```
  unit rules050;
     interface
         uses
            globals,cmmnds1,pusherr,pushStack,
                  ruleAtoF;

      procedure r001; procedure r002; procedure r003; procedure
r004; procedure r005;
           procedure r006; procedure r007; procedure r008;
procedure r009; procedure r010;
           procedure r011; procedure r012; procedure r015;
procedure r016; procedure r017;
           procedure r018; procedure r019; procedure r020;
procedure r021; procedure r022;
           procedure r023; procedure r024; procedure r025;
procedure r026; procedure r027;
           procedure r028; procedure r029; procedure r030;
procedure r031; procedure r032;
           procedure r033; procedure r034; procedure r035;
procedure r036; procedure r037;
           procedure r038; procedure r039; procedure r040;
procedure r041; procedure r042;
           procedure r043; procedure r044; procedure r045;
procedure r046;procedure r047;
           procedure r048; procedure r049; procedure r050;



implementation


procedure r001;
(************************************************)
(*                                              *)
(*         E <= P(P-1)/2-P+Nconn+1              *)
(*            =(P-1)(P-2)/2+Nconn               *)
(*                                              *)
(************************************************)
begin if activerule[1] then
  begin
    rule:='  1/ ';
    z:=max[nodes];
    if z < infinity then
            begin
              if max[nconn] < infinity then
                  begin
                    z:=((z-1)*(z-2)) div 2 + max[nconn];
                    if z < max[edges] then pushmax(edges);
                  end;
              z:=max[nodes];
```

```
              z:=min[edges] - ((z-1)*(z-2)) div 2;
              if z > min[nconn] then pushmin(nconn);
            end;
     if max[nconn] < infinity then
              begin
                z:=8*min[edges]+1;
                if z-8*max[nconn] < 0 then
                         z:=round((1+sqrt(z))/2+hf)
                       else
                         z:=round((3+sqrt(z-8*max[nconn]))/2+hf);
                if z > min[nodes] then pushmin(nodes);
              end;
   end;
end;

procedure  r002;
(*********************************)
(*                               *)
(*   chr <= (ncov+clique+1)/2    *)
(*                               *)
(*********************************)
begin if activerule[2] then
  begin
    rule:='  2/ ';
    z:= max[ncov];
    if z < infinity then
          begin
            z:=2*min[chr]-z-1;
            if z > min[clique] then pushmin(clique);
            k:= max[clique];
            if k < infinity then
               begin
                 z:=(max[ncov]+k+1) div 2;
                 if z < max[chr] then pushmax(chr);
               end;
          end;
    if max[clique] < infinity then
                begin
                  z:=2*min[chr]-max[clique]-1;
                  if z > min[ncov] then pushmin(ncov);
                end;
  end;
end;

procedure r003;
(*************************************)
(*                                   *)
(*         spectr >= 2E/P            *)
(*                                   *)
(*************************************)
begin
  if activerule[3] then
```

```
      begin
        rule:='  3/ ';
        z:=max[nodes];
        if z < infinity then
            begin
               rz:=2*min[edges]/z;
               pushlammin;
             end;
        if lammax < infinity then
                     begin
                       if z < infinity then
                            begin
                               z:=trunc(lammax*z/2);
                               if z < max[edges] then
pushmax(edges);
                              end;
                       if lammax > 0 then
                            begin
                              z:=round(2*min[edges]/lammax+hf);
                              if z > min[nodes] then
pushmin(nodes);
                            end;
                    end;
        end;
end;

procedure r004;
(*************************************)
(*                                   *)
(*     spectr <= sqrt(2E*ncov/(ncov+1)) *)
(*                                   *)
(*************************************)
begin if activerule[4] then
     begin
       rule:='  4/ ';
       z:=max[ncov]+1;
       rz:=z;
       if max[ncov] < infinity then
              begin
                z:=round(z*lammin*lammin/(2*(z-1))+hf);
                if z > min[edges] then pushmin(edges);
              end;
       if max[edges] < infinity then
             begin
               if max[ncov] < infinity then
                   begin
                     rz:=sqrt(2*max[edges]*(rz-1)/rz);
                     pushlammax;
                   end;
               rz:=lammin*lammin;
               if 2*max[edges] > rz then
                   begin
```

```
                    z:=round(rz/(2*max[edges]-rz)+hf);
                    if z > min[ncov] then pushmin(ncov);
                  end;
             end;
      end;
end;

procedure r005;
(*********************************)
(*                               *)
(*      clique >= P**2/(P**2-2E)    *)
(*                               *)
(*********************************)
begin if activerule[5] then
        begin
          rule:='  5/ ';
          z1:=max[nodes];
          if max[clique] < infinity then
            begin
              z:=round(sqrt(2
*min[edges]*max[clique]/(max[clique]-1))+hf);
              if z > min[nodes] then pushmin(nodes);
            end;
          z1:=z1*z1;
          if z1 < infinity then
                begin
                   z:=(z1-1) div (z1-2*min[edges])+1;
                   if z > min[clique] then pushmin(clique);
                   if max[clique] < infinity then
                      begin
                        z:=(z1-((z1-1) div max[clique])-1) div 2;
                        if z < max[edges] then pushmax(edges);
                      end;
                 end;
           end;
end;

procedure r006;
(*****************************************)
(*                                       *)
(*           spectr <= maxdeg            *)
(*                                       *)
(*****************************************)
begin if activerule[6] then
     begin
         rule:='  6/ ';
         rz:=max[maxdeg];
         if rz < lammax then pushlammax;
         z:=round(lammin+hf);
         if z > min[maxdeg] then pushmin(maxdeg);
         end;
end;
```

```
procedure r007;
(*********************************************************)
(*                                                       *)
(*      if diam < infinity then                          *)
(*         (a)    if mindeg > 3*Nconn - 1 then           *)
(*                  P >= 1+mindeg+diam*Nconn+            *)
(*                        (diam div 3)*(mindeg-3*Nconn+1)  *)
(*                                                       *)
