
    rulemn[69]:=[spectr];
rulemx[70]:=[chr,genus,girth];
    rulemn[70]:=[genus,girth];
rulemx[71]:=[nodes,edges,maxdeg,girth];
    rulemn[71]:=[girth,mindeg,circ,ncomp];
rulemx[72]:=[nodes,circ,hamil];
    rulemn[72]:=[hamil,circ,nodes];
rulemx[73]:=[nodes,eind,ncov,nconn,arbor,hamil];
    rulemn[73]:=[hamil,nodes,nind,ecov,nccov];
rulemx[74]:=[nodes,nind,bwidth];
    rulemn[74]:=[];
rulemx[75]:=[edges,nind];
    rulemn[75]:=[nodes];
rulemx[76]:=[nodes,nccov];
    rulemn[76]:=[eccov];
rulemx[77]:=[nodes,mindeg,hamil];
    rulemn[77]:=[hamil,edges,mindeg,nodes];
rulemx[78]:=[nodes,circ];
    rulemn[78]:=[nodes,edges,girth];
rulemx[79]:=[nind,eind,radius];
    rulemn[79]:=[girth,circ,forest];
rulemx[80]:=[girth,compl];
    rulemn[80]:=[nodes,girth];
rulemx[81]:=[nodes,mindeg,reg];
    rulemn[81]:=[reg,clique,nind];
rulemx[82]:=[nodes,econn];
    rulemn[82]:=[mindeg];
rulemx[83]:=[genus];
    rulemn[83]:=[genus,arbor];
rulemx[84]:=[clique,genus];
    rulemn[84]:=[genus,arbor];
```

```
rulemx[85]:=[clique,genus];
    rulemn[85]:=[clique,chr];
rulemx[86]:=[nodes,maxdeg,mindeg,eind];
    rulemn[86]:=[nodes,mindeg,maxdeg];
rulemx[87]:=[genus];
    rulemn[87]:=[genus,mindeg];
rulemx[88]:=[clique,genus];
    rulemn[88]:=[genus,econn];
rulemx[89]:=[genus,girth];
    rulemn[89]:=[genus,girth,econn];
rulemx[90]:=[girth,genus];
    rulemn[90]:=[girth,econn];
rulemx[91]:=[nodes];
    rulemn[91]:=[mindeg,girth];
rulemx[92]:=[nodes,nconn,circ,hamil];
    rulemn[92]:=[mindeg,nconn];
rulemx[93]:=[nodes,edges,mindeg,diam];
    rulemn[93]:=[nodes,mindeg,diam];
rulemx[94]:=[nodes,edges,diam,nconn];
    rulemn[94]:=[nodes,diam,nconn];
rulemx[95]:=[nodes,edges,diam,econn];
    rulemn[95]:=[nodes,econn,diam];
rulemx[96]:=[nodes,girth];
    rulemn[96]:=[girth,arbor];
rulemx[97]:=[clique];
    rulemn[97]:=[nodes,chr];
rulemx[98]:=[genus];
    rulemn[98]:=[chr,girth];
rulemx[99]:=[diam,econn];
    rulemn[99]:=[diam,mindeg];
rulemx[100]:=[nind,eind,ncov];
    rulemn[100]:=[ncov,nind,eccov];
rulemx[101]:=[nodes,connct];
    rulemn[101]:=[ncov,ecov];
rulemx[102]:=[bwidth];
    rulemn[102]:=[echr];
rulemx[103]:=[clique,circ];
    rulemn[103]:=[mindeg];
rulemx[104]:=[clique,circ];
    rulemn[104]:=[chr];
rulemx[105]:=[clique,circ];
    rulemn[105]:=[chr];
rulemx[106]:=[chr,nind,ncov];
    rulemn[106]:=[edges];
rulemx[107]:=[maxdeg];
    rulemn[107]:=[mindeg];
rulemx[108]:=[nodes,nind];
    rulemn[108]:=[clique,nccov];
rulemx[109]:=[eind];
    rulemn[109]:=[ncov];
rulemx[110]:=[circ];
    rulemn[110]:=[girth,mindeg];
```

```
rulemx[111]:=[nind,connct];
    rulemn[111]:=[diam,connct];
rulemx[112]:=[nodes,nind,hamil];
    rulemn[112]:=[mindeg,hamil,connct];
rulemx[113]:=[edges];
    rulemn[113]:=[mindeg,clique,nind];
rulemx[114]:=[edges];
    rulemn[114]:=[clique,ncov];
rulemx[115]:=[edges,ncomp];
    rulemn[115]:=[nodes,chr];
rulemx[116]:=[nodes,bipart];
    rulemn[116]:=[genus];
rulemx[117]:=[nodes,nconn];
    rulemn[117]:=[mindeg,compl];
rulemx[118]:=[nodes,mindeg,circ];
    rulemn[118]:=[nodes,edges];
rulemx[119]:=[nodes,clique];
    rulemn[119]:=[mindeg,chr];
rulemx[120]:=[nodes,edges,chr,hamil];
    rulemn[120]:=[nodes,chr,hamil];
rulemx[121]:=[nodes,diam];
    rulemn[121]:=[chr,diam,nconn];
rulemx[122]:=[nodes,eccov];
    rulemn[122]:=[edges];
rulemx[123]:=[nodes,econn];
    rulemn[123]:=[mindeg,econn];
rulemx[124]:=[diam,bwidth,connct];
    rulemn[124]:=[nodes];
rulemx[125]:=[nodes,genus,girth];
    rulemn[125]:=[girth,genus];
rulemx[126]:=[nodes,econn];
    rulemn[126]:=[nodes,mindeg];
rulemx[127]:=[nodes,maxdeg];
    rulemn[127]:=[nodes,maxdeg,echr,hamil];
rulemx[128]:=[nodes,maxdeg,eind];
    rulemn[128]:=[nodes,edges,maxdeg,eind];
rulemx[129]:=[nodes,mindeg,diam];
    rulemn[129]:=[diam];
rulemx[130]:=[nodes,edges,maxdeg,diam];
    rulemn[130]:=[nodes,maxdeg,diam];
rulemx[131]:=[maxdeg,clique];
    rulemn[131]:=[chr];
rulemx[132]:=[nodes,maxdeg];
    rulemn[132]:=[nodes,mindeg];
rulemx[133]:=[nind,nodes,edges];
    rulemn[133]:=[nodes,edges];
rulemx[134]:=[edges,diam];
    rulemn[134]:=[nodes,radius];
rulemx[135]:=[edges,ncomp];
    rulemn[135]:=[ncov];
rulemx[136]:=[maxdeg,eind];
    rulemn[136]:=[edges,maxdeg,eind];
```