(*         (b)    P >= Nconn(diam-3)+2mindeg+2           *)
(*                                                       *)
(*********************************************************)
var k:longint;
begin if (activerule[7]) and (max[diam] < infinity) then
 begin
   rule:='  7/ ';
   k:=min[mindeg]-3*min[nconn]+1;
   z:=1+min[mindeg]+min[diam]*min[nconn]+(min[diam] div 3)*k;
   if min[mindeg] > 3*max[nconn]-1 then
         begin
           if z > min[nodes] then pushmin(nodes);
           if max[nodes] < infinity then
             begin
               z:=(3*max[nodes]-(min[mindeg]+1)*(min[diam]+1))
div 6;
               if z < max[nconn] then pushmax(nconn);
               z:=(3*max[nodes]-6*min[nconn]) div (min[diam]+
1) -1;
               if z < max[mindeg] then pushmax(mindeg);
               z:=(3*max[nodes]-6*min[nconn]) div (min[mindeg]+
1) -1;
               z1:=1+min[mindeg]+z*min[nconn]+(z div 3)*k;
               while z1 > max[nodes] do
                   begin
                     z:=z-1;
                     z1:=1+min[mindeg]+z*min[nconn]+(z div 3)*k;
                   end;
               if z < max[diam] then pushmax(diam);
             end;
         end
       else
         if max[nodes] < z then
             begin
               z:=3*max[nconn]-1;
               if z < max[mindeg] then pushmax(mindeg);
               z:=min[mindeg] div 3+1;
               if z > min[nconn] then pushmin(nconn);
             end;
   if min[diam] < 3 then z1:=max[nconn]
                    else z1:=min[nconn];
   z:=z1*(min[diam]-3)+2*min[mindeg]+2;
   if z > min[nodes] then pushmin(nodes);
```

```
   if max[nodes] < infinity then
              begin
                z:=(max[nodes]-z1*(min[diam]-3)-2) div 2;
                if z < max[mindeg] then pushmax(mindeg);
                if min[diam] > 1 then
                    begin
                      z:=(max[nodes]-2) div (min[diam]-1);
                      if z < max[nconn] then pushmax(nconn);
                    end;
                if min[diam] > 3 then
                         begin
                           z:=(max[nodes]-2*min[mindeg]-2) div
(min[diam]-3);
                           if z < max[nconn] then pushmax(nconn);
                         end
                       else
                         if max[diam] < 3 then
                              begin
                                z:=2*min[mindeg]+1-max[nodes];
                                z:=z div (3-min[diam])+1;
                                if z > min[nconn] then
pushmin(nconn);
                              end;
                if min[nconn] > 0 then
                    begin
                      z:=max[nodes]-2*min[mindeg]-2;
                      if z >= 0 then z:=z div min[nconn]+3
                                else z:=2;
                      if z < max[diam] then pushmax(diam);
                    end;
              end;
  end;
end;

procedure r008;
(*****************************************************)
(*                                                   *)
(*     P >= maxdeg+1+(mindeg+1)*(Ncomp-1)            *)
(*                                                   *)
(*****************************************************)
begin if activerule[8] then
   begin
     rule:='  8/ ';
     z:=min[maxdeg]+1+(min[mindeg]+1)*(min[ncomp]-1);
     if z > min[nodes] then pushmin(nodes);
     if max[nodes] < infinity then
         begin
           z:=max[nodes]-(z-min[maxdeg]);
           if z < max[maxdeg] then pushmax(maxdeg);
           z:=(max[nodes]-min[maxdeg]-1) div (min[mindeg]+1)+1;
           if z < max[ncomp] then pushmax(ncomp);
           if min[ncomp] > 1 then
```

```
              begin
                z:=(max[nodes]-min[maxdeg]-1) div
(min[ncomp]-1)-1;
                z1:=max[nodes] div min[ncomp] - 1;
                if z1 < z then z:=z1;
                if z < max[mindeg] then pushmax(mindeg);
              end;
         end;
   end;
end;

procedure r009;
(*****************************)
(*                           *)
(*      eccov<=P**2/4        *)
(*                           *)
(*****************************)
begin if activerule[9] then
       begin
         rule:='  9/ ';
         z:=max[nodes];
         if z < infinity then
                     begin
                       z:=(z*z) div 4;
                       if z < max[eccov] then pushmax(eccov);
                      end;
         z:=min[eccov];
         z:=round(2*sqrt(z)+hf);
         if z > min[nodes] then pushmin(nodes);
       end;
end;

procedure r010;
(*************************************************)
(*                                               *)
(*           diam <= 2*radius                    *)
(*                                               *)
(*************************************************)
begin if activerule[10] then
     begin
       rule:=' 10/ ';
        z:=2*max[radius];
        if z < max[diam] then pushmax(diam);
        if min[diam] >= max[nodes] then
               begin
                 z:=infinity;
                 if z > min[radius] then
                                 begin
                                   pushmin(radius);
                                   pushmin(diam);
                                 end;
               end
```

```
            else
              begin
                  z:=(min[diam]+1) div 2;
                  if z > min[radius] then pushmin(radius);
               end;
        end;
end;

procedure r011;
(*********************************************)
(*                                           *)
(*             eind <= P/2                   *)
(*                                           *)
(*********************************************)
begin if activerule[11] then
        begin
          rule:=' 11/ ';
          z:=max[nodes];
          if z < infinity then
                   begin
                     z:=z div 2;
                     if z < max[eind] then pushmax(eind);
                   end;
          z:=2*min[eind];
          if z > min[nodes] then pushmin(nodes);
         end;
end;

procedure r012;
(*****************************)
(*                           *)
(*   eind >= P/(1+maxdeg)    *)
(*                           *)
(*****************************)
begin if activerule[12] then
        begin
          rule:=' 12/ ';
          if max[maxdeg] < infinity then
            begin
              z:=(min[nodes]-1) div (1+max[maxdeg])+1;
              if z > min[eind] then pushmin(eind);
            end;
          z:=max[eind];
          if z < infinity then
                   begin
                     z:=(min[nodes]-1) div z;
                     if z > min[maxdeg] then pushmin(maxdeg);
                     if max[maxdeg] < infinity then
                       begin
                         z:=max[eind]*(1+max[maxdeg]);
                         if z < max[nodes] then pushmax(nodes);
                       end;
```

```
                  end;
       end;
end;

(*             R013 has been "RETIRED"
procedure r013;                                          *)
(************************************************)
(*                                                      *)
(*    if girth=2*diam +1 then regular                   *)
(*                                                      *)
(************************************************)

(*            R014 has been "RETIRED"
procedure r014;                                          *)
(*************************************)
(*                                   *)
(*       chr >= P/(P-Lambda)         *)
(*                                   *)
(*************************************)

procedure r015;
(*********************************************************)
(*                                                       *)
(*    if mindeg >= 3 then girth <= 2*log2(E-P+Ncomp)     *)
(*                                                       *)
(*********************************************************)
begin if (activerule[15]) and (max[mindeg] > 2) then
  begin
    rule:=' 15/ ';
    rz:=max[edges]-min[nodes]+max[ncomp];
    if rz < 1 then k2:=0
       else if max[edges]+max[ncomp] >= infinity then k2:
=infinity
                     else k2:=trunk(2*log2(rz));
    if min[mindeg] > 2 then
       begin
         z:=k2;
         if z < max[girth] then pushmax(girth);
         z:=min[girth];
         power(2,z,k1);
         if k1 < infinity then
            begin
              k1:=round(sqrt(k1)+hf);
              if max[ncomp] < infinity then
                begin
                  z:=k1+min[nodes]-max[ncomp];
                  if z > min[edges] then pushmin(edges);
                end;
              if max[edges] < infinity then
                begin
                  if max[ncomp] < infinity then
                     begin
```

```
                      z:=max[edges]-k1+max[ncomp];
                      if z < max[nodes] then pushmax(nodes);
                    end;
                  z:=k1-max[edges]+min[nodes];
                  if z > min[ncomp] then pushmin(ncomp);
                end;
             end;
        end
         else
           if min[girth] > k2 then
               begin
                 z:=2;
                 pushmax(mindeg);
               end;
  end;
end;

procedure r016;
(*************************************************************)
(*                                                           *)
(*            Nconn=0  <==>  Econn=0                         *)
(*                                                           *)
(*************************************************************)
begin if (activerule[16]) and (max[econn] > 0) and (min[nconn] =
0) then
   begin
     rule:=' 16/ ';
     z:=0;
     if max[nconn]=0 then pushmax(econn)
        else
          if min[econn] > 0 then
                  begin
                    z:=1;
                    pushmin(nconn);
                  end;
    end;
end;

procedure r017;
(***********************************************)
(*                                             *)
(*      E<=eccov*clique(clique-1)/2            *)
(*                                             *)
(***********************************************)
var w:longint;
begin if activerule[17] then
     begin
       rule:=' 17/ ';
       w:=max[clique];
       if (max[eccov] < infinity) and (w < infinity) then
            begin
              rz:=max[eccov];
```

```
              rz:=rz*w*(w-1)/2;
              if rz < infinity then
                 begin
                   z:=max[eccov]*w*(w-1) div 2;
                   if z < max[edges] then pushmax(edges);
                 end;
             end;
       if w < infinity  then
            begin
              w:=(w*(w-1)) div 2;
              z:=(min[edges]-1) div w+1;
              if z > min[eccov] then pushmin(eccov);
            end;
       if max[eccov] < infinity then
             begin
               z:=(2*min[edges]-1) div max[eccov];
               z:=round((1+sqrt(2+4*z))/2+hf);
               if z > min[clique] then pushmin(clique);
             end;
      end;
end;

procedure r018;
(*****************************************)
(*                                       *)
(*      chr <= (7+sqrt(1+48*genus))/2    *)
(*                                       *)
(*****************************************)
begin if activerule[18] then
       begin
         rule:=' 18/ ';
         if max[genus] < infinity then
                begin
                  z:=1+48*max[genus];
                  if z < infinity then
                    begin
                      z:=trunk((7+sqrt(z))/2);
                      if z < max[chr] then pushmax(chr);
                    end;
                 end;
         z:=min[chr];
         if z > 4 then
              begin
                   z:=((z-4)*(z-3)-1) div 12+1;
                   if z > min[genus] then pushmin(genus);
                end;
       end;
end;

procedure r019;
(*****************************************)
(*                                       *)
```

```
(*      if clique = 2 then                *)
(*                   maxdeg<=nind         *)
(*                   E<=ncov*nind         *)
(*                                        *)
(******************************************)
begin if (activerule[19]) and (min[clique] <= 2) then
     begin
       rule:=' 19/ ';
       rulef(ncov,nind);
       z1:=z;
       if max[clique] = 2 then
                 begin
                   z:=min[maxdeg];
                   if z > min[nind] then pushmin(nind);
                   z:=max[nind];
                   if z < infinity then
                         begin
                           if z < max[maxdeg] then
pushmax(maxdeg);
                           z:=(min[edges]-1) div z+1;
                           if z > min[ncov] then pushmin(ncov);
                           if max[ncov] < infinity then
                               begin
                                 z:=z1;
                                 if z < max[edges] then
pushmax(edges);
                               end;
                         end;
                    if max[ncov] < infinity then
                             begin
                               z:=(min[edges]-1) div max[ncov]+1;
                               if z > min[nind] then
pushmin(nind);
                             end;
                end
               else
                  if (min[edges] > z1) or
                     (min[maxdeg] > max[nind]) then
                          begin
                            z:=3;
                            pushmin(clique);
                          end;
        end;
end;

procedure r020;
(**************************************)
(*                                    *)
(*    if chr = 2 then                 *)
(*     (i.e. bipartite)               *)
(*              nind=nccov            *)