```
rulemx[137]:=[nodes,edges,maxdeg,diam];
    rulemn[137]:=[];
rulemx[138]:=[nodes,circ];
    rulemn[138]:=[nodes,edges];
rulemx[139]:=[nodes,girth,circ];
    rulemn[139]:=[edges,girth];
rulemx[140]:=[nodes,xnum];
    rulemn[140]:=[edges,xnum];
rulemx[141]:=[nodes,circ];
    rulemn[141]:=[edges];
rulemx[142]:=[nodes,edges,girth];
    rulemn[142]:=[];
rulemx[143]:=[nodes];
    rulemn[143]:=[edges,nccov];
rulemx[144]:=[edges];
    rulemn[144]:=[chr];
rulemx[145]:=[clique,nind];
    rulemn[145]:=[nodes];
rulemx[146]:=[nodes,diam,connct];
    rulemn[146]:=[diam,connct,bwidth];
rulemx[147]:=[nodes,clique];
    rulemn[147]:=[clique,maxdeg,chr];
rulemx[148]:=[maxdeg,mindeg];
    rulemn[148]:=[reg,maxdeg,mindeg,econn,echr,plnar];
rulemx[149]:=[maxdeg];
    rulemn[149]:=[maxdeg,echr,plnar];
rulemx[150]:=[maxdeg,spectr];
    rulemn[150]:=[maxdeg,echr];
rulemx[151]:=[nconn,echr];
    rulemn[151]:=[nodes,maxdeg,nconn,reg];
rulemx[152]:=[nodes,clique,nind];
    rulemn[152]:=[nodes,edges,nind];
rulemx[153]:=[maxdeg,radius];
    rulemn[153]:=[maxdeg,bwidth];
rulemx[154]:=[nodes,maxdeg,diam];
    rulemn[154]:=[];
rulemx[155]:=[maxdeg,mindeg,eind];
    rulemn[155]:=[nodes,maxdeg,mindeg];
rulemx[156]:=[mindeg,eind];
    rulemn[156]:=[nodes,mindeg];
rulemx[157]:=[maxdeg,clique];
    rulemn[157]:=[maxdeg,chr];
rulemx[158]:=[edges,genus,connct];
    rulemn[158]:=[nodes,connct,genus];
rulemx[159]:=[maxdeg,mindeg,nind];
    rulemn[159]:=[nodes];
rulemx[160]:=[nodes,maxdeg,mindeg,clique,nind,connct,cycle];
    rulemn[160]:=[nodes,maxdeg,connct,cycle];
rulemx[161]:=[nodes,mindeg,reg,hamil,genus];
    rulemn[161]:=[nodes,mindeg,genus,girth];
rulemx[162]:=[nodes,hamil];
    rulemn[162]:=[nodes,hamil,nconn,mindeg,reg];
```

```
rulemx[163]:=[nodes,hamil];
    rulemn[163]:=[mindeg,reg,nconn];
rulemx[164]:=[edges];
    rulemn[164]:=[spectr,girth];
rulemx[165]:=[spectr];
    rulemn[165]:=[maxdeg];
rulemx[166]:=[edges,diam];
    rulemn[166]:=[nodes,diam,nconn];
rulemx[167]:=[maxdeg];
    rulemn[167]:=[chr,girth];
rulemx[168]:=[maxdeg];
    rulemn[168]:=[chr,girth];
rulemx[169]:=[edges,nind];
    rulemn[169]:=[nodes];
rulemx[170]:=[edges,nind,compl,connct];
    rulemn[170]:=[nodes,connct];
rulemx[171]:=[genus];
    rulemn[171]:=[mindeg,genus,girth];
rulemx[172]:=[ncov];
    rulemn[172]:=[diam,connct];
rulemx[173]:=[maxdeg,clique,nind];
    rulemn[173]:=[nodes];
rulemx[174]:=[maxdeg,mindeg,clique,nind];
    rulemn[174]:=[nodes];
rulemx[175]:=[nodes,bwidth];
    rulemn[175]:=[edges,bwidth];
rulemx[176]:=[clique,bwidth];
    rulemn[176]:=[mindeg,clique];
rulemx[177]:=[nodes,bwidth,tree];
    rulemn[177]:=[];
rulemx[178]:=[nodes,nind];
    rulemn[178]:=[mindeg,nccov];
rulemx[179]:=[nodes,nind];
    rulemn[179]:=[edges,nind,dom];
rulemx[180]:=[nodes,maxdeg];
    rulemn[180]:=[nodes,maxdeg,chr,reg];
rulemx[181]:=[maxdeg,nind];
    rulemn[181]:=[nodes,nccov];
rulemx[182]:=[nodes,clique,nind];
    rulemn[182]:=[mindeg];
rulemx[183]:=[clique,nind];
    rulemn[183]:=[nodes];
rulemx[184]:=[nodes,edges];
    rulemn[184]:=[nodes,mindeg,clique];
rulemx[185]:=[chr,clique,nccov];
    rulemn[185]:=[chr,clique,nccov];
rulemx[186]:=[nodes,nind];
    rulemn[186]:=[nodes,nccov];
rulemx[187]:=[nodes,maxdeg];
    rulemn[187]:=[nodes,maxdeg,clique,nind,reg];
rulemx[188]:=[nodes,girth,thick];
    rulemn[188]:=[edges,girth];
```

```
rulemx[189]:=[nodes];
    rulemn[189]:=[thick];
rulemx[190]:=[echr];
    rulemn[190]:=[thick];
rulemx[191]:=[bwidth];
    rulemn[191]:=[thick];
rulemx[192]:=[clique,nind];
    rulemn[192]:=[mindeg,diam];
rulemx[193]:=[ncov];
    rulemn[193]:=[thick];
rulemx[194]:=[thick];
    rulemn[194]:=[clique];
rulemx[195]:=[nodes,clique];
    rulemn[195]:=[edges,nind];
rulemx[196]:=[nodes,maxdeg,clique];
    rulemn[196]:=[edges,nodes,clique,nind];
rulemx[197]:=[nodes,thick];
    rulemn[197]:=[nodes,eccov,ncomp];
rulemx[198]:=[nodes,chr];
    rulemn[198]:=[ncov];
rulemx[199]:=[nodes,ncov];
    rulemn[199]:=[bwidth];
rulemx[200]:=[eind];
    rulemn[200]:=[nodes,ncov,nconn];
end;

procedure initMxMn201to400;
(*********************************)
(*                               *)
(*   initializes the next 200    *)
(*   elements of rulemx and      *)
(*   rulemn arrays.              *)
(*                               *)
(*********************************)
begin
rulemx[201]:=[eind,connct];
    rulemn[201]:=[nodes,mindeg,ncov,connct];
rulemx[202]:=[mindeg,ncov,reg];
    rulemn[202]:=[nodes,clique,reg];
rulemx[203]:=[nodes,clique];
    rulemn[203]:=[clique,ncov];
rulemx[204]:=[nodes,clique,ncov];
    rulemn[204]:=[nodes,edges,clique,ncov];
rulemx[205]:=[nodes,maxdeg,mindeg];
    rulemn[205]:=[mindeg,ecov];
rulemx[206]:=[nodes,maxdeg,mindeg];
    rulemn[206]:=[ncov];
rulemx[207]:=[nodes,maxdeg,mindeg,clique,cycle];
    rulemn[207]:=[nodes,maxdeg,clique,ncov,connct,cycle];
rulemx[208]:=[nodes,edges,compl];
    rulemn[208]:=[ncov,connct];
rulemx[209]:=[nodes,maxdeg,clique];
```