(*              eind=alpaha0          *)
```

```
(*              echr=maxdeg             *)
(*              girth and circ even     *)
(*              if P > 2 then           *)
(*                      not complete    *)
(*                                      *)
(****************************************)
begin if (activerule[20]) and (min[chr] <= 2) then
     begin
       rule:=' 20/ ';
       z:=3;
       if max[chr] = 2 then
            begin
              rulea(nccov,nind,0);
              rulea(ncov,eind,0);
              rulea(echr,maxdeg,0);
              z:=min[girth]+1;
              if not(odd(z)) then pushmin(girth);
              z:=min[circ]+1;
              if not(odd(z)) then pushmin(circ);
              z:=max[girth]-1;
              if (max[girth] < infinity) and (not(odd(z))) then
                                          pushmax(girth);
              z:=max[circ]-1;
              if (max[circ] < infinity) and (not(odd(z))) then
pushmax(circ);
              echrRmaxdeg:=eq;
              z:=0;
              if min[nodes] > 2 then pushmax(compl);
             end
           else
             if (max[nind] < min[nccov]) or
                (min[echr] > max[maxdeg]) or
                (echrRmaxdeg = gt) or
                (max[eind] < min[ncov]) or
                ((min[girth]=max[girth]) and (max[girth]
<infinity)
                 and (odd(max[girth]))) or
                ((min[circ]=max[circ]) and (max[circ]<infinity)
                 and (odd(max[circ]))) then pushmin(chr);
       end;
end;

procedure r021;
(****************************************)
(*                                      *)
(*    genus<=CL((P-3)(P-4)/12)          *)
(*                                      *)
(****************************************)
begin if activerule[21] then
          begin
             rule:=' 21/ ';
             z:=max[nodes];
```

```
             if z < infinity then
                  begin
                    if z < 5 then z:=0
                          else  z:=((z-3)*(z-4)+11) div 12;
                    if z < max[genus] then pushmax(genus);
                  end;
              if min[genus] > 0 then
                  begin
                    z:=round((7+sqrt(48*min[genus]-43))/2+hf);
                    if z > min[nodes] then pushmin(nodes);
                  end;
            end;
end;

procedure r022;
(*******************************************)
(*                                         *)
(*       if E > maxdeg*eind then           *)
(*                    echr=maxdeg+1        *)
(*           (i.e. a 'class two' graph)    *)
(*                                         *)
(*******************************************)
begin if activerule[22] then
     begin
       rule:=' 22/ ';
       if  min[edges] > max[maxdeg]*max[eind] then
                           begin
                             rulea(maxdeg,echr,-1);
                             echrRmaxdeg:=gt;
                           end
                else
                    if (min[maxdeg]=max[echr]) or (echrRmaxdeg =
eq) then
                           begin
                              ruleb(edges,maxdeg,eind);
                              echrRmaxdeg:=eq;
                            end;
      end;
end;

procedure r023;
(*************************************)
(*                                   *)
(*         eccov <= ecov*eind        *)
(*                                   *)
(*************************************)
begin if activerule[23] then
   begin
     rule:=' 23/ ';
     if max[ecov] < infinity then
        begin
          z:=(min[eccov]-1) div max[ecov]+1;
```

```
          if z > min[eind] then pushmin(eind);
          if max[eind] < infinity then
               begin
                 rulef(ecov,eind);
                 if z < max[eccov] then pushmax(eccov);
               end;
         end;
       if max[nind] < infinity then
            begin
              z:=(min[eccov]-1) div max[eind]+1;
              if z > min[ecov] then pushmin(ecov);
            end;
   end;
end;

procedure r024;
(*********************************************************)
(*                                                       *)
(*       d=mindeg > 2                                    *)
(*         if girth = 2*r+1 then                         *)
(*                    P >= (d(d-1)**r-2)/(d-2)           *)
(*         if girth = 2*r  then                          *)
(*                    P >= (2(d-1)**r-2)/(d-2)           *)
(*                                                       *)
(*********************************************************)
var r,d:longint;
begin
  if (activerule[24]) and (min[mindeg] > 2) then
     begin
       rule:=' 24/ ';
       d:=min[mindeg];
       r:=min[girth] div 2;
       power(d-1,r,z);
       if z < infinity then
                begin
                   if odd(min[girth]) then z:=(d*z-2) div (d-2)
                                      else z:=(2*z-2) div (d-2);
                   if z > min[nodes] then pushmin(nodes);
                end
                 else pushmin(nodes);
       if max[nodes] < infinity then
             begin
               rz:=d-1;
               rhb:=((rz-1)*max[nodes]+2)/2;
               z:=2*round(log2(rhb)/log2(rz)+hf);
               if z < max[girth] then pushmax(girth);
               if odd(min[girth]) then
                   begin
                     z:=trunk(root(max[nodes],r))+1;
                     power(z-1,r,z1);
                     if z1 < infinity then
                        while (z*z1-2) div (z-2) > max[nodes] do
```

```
                          begin
                            z:=z-1;
                            power(z-1,r,z1);
                          end;
                    end
                 else
                    begin
                      z:=trunk(root(max[nodes] div 2,r-1))+1;
                      power(z-1,r,z1);
                      if z1 < infinity then
                        while (2*z1-2) div (z-2) > max[nodes] do
                          begin
                            z:=z-1;
                            power(z-1,r,z1);
                          end;
                    end;
                if z < max[mindeg] then pushmax(mindeg);
              end;
 end;
end;

procedure r025;
(************************)
(*                      *)
(*    ecov >= P/2       *)
(*                      *)
(************************)
begin
  if activerule[25] then
       begin
         rule:=' 25/ ';
         z:=(min[nodes]+1) div 2;
         if z > min[ecov] then pushmin(ecov);
         z:=2*max[ecov];
         if z < max[nodes] then pushmax(nodes);
       end;
end;

procedure r026;
(*************************************************)
(*                                               *)
(*     E <= (P-1)*(P-2)/2 + P/dom - 1            *)
(*                                               *)
(*************************************************)
begin
  if activerule[26] then
    begin
      rule:=' 26/ ';
      rz:=3.0-2.0/min[dom];
      z:=round((rz+sqrt(rz*rz+8*min[edges]))/2+hf);
      if z > min[nodes] then pushmin(nodes);
      if max[nodes] < infinity then
```

```
          begin
            k:=max[nodes];
            z:=(k-1)*(k-2) div 2-1+k div min[dom];
            if z < max[edges] then pushmax(edges);
            z:=min[edges]+1-(((k-1)*(k-2)) div 2);
            if z > 0 then
               begin
                 z:=k div z;
                 if z < max[dom] then pushmax(dom);
               end;
          end;
     end;
end;

procedure r027;
(*****************************************)
(*                                       *)
(*         ecov<=P*maxdeg/(1+maxdeg)     *)
(*                                       *)
(*****************************************)
begin if activerule[27] then
     begin
       rule:=' 27/ ';
       z:=max[nodes];
       if z < infinity then
               begin
                if z < max[maxdeg] then
                     begin
                       z:=z-1;
                       pushmax(maxdeg);
                       z:=z+1;
                     end;
                z:=z*max[maxdeg] div (1+max[maxdeg]);
                if z < max[ecov] then pushmax(ecov);
                z:=max[nodes]-min[ecov];
                if z > 0 then
                    begin
                      z:=(min[ecov]-1) div z+1;
                      if z > min[maxdeg] then pushmin(maxdeg);
                    end;
             end;
        if max[maxdeg] < infinity then
            begin
              z:=min[ecov]-1;
              z:=z div max[maxdeg]+z+2;
              if z > min[nodes] then pushmin(nodes);
            end;
     end;
end;

procedure r028;
(*****************************************)
```

```
(*                                          *)
(*  if diam < infinity then                 *)
(*  (a)  if  diam <= 3 then                 *)
(*                  maxdeg <= P-diam+1      *)
(*  (b)   maxdeg <= P-Nconn*(diam-4)-3      *)
(*                                          *)
(******************************************)
begin if (activerule[28]) and (max[diam] < infinity) then
begin
  rule:=' 28/ ';
  if min[diam] <= 3 then
      begin
        z:=min[maxdeg]+min[diam]-1;
        if z > min[nodes] then pushmin(nodes);
        if max[nodes] < infinity then
             begin
               z:=max[nodes]-min[diam]+1;
               if z < max[maxdeg] then pushmax(maxdeg);
               z:=max[nodes]-min[maxdeg]+1;
               if z < max[diam] then pushmax(diam);
             end;
      end
     else
       begin
         if min[nconn] = 0 then k:=1
                           else k:=min[nconn];
         z:=min[maxdeg]+k*(min[diam]-4)+3;
         if z > min[nodes] then pushmin(nodes);
         if max[nodes] < infinity then
           begin
             z:=max[nodes]-k*(min[diam]-4)-3;
             if z < max[maxdeg] then pushmax(maxdeg);
             z:=(max[nodes]-min[maxdeg]-3) div k +4;
             if z < max[diam] then pushmax(diam);
             if min[diam] > 4 then
                    begin
                      z:=(max[nodes]-min[maxdeg]-3) div
(min[diam]-4);
                      if z < max[nconn] then pushmax(nconn);
                    end
           end;
        end;
 end;
end;

procedure r029;
(*************************************************)
(*                                               *)
(*      eccov<=E-clique(clique-1)/2 + 1          *)
(*                                               *)
(*************************************************)
var w:longint;
```

```
begin if activerule[29] then
   begin
     rule:=' 29/ ';
     w:=min[clique];
     w:=(w*(w-1)) div 2;
     if max[edges] < infinity then
             begin
               z:=max[edges]-w+1;
               if z < max[eccov] then pushmax(eccov);
               z:=8*(max[edges]-min[eccov]);
               z:=trunk((1+sqrt(z+9))/2);
               if z < max[clique] then pushmax(clique);
             end;
     z:=min[eccov]+w-1;
     if z > min[edges] then pushmin(edges);
   end;
end;

procedure r030;
(*************************************************************
**)
(*
*)
(*        if Nconn > 0 then
*)
(*              E <= f(P,Radius)
*)
(*
*)
(*-----------------------------------------------------------
-*)
(*           ___
*)
(*          |  infinity          not conn.  or     P=infinity
*)
(*          |
*)
(*          |  P(P-k)/2          1<=k<=min(2,P/2)
*)
(* f(P,k)=  |
*)
(*          |(P*P-4kP+5P+4k*k-6k)/2   3<=k<=P/2
*)
(*
*)
(*
*)
(*************************************************************
**)

      function f(p,k:longint):longint;
            begin
```

```
             if k < 3 then f:=p*(p-k) div 2
                      else f:=(p*p-4*k*p+5*p+4*k*k-6*k) div 2;
            end;

begin if (activerule[30]) and (min[nconn] > 0) and (max[nodes]
< infinity) then
  begin
    rule:=' 30/ ';
    z:=f(max[nodes],min[radius]);
    if z < max[edges] then pushmax(edges);
    z:=max[nodes];
    rz:=min[edges];
    if rz > z*(z-2)/2 then
        begin
          z:=1;
          pushmax(radius);
          if z > min[radius] then pushmin(radius);
        end
      else
        if (min[edges]=max[edges]) and (min[nodes]=z) and
(min[edges]=z-1) then
            begin
              z:=z div 2;
              if z < max[radius] then pushmax(radius);
            end
          else
            begin
              z:=8*min[edges]-8*max[nodes]+9;
              if z >=0 then
                   begin
                     z:=trunk((2*max[nodes]+3-sqrt(z))/4);
                     if z < 2 then z:=2;
                     if z < max[radius] then pushmax(radius);
                   end;
            end;
    z:=min[radius];
    if z <= 2 then
         begin
           z:=round((z+sqrt(z*z+8*min[edges]))/2+hf);
           if z > min[nodes] then pushmin(nodes);
         end
       else
         begin
           z:=8*min[edges]-16*z+25;
           if z >= 0 then
                begin
                  z:=round((4*min[radius]-5+sqrt(z))/2+hf);
                  if z > min[nodes] then pushmin(nodes);
                end;
         end;
   end;
end;
```

```
procedure r031;
(*************************************)
(*                                   *)
(*       chr*nccov<=((P+1)/2)**2     *)
(*                                   *)
(*************************************)
begin if activerule[31] then
    begin
      rule:=' 31/ ';
      if max[nodes] < infinity then
        begin
          z:=max[nodes]+1;
          z1:=z*z;
          z:=min[nccov];
          z:=z1 div (4*z);
          if z < max[chr] then pushmax(chr);
          z:=z1 div (4*min[chr]);
          if z < max[nccov] then pushmax(nccov);
         end;
      z:=min[chr]*min[nccov];
      z:=round(2*sqrt(z)+hf)-1;
      if z > min[nodes] then pushmin(nodes);
     end;
end;

procedure r032;