```
    rulemn[209]:=[ncov];
rulemx[210]:=[nodes,maxdeg,mindeg,clique];
    rulemn[210]:=[ncov];
rulemx[211]:=[nodes,maxdeg];
    rulemn[211]:=[nccov,ncov];
rulemx[212]:=[ecov];
    rulemn[212]:=[nccov];
rulemx[213]:=[eind];
    rulemn[213]:=[dom];
rulemx[214]:=[nind];
    rulemn[214]:=[dom];
rulemx[215]:=[dom];
    rulemn[215]:=[ncomp];
rulemx[216]:=[echr];
    rulemn[216]:=[maxdeg];
rulemx[217]:=[maxdeg];
    rulemn[217]:=[echr];
rulemx[218]:=[ncov];
    rulemn[218]:=[mindeg];
rulemx[219]:=[mindeg];
    rulemn[219]:=[econn];
rulemx[220]:=[chr];
    rulemn[220]:=[clique];
rulemx[221]:=[ncov];
    rulemn[221]:=[chr];
rulemx[222]:=[ncov];
    rulemn[222]:=[eind];
rulemx[223]:=[nccov];
    rulemn[223]:=[nind];
rulemx[224]:=[eccov];
    rulemn[224]:=[nccov];
rulemx[225]:=[diam];
    rulemn[225]:=[radius];
rulemx[226]:=[econn];
    rulemn[226]:=[nconn];
rulemx[227]:=[circ];
    rulemn[227]:=[girth];
rulemx[228]:=[circ];
    rulemn[228]:=[chr];
rulemx[229]:=[xnum];
    rulemn[229]:=[genus];
rulemx[230]:=[circ];
    rulemn[230]:=[mindeg];
rulemx[231]:=[bwidth];
    rulemn[231]:=[chr];
rulemx[232]:=[bwidth];
    rulemn[232]:=[mindeg];
rulemx[233]:=[chr,nind];
    rulemn[233]:=[nodes];
rulemx[234]:=[clique,nccov];
    rulemn[234]:=[nodes];
rulemx[235]:=[eind,echr];
```

```
    rulemn[235]:=[edges];
rulemx[236]:=[maxdeg,ncov];
    rulemn[236]:=[edges];
rulemx[237]:=[nind,bwidth];
    rulemn[237]:=[ncov];
rulemx[238]:=[spectr];
    rulemn[238]:=[chr];
rulemx[239]:=[nodes,nind,ncov];
    rulemn[239]:=[nodes,nind,ncov];
rulemx[240]:=[nodes,eind,ecov];
    rulemn[240]:=[nodes,eind,ecov];
rulemx[241]:=[nodes,clique];
    rulemn[241]:=[chr];
rulemx[242]:=[nodes];
    rulemn[242]:=[chr,nccov];
rulemx[243]:=[nodes];
    rulemn[243]:=[maxdeg,dom];
rulemx[244]:=[nodes,nind];
    rulemn[244]:=[nccov];
rulemx[245]:=[nodes,genus];
    rulemn[245]:=[mindeg,eccov,ncomp];
rulemx[246]:=[nodes,genus];
    rulemn[246]:=[nodes,eccov,ncomp];
rulemx[247]:=[nodes,maxdeg];
    rulemn[247]:=[mindeg,nind];
rulemx[248]:=[maxdeg,ncov];
    rulemn[248]:=[nodes,mindeg];
rulemx[249]:=[nind,bwidth];
    rulemn[249]:=[nodes];
rulemx[250]:=[nodes,bwidth];
    rulemn[250]:=[ncov];
rulemx[251]:=[edges,girth];
    rulemn[251]:=[girth,arbor];
rulemx[252]:=[maxdeg,eind];
    rulemn[252]:=[maxdeg,nconn,girth];
rulemx[253]:=[nodes];
    rulemn[253]:=[maxdeg,nconn,girth];
rulemx[254]:=[nodes,mindeg,nind];
    rulemn[254]:=[nind,nccov];
rulemx[255]:=[edges];
    rulemn[255]:=[nodes,thick,connct];
rulemx[256]:=[maxdeg,diam];
    rulemn[256]:=[nodes,maxdeg];
rulemx[257]:=[eind,diam,nconn];
    rulemn[257]:=[diam,nconn];
rulemx[258]:=[nodes,genus];
    rulemn[258]:=[nodes,edges,mindeg,thick,nconn];
rulemx[259]:=[nodes];
    rulemn[259]:=[chr,girth];
rulemx[260]:=[edges,ncomp];
    rulemn[260]:=[nodes,genus];
rulemx[261]:=[chr,genus];
```