(*************************************)
(*                                   *)
(*       chr + nccov >= 2*sqrt(P)    *)
(*                                   *)
(*************************************)
begin if activerule[32] then
      begin
        rule:=' 32/ ';
        z:=round(2*sqrt(min[nodes])+hf)-max[nccov];
        if z > min[chr] then pushmin(chr);
        z:=z+max[nccov]-max[chr];
        if z > min[nccov] then pushmin(nccov);
        z:=max[chr]+max[nccov];
        rz:=z;
        rz:=rz*rz/4;
        if rz < infinity then
                  begin
                    z:=z*z div 4;
                    if z < max[nodes] then pushmax(nodes);
                  end;
      end;
end;

procedure r033;
(*************************************)
```

```
(*                                      *)
(*       dom <= P+1-sqrt(1+2*E)         *)
(*                                      *)
(****************************************)
begin if activerule[33] then
   begin
     rule:=' 33/ ';
     z:=round(min[dom]-1+sqrt(1+2*min[edges])+hf);
     if z > min[nodes] then pushmin(nodes);
     if max[nodes] < infinity then
       begin
         z:=trunk(max[nodes]+1-sqrt(1+2*min[edges]));
         if z < max[dom] then pushmax(dom);
         z:=max[nodes]-min[dom];
         z:=(z*(z+2)) div 2;
         if z < max[edges] then pushmax(edges);
       end;
   end;
end;

procedure r034;
(*********************************************************)
(*                                                       *)
(*        if Nconn > 0 and not a tree then               *)
(*               girth <= 2*diam + 1                     *)
(*                                                       *)
(*********************************************************)
begin if (activerule[34]) and (max[nconn] > 0) and (min[tree] =
0) then
       begin
         rule:=' 34/ ';
         z:=0;
         if (min[nconn] > 0) and (max[tree]=0)  then
                begin
                  z:=2*max[diam] + 1;
                  if z < max[girth] then pushmax(girth);
                  z:=min[girth] div 2;
                  if z > min[diam] then pushmin(diam);
                end
            else
              if min[girth] > (2*max[diam] +1) then
                 if max[tree]=0 then pushmax(nconn)
                       else
                         if min[nconn] > 0 then
                                  begin
                                    z:=1;
                                    pushmin(tree);
                                  end;
        end;
end;

procedure r035;
```

```
(*******************************************)
(*                                         *)
(*    if planar then either                *)
(*                  nind >= (P+1)/3        *)
(*                  ncov<= (2P-1)/3        *)
(*            or    clique >= 3            *)
(*                                         *)
(*******************************************)
begin if (activerule[35]) and (max[plnar]=1) then
begin
  rule:=' 35/ ';
  z:=0;
  if min[plnar]=1 then
     begin
       if max[clique] < 3 then
          begin
            z:=min[nodes] div 3+1;
            if z > min[nind] then pushmin(nind);
            if max[nind] < infinity then
                begin
                  z:=3*max[nind]-1;
                  if z < max[nodes] then pushmax(nodes);
                end;
            if max[nodes] < infinity then
                 begin
                   z:=(2*max[nodes]-1) div 3;
                   if z < max[ncov] then pushmax(ncov);
                 end;
            z:=(3*min[ncov]) div 2+1;
            if z > min[nodes] then pushmin(nodes);
           end
          else
            if (max[nind] < (min[nodes]+3) div 3) or
               (min[ncov] > (2*max[nodes]-1) div 3) then
                 begin
                   z:=3;
                   if z > min[clique] then pushmin(clique);
                 end;
     end
       else
           if ((max[nind] < (min[nodes]+3) div 3) or
              (min[ncov] > (2*max[nodes]-1) div 3)) and
              (max[clique] < 3) then
                          pushmax(plnar);
          end;
end;

procedure r036;
(*************************************************)
(*                                               *)
(*     Nonplnar ==> maxdeg>=3, P>=5, E>=9,      *)
(*                   eind>=2, ncov>=3, ecov>=3    *)
```

```
(*                    Bwidth>=4                          *)
(*                                                       *)
(*******************************************************)
begin if (activerule[36]) and (min[plnar] = 0) then
begin
  rule:=' 36/ ';
  z:=1;
  if max[plnar]=0 then
         begin
           z:=3;
           if z > min[maxdeg] then pushmin(maxdeg);
           if z > min[ncov] then pushmin(ncov);
           if z > min[ecov] then pushmin(ecov);
           z:=2;
           if z > min[eind] then pushmin(eind);
           z:=4;
           if z > min[bwidth] then pushmin(bwidth);
           z:=5;
           if z > min[nodes] then pushmin(nodes);
           z:=9;
           if z > min[edges] then pushmin(edges);
         end
       else
         if (max[maxdeg] < 3) or (max[nodes] < 5) or
             (max[edges] < 9) or (max[eind] < 2) or
             (max[bwidth] < 4) or
             (max[ncov] < 3) or (max[ecov] < 3) then
               pushmin(plnar);
  end;
end;

procedure r037;
(*******************************************************)
(*                                                     *)
(*   E <= Ncomp-1+(P-2(Ncomp-1))*(P-2(Ncomp-1)-1)/2    *)
(*                                                     *)
(*******************************************************)
var p,nc:longint;
begin
  if activerule[37] then
     begin
       rule:=' 37/ ';
       nc:=min[ncomp];
       if max[nodes] < infinity then
           begin
             p:=max[nodes]-2*nc+2;
             z:=p*(p-1) div 2 +nc-1;
             if z < max[edges] then pushmax(edges);
             rz:=2*min[edges]-max[nodes]+1;
             if rz >= 0 then
                 begin
                   rz:=(max[nodes]+1-sqrt(rz));
```

```
                  z:=trunk(rz/2);
                  if z < max[ncomp] then pushmax(ncomp);
                end;
           end;
       p:=nc-1;
       z:=1+8*(min[edges]-p);
       if z >= 0 then
          begin
            z:=round((1+sqrt(z))/2+hf)+nc-1+p;
            if z > min[nodes] then pushmin(nodes);
          end;
     end;
end;

procedure r038;
(*****************************************)
(*                                       *)
(*     dom >= P/(maxdeg+1)               *)
(*                                       *)
(*****************************************)
begin if activerule[38] then
    begin
      rule:=' 38/ ';
      if max[maxdeg] < infinity then
         begin
           z:=(min[nodes]-1) div (max[maxdeg]+1)+1;
           if z > min[dom] then pushmin(dom);
           if max[dom] < infinity then