```
    rulemn[261]:=[chr,girth];
rulemx[262]:=[chr,genus];
    rulemn[262]:=[chr,girth];
rulemx[263]:=[radius,maxdeg];
    rulemn[263]:=[nodes];
rulemx[264]:=[diam,dom];
    rulemn[264]:=[connct,diam,dom];
rulemx[265]:=[diam,maxdeg,mindeg];
    rulemn[265]:=[nodes,maxdeg,connct];
rulemx[266]:=[maxdeg,mindeg,diam];
    rulemn[266]:=[nodes];
rulemx[267]:=[mindeg,diam,radius];
    rulemn[267]:=[diam,radius];
rulemx[268]:=[forest,bwidth];
    rulemn[268]:=[girth,arbor];
rulemx[269]:=[forest,bwidth,nind];
    rulemn[269]:=[nodes,girth];
rulemx[270]:=[nodes,forest];
    rulemn[270]:=[dom,circ];
rulemx[271]:=[nodes,circ];
    rulemn[271]:=[nodes,mindeg];
rulemx[272]:=[nconn,diam];
    rulemn[272]:=[dom];
rulemx[273]:=[diam];
    rulemn[273]:=[radius,tree];
rulemx[274]:=[nodes];
    rulemn[274]:=[maxdeg,girth,hamil];
rulemx[275]:=[nind,chr];
    rulemn[275]:=[maxdeg];
rulemx[276]:=[clique,nccov,mindeg];
    rulemn[276]:=[nodes];
rulemx[277]:=[nconn,maxdeg];
    rulemn[277]:=[nodes,econn];
rulemx[278]:=[maxdeg,radius];
    rulemn[278]:=[nodes,connct];
rulemx[279]:=[nodes];
    rulemn[279]:=[maxdeg,radius,connct];
rulemx[280]:=[nodes];
    rulemn[280]:=[bwidth,ncomp,forest];
rulemx[281]:=[nodes,edges,mindeg,hamil];
    rulemn[281]:=[nodes,edges,mindeg,nconn];
rulemx[282]:=[nodes,bipart];
    rulemn[282]:=[nodes,cycle];
rulemx[283]:=[circ,maxdeg,mindeg];
    rulemn[283]:=[nodes,maxdeg,mindeg,nconn];
rulemx[284]:=[edges,maxdeg,clique,nind];
    rulemn[284]:=[nodes,connct];
rulemx[285]:=[nodes,maxdeg,nccov];
    rulemn[285]:=[nodes,eccov];
rulemx[286]:=[edges,chr];
    rulemn[286]:=[nodes,maxdeg,mindeg,chr];
rulemx[287]:=[nind];
```

```
    rulemn[287]:=[mindeg,plnar];
rulemx[288]:=[nodes,maxdeg,clique,diam,bipart];
    rulemn[288]:=[maxdeg,diam,reg];
rulemx[289]:=[nodes,hamil,tree];
    rulemn[289]:=[circ,connct,dom];
rulemx[290]:=[nodes];
    rulemn[290]:=[edges,dom,connct];
rulemx[291]:=[nodes];
    rulemn[291]:=[mindeg,dom];
rulemx[292]:=[nodes,girth,mindeg,diam,compl,reg,cycle,tree];
    rulemn[292]:=[nodes,maxdeg,girth,connct];
rulemx[293]:=[nodes,maxdeg,mindeg,diam];
    rulemn[293]:=[nodes,connct,mindeg,diam];
rulemx[294]:=[nodes];
    rulemn[294]:=[clique,eccov];
rulemx[295]:=[nodes,eind];
    rulemn[295]:=[eind,edges];
rulemx[296]:=[maxdeg,nconn];
    rulemn[296]:=[mindeg,econn];
rulemx[297]:=[nodes,hamil,genus];
    rulemn[297]:=[nconn];
rulemx[298]:=[maxdeg,plnar,bipart,nodes];
    rulemn[298]:=[mindeg,nconn];
rulemx[299]:=[maxdeg,nconn,nodes];
    rulemn[299]:=[mindeg,nconn];
rulemx[300]:=[maxdeg,nconn,plnar,nodes];
    rulemn[300]:=[mindeg,nconn];
rulemx[301]:=[maxdeg,nodes,hamil];
    rulemn[301]:=[mindeg,nconn,plnar,bipart];
rulemx[302]:=[nodes,maxdeg,hamil];
    rulemn[302]:=[mindeg,nconn,plnar];
rulemx[303]:=[maxdeg,hamil,nodes];
    rulemn[303]:=[mindeg,nconn,bipart];
rulemx[304]:=[eind,mindeg];
    rulemn[304]:=[nodes,mindeg,reg,econn];
rulemx[305]:=[maxdeg,ncomp,eind];
    rulemn[305]:=[nodes,mindeg];
rulemx[306]:=[nind,maxdeg,clique,edges];
    rulemn[306]:=[nodes];
rulemx[307]:=[maxdeg,econn,diam];
    rulemn[307]:=[nodes,mindeg,econn];
rulemx[308]:=[nind,circ];
    rulemn[308]:=[nodes,nconn,nind];
rulemx[309]:=[maxdeg,circ,hamil];
    rulemn[309]:=[mindeg,nconn,plnar,nodes];
rulemx[310]:=[maxdeg,nodes,hamil];
    rulemn[310]:=[mindeg,nconn,plnar];
rulemx[311]:=[clique,maxdeg,nodes];
    rulemn[311]:=[maxdeg,chr];
rulemx[312]:=[maxdeg,eind];
    rulemn[312]:=[mindeg,econn,nodes];
rulemx[313]:=[circ,hamil];
```

```
    rulemn[313]:=[nodes,reg,nconn,mindeg];
rulemx[314]:=[nodes,circ,hamil];
    rulemn[314]:=[mindeg,reg,nconn];
rulemx[315]:=[circ,hamil];
    rulemn[315]:=[nodes,mindeg,reg,nconn];
rulemx[316]:=[nodes,echr,maxdeg];
    rulemn[316]:=[reg,nodes,maxdeg,echr];
rulemx[317]:=[nodes,edges];
    rulemn[317]:=[nodes,mindeg,maxdeg,echr];
rulemx[318]:=[nodes,maxdeg];
    rulemn[318]:=[girth,spectr];
rulemx[319]:=[girth,edges,maxdeg];
    rulemn[319]:=[spectr];
rulemx[320]:=[nodes];
    rulemn[320]:=[connct,reg,nodes,mindeg,girth];
rulemx[321]:=[mindeg,eccov,compl];
    rulemn[321]:=[connct,reg,nodes,mindeg];
rulemx[322]:=[mindeg,eccov,compl];
    rulemn[322]:=[nodes,mindeg,connct,reg];
rulemx[323]:=[maxdeg,nind];
    rulemn[323]:=[girth,nodes];
rulemx[324]:=[mindeg,maxdeg,nind];
    rulemn[324]:=[girth,maxdeg,mindeg,nodes];
rulemx[325]:=[girth,nodes,bipart];
    rulemn[325]:=[mindeg,girth,reg,diam,connct];
rulemx[326]:=[nodes];
    rulemn[326]:=[nodes,bipart,thick];
rulemx[327]:=[maxdeg];
    rulemn[327]:=[edges,thick];
rulemx[328]:=[thick];
    rulemn[328]:=[nconn];
rulemx[329]:=[nodes,mindeg,girth];
    rulemn[329]:=[mindeg,girth,reg];
rulemx[330]:=[nodes,mindeg,hamil];
    rulemn[330]:=[edges];
rulemx[331]:=[nodes,nconn];
    rulemn[331]:=[edges,mindeg];
rulemx[332]:=[nodes,maxdeg];
    rulemn[332]:=[nodes,bwidth,tree];
rulemx[333]:=[edges];
    rulemn[333]:=[bwidth];
rulemx[334]:=[nodes];
    rulemn[334]:=[mindeg,dom,diam,girth,connct];
rulemx[335]:=[nodes,bwidth,edges];
    rulemn[335]:=[nodes,bwidth];
rulemx[336]:=[nodes,edges];
    rulemn[336]:=[bwidth];
rulemx[337]:=[nodes,girth];
    rulemn[337]:=[edges,mindeg,dom];
rulemx[338]:=[nodes,girth];
    rulemn[338]:=[mindeg,maxdeg,girth,dom,connct];
rulemx[339]:=[dom];
```

```
    rulemn[339]:=[mindeg,ncomp,girth];
rulemx[340]:=[dom];
    rulemn[340]:=[mindeg,girth];
rulemx[341]:=[dom];
    rulemn[341]:=[mindeg,maxdeg,girth];
rulemx[342]:=[nodes,girth];
    rulemx[342]:=[nodes,edges,girth];
rulemx[343]:=[nodes];
    rulemn[343]:=[nodes,edges,girth];
rulemx[344]:=[nodes,girth,forest];
    rulemn[344]:=[nodes,edges,girth,ncomp];
rulemx[345]:=[nodes,forest];
    rulemn[345]:=[edges,girth];
rulemx[346]:=[forest,genus];
    rulemn[346]:=[girth,mindeg,edges,nconn,ncomp];
rulemx[347]:=[diam,nconn,maxdeg];
    rulemn[347]:=[nodes];
rulemx[348]:=[nodes,forest];
    rulemn[348]:=[nodes,edges,girth,ncomp];
rulemx[349]:=[nodes,forest];
    rulemn[349]:=[nodes,edges,girth,ncomp];
rulemx[350]:=[nodes,forest];
    rulemn[350]:=[nodes,girth,ncomp,edges];
rulemx[351]:=[nodes,edges,girth,nconn,econn];
    rulemn[351]:=[nodes,edges,girth,mindeg,plnar];
rulemx[352]:=[nodes];
    rulemn[352]:=[edges,mindeg,girth];
rulemx[353]:=[nodes];
    rulemn[353]:=[edges,girth];
rulemx[354]:=[nodes,forest];
    rulemn[354]:=[edges,girth];
rulemx[355]:=[nodes];
    rulemn[355]:=[mindeg,maxdeg,girth];
rulemx[356]:=[eind,echr];
    rulemn[356]:=[nodes,maxdeg,reg];
rulemx[357]:=[eind];
    rulemn[357]:=[nodes,maxdeg,reg];
rulemx[358]:=[nodes,nconn,hamil];
    rulemn[358]:=[mindeg,nconn];
rulemx[359]:=[mindeg,nconn,circ];
    rulemn[359]:=[nodes,mindeg,nconn,hamil];
rulemx[360]:=[nodes,maxdeg,nconn];
    rulemn[360]:=[nodes,maxdeg,reg,nind];
rulemx[361]:=[mindeg,nind];
    rulemn[361]:=[maxdeg,mindeg,girth];
rulemx[362]:=[nind];
    rulemn[362]:=[maxdeg,mindeg,girth];
rulemx[363]:=[edges,diam,nconn];
    rulemn[363]:=[nodes,diam,nconn];
rulemx[364]:=[edges,diam,nconn];
    rulemn[364]:=[nodes,diam,nconn];
rulemx[365]:=[nodes];
```

```
    rulemn[365]:=[bipart,xnum];
rulemx[366]:=[spectr];
    rulemn[366]:=[connct,nodes];
rulemx[367]:=[nodes,girth,bipart];
    rulemn[367]:=[mindeg,reg,girth];
rulemx[368]:=[nodes];
    rulemn[368]:=[thick,bipart];
rulemx[369]:=[clique,genus];
    rulemn[369]:=[thick];
rulemx[370]:=[genus,thick];
    rulemn[370]:=[genus,thick];
rulemx[371]:=[earbor];
    rulemn[371]:=[arbor];
rulemx[372]:=[earbor];
    rulemn[372]:=[thick];
rulemx[373]:=[genus];
    rulemn[373]:=[genus,earbor];
rulemx[374]:=[genus];
    rulemn[374]:=[thick];
rulemx[375]:=[genus];
    rulemn[375]:=[connct,reg,earbor,nodes];
rulemx[376]:=[thick,nodes];
    rulemn[376]:=[genus];
rulemx[377]:=[maxdeg];
    rulemn[377]:=[earbor];
rulemx[378]:=[earbor];
    rulemn[378]:=[mindeg];
rulemx[379]:=[nodes,earbor];
    rulemn[379]:=[edges,ncomp];
rulemx[380]:=[thick];
    rulemn[380]:=[earbor];
rulemx[381]:=[nodes,econn];
    rulemn[381]:=[nodes,plnar,edges];
rulemx[382]:=[nodes,econn];
    rulemn[382]:=[edges,mindeg,plnar];
rulemx[383]:=[nodes,forest];
    rulemn[383]:=[maxdeg,ncomp,girth,circ];
rulemx[384]:=[nodes,edges];
    rulemn[384]:=[eind,ncov];
rulemx[385]:=[genus,nind,ncov];
    rulemn[385]:=[nodes,eccov];
rulemx[386]:=[dom];
    rulemn[386]:=[mindeg,girth,ncomp];
rulemx[387]:=[nodes];
    rulemn[387]:=[mindeg,girth,dom];
rulemx[388]:=[nodes];
    rulemn[388]:=[mindeg,girth,dom];
rulemx[389]:=[maxdeg,nind,clique];
    rulemn[389]:=[nodes,maxdeg];
rulemx[390]:=[nind,clique];
    rulemn[390]:=[nodes];
rulemx[391]:=[nodes,mindeg,circ];
```

```
    rulemn[391]:=[nodes,mindeg,edges];
rulemx[392]:=[nodes,girth];
    rulemn[392]:=[mindeg,girth];
rulemx[393]:=[nodes,girth,reg];
    rulemn[393]:=[mindeg,girth];
rulemx[394]:=[nodes,nind,clique];
    rulemn[394]:=[edges];
rulemx[395]:=[nodes,tree];
    rulemn[395]:=[mindeg,girth,diam,connct];
rulemx[396]:=[nodes];
    rulemn[396]:=[mindeg,girth,edges,forest];
rulemx[397]:=[genus];
    rulemn[397]:=[girth,arbor,genus];
rulemx[398]:=[nodes];
    rulemn[398]:=[girth,mindeg,ncomp,maxdeg];
rulemx[399]:=[maxdeg,diam,nconn];
    rulemn[399]:=[nodes];
rulemx[400]:=[nodes,edges,maxdeg,diam];
    rulemn[400]:=[nodes,maxdeg,diam];
end;



       procedure sortDisplay(first,last:longint);