               begin
                 z:=max[dom]*(max[maxdeg]+1);
                 if z < max[nodes] then pushmax(nodes);
               end;
         end;
       if max[dom] < infinity then
             begin
               z:=(min[nodes]-1) div max[dom];
               if z > min[maxdeg] then pushmin(maxdeg);
             end;
    end;
end;


procedure r039;
(*************************************)
(*                                   *)
(*      clique = 2 <=> girth >=4     *)
(*                                   *)
(*************************************)
begin if activerule[39] then
   begin
     rule:=' 39/ ';
     if (max[clique] < 3) or (min[girth] > 3) then
```

```
        begin
          z:=2;
          if 2 < max[clique] then pushmax(clique);
          z:=4;
          if 4 > min[girth] then pushmin(girth);
        end
      else
        if (min[clique] > 2) or (max[girth] < 4) then
            begin
              z:=3;
              if 3 < max[girth] then pushmax(girth);
              if 3 > min[clique] then pushmin(clique);
            end;
   end;
end;

procedure r040;
(*********************************************)
(*                                           *)
(*       the following are equivalent.       *)
(*          complete, mindeg=P-1,            *)
(*          nind=nccov=eccov=1,              *)
(*          diam=1                           *)
(*                                           *)
(*********************************************)
begin if activerule[40] then
    begin
      rule:=' 40/ ';
      z:=1;
      if (min[compl] = 1) or (min[mindeg] = max[nodes]-1) or
         (mindegRpminus1 = eq) or (max[nind] = 1) or
         (max[nccov] = 1) or (max[eccov] = 1) or
         (max[diam] = 1) then
               begin
                  pushmin(compl);
                  pushmax(nind);
                  pushmax(nccov);
                  pushmax(eccov);
                  pushmax(diam);
                  rulea(nodes,mindeg,1);
                  mindegRpminus1:=eq;
                end
             else
               if (max[compl] = 0) or (max[mindeg]
< min[nodes]-1) or
                  (mindegRpminus1=lt) or (min[nind] > 1) or
                  (min[nccov] > 1) or (min[eccov] > 1) or
                  (min[diam] > 1) then
                      begin
                        z:=0;
                        pushmax(compl);
                        z:=2;
```

```
                        if 2 > min[nind] then pushmin(nind);
                        if 2 > min[nccov] then pushmin(nccov);
                        if 2 > min[eccov] then pushmin(eccov);
                        if 2 > min[diam] then pushmin(diam);
                        rulea(mindeg,nodes,-2);
                        mindegRpminus1:=lt;
                      end;
     end;
end;

procedure r041;
(*********************************)
(*                               *)
(*      bipartite <=> chr=2      *)
(*                               *)
(*********************************)
begin if activerule[41] then
      begin
        rule:=' 41/ ';
        if (min[bipart]=1) or (max[chr] < 3) then
             begin
               z:=1;
               if 1 > min[bipart] then pushmin(bipart);
               z:=2;
               if 2 < max[chr] then pushmax(chr);
             end
           else
             if (max[bipart] =0) or (min[chr] > 2) then
                      begin
                        z:=0;
                        if 0 < max[bipart] then pushmax(bipart);
                        z:=3;
                        if 3 > min[chr] then pushmin(chr);
                      end;
      end;
end;

procedure r042;
(*************************************)
(*                                   *)
(*       radius=1 <==> maxdeg=P-1    *)
(*                                   *)
(*************************************)
begin if activerule[42] then
begin
  rule:=' 42/ ';
  if (max[radius]=1) or (min[maxdeg]=max[nodes]-1) then
            begin
              z:=1;
              pushmax(radius);
              rulea(nodes,maxdeg,1);
            end
```

```
          else
            if (min[radius] > 1) or (max[maxdeg] < min[nodes]-1)
then
                 begin
                   z:=2;
                   if 2 > min[radius] then pushmin(radius);
                   rulea(maxdeg,nodes,-2);
                 end;
end;
end;

procedure r043;
(*************************************)
(*                                   *)
(*   tree <=> forest and connected   *)
(*                                   *)
(*************************************)
begin if activerule[43] then
    begin
      rule:=' 43/ ';
      z:=1;
      if (min[tree] = 1) or ((min[forest]=1) and (min[connct]=1))
then
             begin
               if 1 > min[forest] then pushmin(forest);
               if 1 > min[connct] then pushmin(connct);
               if 1 > min[tree] then pushmin(tree);
             end
         else
           if (max[tree]=0) or (max[forest]=0) or (max[connct]=0)
then
              begin
                z:=0;
                if 0 < max[tree] then pushmax(tree);
                if (min[forest] = 1) and (0 < max[connct]) then
pushmax(connct);
                if (min[connct] = 1) and (0 < max[forest]) then
pushmax(forest);
              end;
    end;
end;

procedure r044;
(*****************************************)
(*                                       *)
(* the following are equivalent:         *)
(*                                       *)
(*      connected,Nconn > 0,Ncomp = 1,   *)
(*      radius <= P/2,diam <= P-1        *)
(*                                       *)
(*****************************************)
begin if activerule[44] then
```

```
      begin
        rule:=' 44/ ';
        z:=1;
        if (min[connct] = 1) or (max[ncomp] = 1) or
           (min[nconn] > 0) or
           (max[radius] <= max[nodes] div 2) or
           (max[diam] < max[nodes]) then
                begin
                  if 1 > min[connct] then pushmin(connct);
                  if 1 < max[ncomp] then pushmax(ncomp);
                  if 1 > min[nconn] then pushmin(nconn);
                  z:=max[nodes];
                  if z < infinity then
                           begin
                             z:=z-1;
                             if z < max[diam] then
pushmax(diam);
                             z:=(z+1) div 2;
                             if z < max[radius] then
pushmax(radius);
                           end;
                   z:=min[diam]+1;
                   if z > min[nodes] then pushmin(nodes);
                   z:=min[radius]*2;
                   if z > min[nodes] then pushmin(nodes);
                 end
            else
              if (max[connct]=0) or (min[ncomp]>1) or
                 (max[nconn]=0) or
                 (min[radius] = infinity) or
                 (min[diam] = infinity) then
                        begin
                          z:=0;