(*************************************************************)
       (*
*)
       (*        sorts invariant names for an alphabetic output
*)
       (*        listing when LIST A is used.
*)
       (*        a simple bubble sort is used.
*)
       (*
*)

(*************************************************************)
       var sj,high,i,j:longint;
       begin
         high:=last+1;
         for i:=first to last-1 do
           begin
             high:=high-1;;
             z:=alphadisp[first];
             sj:=first;
             for j:=first+1 to high do
                if parameter[z] < parameter[alphadisp[j]] then
                             begin
                               z:=alphadisp[j];
                               sj:=j;
```

```
                              end;
              alphadisp[sj]:=alphadisp[high];
              alphadisp[high]:=z;
            end;
        end;




begin
     weekdays[1]:='Sunday,   ';
     weekdays[2]:='Monday,   ';
     weekdays[3]:='Tuesday,  ';
     weekdays[4]:='Wednesday,';
     weekdays[5]:='Thursday, ';
     weekdays[6]:='Friday,   ';
     weekdays[7]:='Saturday, ';
     months[1]:='Jan.,';
     months[2]:='Feb.,';
     months[3]:='Mar.,';
     months[4]:='Apr.,';
     months[5]:='May, ';
     months[6]:='June,';
     months[7]:='July,';
     months[8]:='Aug.,';
     months[9]:='Spt.,';
     months[10]:='Oct.,';
     months[11]:='Nov.,';
     months[12]:='Dec.,';
  space(2);
  stars;
  stars;
  writeln('***
***');
  writeln('***                        I N G R I D
***');
  writeln('***
***');
  writeln('***         INteractive GRaph Invariant Delimiter.
***');
  writeln('***
***');
  writeln('***      This program is intended to be a software
tool        ***');
  writeln('***      for extremal graph theory research.
***');
  writeln('***
***');
  writeln('***      Designed by:
```

```
***');
  writeln('***
***');
  writeln('***              R. Dutton and R. Brigham
***');
  writeln('***              Department of Computer Science and
***');
  writeln('***              Department of Mathematics
***');
  writeln('***              University of Central Florida
***');
  writeln('***              Orlando, Florida  32816
***');
  writeln('***
***');
  stars;
  stars;
  dayTime;
  timeon:=false;
  go:=true;
  perrorother:=false;
  trmax:=0;
  pfix:=0;
  ntemptt:=0;
  ntt:=0;
  infinity:=20000;
  mxncopies:=mxncopiesm1+1;
  buffer[0]:=blk;
  zip:=ord('0');
  digit:=['0'..'9'];
  letter:=['a'..'z','A'..'Z'];
         parameter[0]     :=blank;
            parameter[tdate] :='date  ';
         parameter[ucomm] :='ucomm ';
         parameter[bstep] :='bstep ';
         parameter[fstep] :='fstep ';
         parameter[versus]:='versus';
         parameter[recall]:='recall';
         parameter[batch] :='batch ';
         parameter[twith] :='twith ';
         parameter[ftwith]:='ftwith';
         parameter[ftrace]:='ftrace';
         parameter[thmtxt]:='thmtxt';
          parameter[trase] :='trace ';
         parameter[undo]  :='undo  ';
         parameter[dtt]   :='dtt   ';
         parameter[tt]    :='tt    ';
         parameter[helpc]  :='help  ';
         parameter[list]  :='list  ';
         parameter[endd]  :='end   ';
         parameter[bound] :='bound ';
         parameter[rules] :='rules ';
```

```
          parameter[remove]:='remove';
          parameter[tymes] :='time  ';
          parameter[exclud]:='exclud';
          parameter[includ]:='includ';
          parameter[nodes] :='nodes ';
          parameter[edges] :='edges ';
          parameter[maxdeg]:='maxdeg';
          parameter[mindeg]:='mindeg';
          parameter[chr]   :='chr   ';
          parameter[clique]:='clique';
          parameter[ncov]  :='ncov  ';
          parameter[ecov]  :='ecov  ';
          parameter[nind]  :='nind  ';
          parameter[eind]  :='eind  ';
          parameter[nccov] :='nccov ';
          parameter[eccov] :='eccov ';
          parameter[radius]:='radius';
          parameter[diam]  :='diam  ';
          parameter[genus] :='genus ';
          parameter[nconn] :='nconn ';
          parameter[econn] :='econn ';
          parameter[echr]  :='echr  ';
          parameter[girth] :='girth ';
          parameter[circ]  :='circ  ';
          parameter[ncomp] :='ncomp ';
          parameter[xnum]  :='xnum  ';
          parameter[arbor] :='arbor ';
          parameter[earbor] :='earbor';
          parameter[dom]   :='dom   ';
          parameter[bwidth]:='bwidth';
          parameter[thick] :='thick ';
          parameter[compl] :='compl ';
          parameter[bipart]:='bipart';
          parameter[connct]:='connct';
          parameter[forest]:='forest';
          parameter[tree]  :='tree  ';
          parameter[cycle] :='cycle ';
          parameter[reg]   :='reg   ';
          parameter[hamil] :='hamil ';
          parameter[plnar]:='planar';
          parameter[spectr]:='spectr';

bparam:=[compl,bipart,connct,forest,tree,cycle,reg,hamil,plnar];
x:=0;
nbparam:=0;
for i:= 1 to nparam do
        if i in bparam then
              begin
                nbparam:=nbparam+1;
                dispord[nparam-nbparam]:=i;
              end
            else
```

```
              if i <> spectr then
                 begin
                   x:=x+1;
                   dispord[x]:=i;
                 end;
dispord[nparam]:=spectr;
alphadisp:=dispord;
sortDisplay(1,nparam-1-nbparam);
sortDisplay(nparam-nbparam,nparam-1);
for i:=1 to cnumrules do activerule[i]:=true;

initMxMn001to200;

initMxMn201to400;


rulemx[401]:=[nodes,maxdeg,nconn];
    rulemn[401]:=[edges,mindeg,plnar];
rulemx[402]:=[nodes];
    rulemn[402]:=[mindeg,nind,ncomp,bwidth];
rulemx[403]:=[nodes,eind,nconn];
    rulemn[403]:=[nodes,mindeg,nconn,reg];
rulemx[404]:=[nodes,econn];
    rulemn[404]:=[mindeg,maxdeg,econn,nconn];
rulemx[405]:=[econn,diam];
    rulemn[405]:=[mindeg,nconn,diam,dom];
rulemx[406]:=[nodes,edges,econn];
    rulemn[406]:=[nodes,edges,mindeg];
rulemx[407]:=[nodes];
    rulemn[407]:=[maxdeg,mindeg,dom];
rulemx[408]:=[edges,diam];
    rulemn[408]:=[nodes,maxdeg];
rulemx[409]:=[nodes,clique];
    rulemn[409]:=[edges,hamil];
rulemx[410]:=[spectr];
    rulemn[410]:=[mindeg];
rulemx[411]:=[nodes,mindeg,diam];
    rulemn[411]:=[nodes,mindeg,diam,connct];
rulemx[412]:=[eind,mindeg,forest];
    rulemn[412]:=[nodes,mindeg,maxdeg,girth];
rulemx[413]:=[nind];
    rulemn[413]:=[nodes,chr,edges];
rulemx[414]:=[nodes];
    rulemn[414]:=[edges,diam,nconn];
rulemx[415]:=[nodes];
    rulemn[415]:=[mindeg,dom];
rulemx[416]:=[nodes,edges,maxdeg,mindeg,echr];
    rulemn[416]:=[nodes,edges,maxdeg,mindeg,echr];
rulemx[417]:=[edges,maxdeg,nind,clique];
    rulemn[417]:=[nodes];
rulemx[418]:=[edges,maxdeg,nind,clique];
    rulemn[418]:=[nodes];
```

```
rulemx[419]:=[nodes,clique,ncov];
    rulemn[419]:=[nodes,edges,ncov];
rulemx[420]:=[nodes,clique,nind];
    rulemn[420]:=[nodes,edges,nind];
rulemx[421]:=[nodes,edges,clique,nind];
    rulemn[421]:=[clique,nodes];
rulemx[422]:=[nodes,bwidth];
    rulemn[422]:=[ncov,girth,forest];
rulemx[423]:=[nodes,chr];
    rulemn[423]:=[maxdeg,ncov];
rulemx[424]:=[nodes,maxdeg,clique,edges];
    rulemn[424]:=[nodes,ncov,connct,cycle];
rulemx[425]:=[nodes];
    rulemn[425]:=[mindeg,ncov,plnar];
rulemx[426]:=[nodes];
    rulemn[426]:=[edges,ecov];
rulemx[427]:=[nodes,mindeg];
    rulemn[427]:=[mindeg,ecov,econn,reg];
rulemx[428]:=[nodes,maxdeg,ncomp];
    rulemn[428]:=[mindeg,ecov];
rulemx[429]:=[nodes,edges,maxdeg,clique];
    rulemn[429]:=[ncov];
rulemx[430]:=[nodes,circ];
    rulemn[430]:=[nconn,ncov];
rulemx[431]:=[nodes,maxdeg];
    rulemn[431]:=[ncov,girth];
rulemx[432]:=[nodes,maxdeg];
    rulemn[432]:=[mindeg,ncov,girth];
rulemx[433]:=[nodes,echr];
    rulemn[433]:=[maxdeg,ecov,reg];
rulemx[434]:=[nodes];
    rulemn[434]:=[maxdeg,ecov,reg];
rulemx[435]:=[nodes,maxdeg,ncov,nconn];
    rulemn[435]:=[nodes,maxdeg,reg];
rulemx[436]:=[nodes,mindeg];
    rulemn[436]:=[mindeg,maxdeg,girth,ncov];
rulemx[437]:=[nodes];
    rulemn[437]:=[mindeg,maxdeg,ncov,girth];
rulemx[438]:=[edges,nind];
    rulemn[438]:=[nodes,eind];
rulemx[439]:=[nodes,edges,ecov];
    rulemn[439]:=[ncov];
rulemx[440]:=[edges,nind,ecov];
    rulemn[440]:=[nodes];
rulemx[441]:=[nodes,maxdeg,clique];
    rulemn[441]:=[ncov,maxdeg];
rulemx[442]:=[nodes,edges,clique];
    rulemn[442]:=[nodes,ncov];
rulemx[443]:=[nodes,ncov];
    rulemn[443]:=[mindeg,bwidth,ncomp];
rulemx[444]:=[nodes,nconn];
    rulemn[444]:=[nodes,mindeg,nconn,reg,ecov];
```

```
rulemx[445]:=[nodes];
    rulemn[445]:=[chr,ncov,edges];
rulemx[446]:=[nodes,edges,clique,maxdeg];
    rulemn[446]:=[ncov];
rulemx[447]:=[nodes,edges,maxdeg,clique];
    rulemn[447]:=[ncov];
rulemx[448]:=[nodes,maxdeg,mindeg,nccov];
    rulemn[448]:=[edges,nccov];
rulemx[449]:=[nodes,clique,nind];
    rulemn[449]:=[edges,mindeg];
rulemx[450]:=[nodes,mindeg,nind];
    rulemn[450]:=[nodes,mindeg,nccov];
rulemx[451]:=[nodes,mindeg,clique,nind];
    rulemn[451]:=[nodes,mindeg,clique,nind];
rulemx[452]:=[clique,nind];
    rulemn[452]:=[nodes,edges,nind];
rulemx[453]:=[nodes,clique,nccov];
    rulemn[453]:=[edges,clique,nccov];
rulemx[454]:=[nodes,nind,clique];
    rulemn[454]:=[edges,mindeg];
rulemx[455]:=[nodes,nind,clique];
    rulemn[455]:=[edges,mindeg];
rulemx[456]:=[ncov,clique];
    rulemn[456]:=[chr];
rulemx[457]:=[nodes,clique];
    rulemn[457]:=[nind,chr];
rulemx[458]:=[nodes,nind];
    rulemn[458]:=[clique,nccov];
rulemx[459]:=[];
    rulemn[459]:=[];
rulemx[460]:=[];
    rulemn[460]:=[];
actualnumrules:=458;
for i:=1 to actualnumrules do lsinthm[i]:=1;
lsinthm[453]:=3; lsinthm[454]:=2; lsinthm[455]:=2; lsinthm[452]:=
2;
lsinthm[451]:=2; lsinthm[450]:=2; lsinthm[449]:=2; lsinthm[445]:=
3;
lsinthm[444]:=5; lsinthm[442]:=2; lsinthm[437]:=6; lsinthm[441]:=
2;
lsinthm[431]:=2; lsinthm[428]:=2; lsinthm[427]:=5; lsinthm[430]:=
2;
lsinthm[432]:=2; lsinthm[436]:=4; lsinthm[426]:=2; lsinthm[424]:=
3;
lsinthm[422]:=2; lsinthm[421]:=2; lsinthm[420]:=2; lsinthm[419]:=
2;
lsinthm[416]:=2; lsinthm[414]:=5; lsinthm[413]:=2; lsinthm[412]:=
9;
lsinthm[411]:=6; lsinthm[409]:=2; lsinthm[408]:=4; lsinthm[406]:=
2;
lsinthm[403]:=5; lsinthm[400]:=6; lsinthm[399]:=2; lsinthm[396]:=
7;
```

```
lsinthm[395]:=8; lsinthm[394]:=2; lsinthm[390]:=2; lsinthm[388]:=
2;
lsinthm[387]:=3; lsinthm[385]:=2; lsinthm[383]:=2; lsinthm[382]:=
3;
lsinthm[367]:=2; lsinthm[365]:=2; lsinthm[364]:=2; lsinthm[363]:=
2;
lsinthm[362]:=4; lsinthm[361]:=3; lsinthm[355]:=6; lsinthm[354]:=
3;
lsinthm[353]:=5; lsinthm[352]:=4; lsinthm[351]:=3; lsinthm[348]:=
2;
lsinthm[349]:=2; lsinthm[350]:=2; lsinthm[346]:=2; lsinthm[345]:=
3;
lsinthm[344]:=2; lsinthm[342]:=2; lsinthm[338]:=2; lsinthm[337]:=
2;
lsinthm[336]:=2; lsinthm[335]:=2; lsinthm[334]:=2; lsinthm[330]:=
3;
lsinthm[326]:=2; lsinthm[325]:=3; lsinthm[324]:=2; lsinthm[323]:=
2;
lsinthm[322]:=3; lsinthm[321]:=3; lsinthm[319]:=3; lsinthm[318]:=
2;
lsinthm[317]:=3; lsinthm[315]:=2; lsinthm[314]:=2; lsinthm[311]:=
2;
lsinthm[307]:=2; lsinthm[304]:=4; lsinthm[301]:=2; lsinthm[293]:=
3;
lsinthm[292]:=3;
lsinthm[289]:=2; lsinthm[288]:=2; lsinthm[284]:=2; lsinthm[281]:=
2;
lsinthm[278]:=2; lsinthm[266]:=2; lsinthm[265]:=2; lsinthm[262]:=
2;
lsinthm[258]:=2; lsinthm[257]:=2; lsinthm[256]:=2; lsinthm[254]:=
3;
lsinthm[253]:=2; lsinthm[252]:=4; lsinthm[211]:=2; lsinthm[208]:=
2;
lsinthm[207]:=5; lsinthm[205]:=2; lsinthm[204]:=2; lsinthm[201]:=
2;
lsinthm[196]:=2; lsinthm[188]:=2; lsinthm[187]:=2; lsinthm[180]:=
4;
lsinthm[170]:=2; lsinthm[166]:=2; lsinthm[162]:=3; lsinthm[161]:=
3;
lsinthm[160]:=3; lsinthm[158]:=2; lsinthm[154]:=2; lsinthm[152]:=
2;
lsinthm[147]:=3; lsinthm[136]:=2; lsinthm[134]:=2; lsinthm[133]:=
3;
lsinthm[132]:=3; lsinthm[130]:=3; lsinthm[129]:=2; lsinthm[128]:=
14;
lsinthm[125]:=2; lsinthm[122]:=2; lsinthm[120]:=2; lsinthm[118]:=
2;
lsinthm[116]:=2; lsinthm[110]:=2; lsinthm[101]:=3; lsinthm[ 98]:=
3;
lsinthm[ 95]:=3; lsinthm[ 94]:=3; lsinthm[ 93]:=3; lsinthm[ 91]:=
2;
lsinthm[ 90]:=2; lsinthm[ 89]:=3; lsinthm[86]:=15; lsinthm[ 80]:=
```

```
2;
lsinthm[ 79]:=2; lsinthm[ 77]:=3; lsinthm[ 73]:=3; lsinthm[ 71]:=
2;
lsinthm[ 70]:=2; lsinthm[ 63]:=2; lsinthm[ 60]:=2; lsinthm[ 59]:=
2;
lsinthm[ 53]:=2; lsinthm[ 52]:=4; lsinthm[ 50]:=2; lsinthm[ 49]:=
7;
lsinthm[ 44]:=2; lsinthm[ 40]:=2; lsinthm[ 36]:=2; lsinthm[ 30]:=
4;
lsinthm[ 28]:=2; lsinthm[ 24]:=3; lsinthm[ 20]:=2; lsinthm[  7]:=
3;

for i:=1 to nparam do
  begin
    zeros[i]:=0;
    smin[i]:=1;
    if i in bparam then
                  begin
                    smax[i]:=1;
                    smin[i]:=0;
                  end
                else smax[i]:=infinity;
  end;
(*****************************)
(*                           *)
(* assume no isolated nodes *)
(*                           *)
(*****************************)
         smin[nodes]:=2;
         smin[chr]:=2;
         smin[clique]:=2;
         smin[genus]:=0;
         smin[nconn]:=0;
         smin[econn]:=0;
         smin[xnum]:=0;
         smin[girth]:=3;
         smin[circ]:=3;
         slammin:=1.0;
         slammax:=infinity;
end;


procedure mainblock;


      (*************************************************)
      (*************************************************)
      (*                                               *)
      (*                                               *)
      (*              MAIN PROGRAM                     *)
```

```
     (*                                                  *)
     (*                                                  *)
     (****************************************************)
     (****************************************************)
var i:integer;


(*************************************)
                          (*
*)
begin                     (*              The STLIMIT procedure
*)
 (* stlimit(100000000); *) (*   must be pulled on most systems
*)
  tcpustart:=cpuClock;    (*
*)
  cpustart:=tcpustart;
(*************************************)
  oldcpuClock:=cpustart;
  startup;
  newStart;
  while go do
       begin
         writeln;
           pinPut;
         currentClock:=cpuClock;
         rz:=(currentClock-tcpustart)/perSecond;
         tcpustart:=currentClock;
         write(sysm:1,' Time used: ');
            if rz >= 60 then
                  begin
                      i:=trunk(rz/60);
                  if i = 1 then write(i:2,' minute and ')
                           else write(i:4,' minutes and ');
                       rz:=rz-i*60;
                     end;
            writeln(rz:5:2,' seconds.');
         if tnthms > 0 then writeln(sysm:1,tnthms:8,' theorems
executed.');
         writeln;
         write(sysm:1,' To terminate the session type "e", else
return. ?');
         read(op);
         if op = 'e' then go:=false
           else
              begin
                  if op <> blk then readln;
                  writeln;
                  stars;
                newStart;
                   write(sysm:1,' System Reinitialized.');
                if ntt = 0 then writeln
```

```
                    else
                      begin
                        for i:=1 to ntt do tempactive[i]:=false;
                        writeln(' Temporary Theorems have been
deactivated.');
                      end;
                end;
          end;
     dayTime;
     writeln;
     write('      Type any character--');
     readln(i);
  end;

end.
```