                          if 0 < max[connct] then
pushmax(connct);
                          if 0 < max[nconn] then
pushmax(nconn);
                          z:=2;
                          if 2 > min[ncomp] then
pushmin(ncomp);
                          z:=infinity;
                          if z > min[radius] then
                                    begin
                                       pushmin(radius);
                                       pushmin(diam);
                                    end;
                        end;
      end;
end;

procedure r045;
(*********************************************)
```

```
(*                                             *)
(*  cycle <=> maxdeg=mindeg=2 and connected    *)
(*                                             *)
(***********************************************)
begin if activerule[45] then
         begin
           rule:=' 45/ ';
           z:=1;
           if (min[cycle] = 1) or ((min[mindeg] = max[maxdeg])
and
              (max[maxdeg] = 2) and (min[connct] = 1)) then
                 begin
                   pushmin(cycle);
                   pushmin(connct);
                   z:=2;
                   pushmin(mindeg);
                   pushmax(maxdeg);
                  end
              else
                if (max[cycle] = 0) or (max[connct] = 0) or
                   (max[mindeg] < 2) or (min[mindeg] > 2) or
                   (max[maxdeg] < 2) or (min[maxdeg] > 2) then
                     begin
                       z:=0;
                       if 0 < max[cycle] then pushmax(cycle);
                       if min[connct] = 1 then
                         begin
                           if min[mindeg] = 2 then
                             begin
                               z:=3;
                               if 3 > min[maxdeg] then
pushmin(maxdeg);
                             end;
                           if (min[maxdeg] = max[maxdeg]) and
                              (min[maxdeg] = 2) then
                                 begin
                                   z:=1;
                                   if 1 < max[mindeg] then
pushmax(mindeg);
                                 end;
                         end
                       else
                         if (min[mindeg] = max[maxdeg]) and
                            (min[mindeg] = 2) and (z
< max[connct]) then
                                    pushmax(connct);
                     end;
            end;
end;

procedure r046;
(*******************************)
```

```
(*                                *)
(*  regular <=> mindeg=maxdeg     *)
(*                                *)
(**********************************)
begin if activerule[46] then
        begin
          rule:=' 46/ ';
          z:=0;
          if (min[reg] = 1) or (min[mindeg] = max[maxdeg]) then
                begin
                  rulea(maxdeg,mindeg,0);
                  rulea(mindeg,maxdeg,0);
                  z:=1;
                  if 1 > min[reg] then pushmin(reg);
                end
             else
               if (max[reg] = 0) or (max[mindeg] < min[maxdeg])
then
                     begin
                       pushmax(reg);
                       rulea(mindeg,maxdeg,-1);
                     end;
         end;
end;

procedure r047;
(************************************)
(*                                  *)
(* plnar <=> genus=0 <=> thick=1    *)
(*                                  *)
(************************************)
begin if activerule[47] then
         begin
           rule:=' 47/ ';
           z:=1;
           if (min[plnar] = 1) or (max[genus] = 0) or
              (max[thick] = 1) then
                 begin
                   if 1 > min[plnar] then pushmin(plnar);
                   if 1 < max[thick] then pushmax(thick);
                   z:=0;
                   if 0 < max[genus] then pushmax(genus);
                 end
             else
               if (max[plnar] = 0) or (min[genus] > 0) or
                  (min[thick] > 1) then
                      begin
                        if 1 > min[genus] then pushmin(genus);
                        z:=0;
                        if 0 < max[plnar] then pushmax(plnar);
                        z:=2;
                        if 2 > min[thick] then pushmin(thick);
```

```
                    end;
      end;
end;

procedure r048;
(*****************************************)
(*                                       *)
(*   Forest  => plnar,chr=2,mindeg = 1    *)
(*                                       *)
(*****************************************)
begin if (activerule[48]) and (max[forest] = 1) then
begin
  rule:=' 48/ ';
  z:=0;
  if min[forest] = 1 then
      begin
        z:=1;
        if 1 > min[plnar] then pushmin(plnar);
        if 1 < max[mindeg] then pushmax(mindeg);
        z:=2;
        if 2 < max[chr] then pushmax(chr);
        if 2 > min[chr] then pushmin(chr);
      end
    else
      if ((max[plnar] = 0) or (min[chr] > 2) or
         (min[mindeg] > 1)) and (z < max[forest]) then
pushmax(forest);
end;
end;

procedure r049;
(*********************************************************)
(*                                                       *)
(*       Cycle => plnar,girth=circ=nodes,forest=no,      *)
(*                P >= 3                                 *)
(*                chr=2 if nodes is even else 3          *)
(*                echr=chr                               *)
(*                clique=2 if P > 3, else = 3            *)
(*                arbor=2                                *)
(*                ecov=ncov=(P+1) div 2                  *)
(*                nind=eind=P div 2                      *)
(*                nccov=ecov (P > 3)                     *)
(*                radius=eind                            *)
(*                xnum=0                                 *)
(*                Nconn=2                                *)
(*                regular                                *)
(*                Bwidth=2                               *)
(*                                                       *)
(*********************************************************)
begin if (activerule[49]) and (max[cycle] = 1) then
       begin
         rule:=' 49/ ';
```

```
          if min[cycle]=1 then
              begin
                z:=1;
                pushmin(plnar);
                pushmin(reg);
                z:=0;
                pushmax(forest);
                pushmax(xnum);
                z:=3;
                pushmin(nodes);
                pushmin(edges);
                pushmax(chr);
                pushmax(clique);
                z:=2;
                pushmin(arbor);
                pushmax(arbor);
                pushmin(nconn);
                pushmax(nconn);
                pushmin(bwidth);
                pushmax(bwidth);
                if min[nodes] > 3 then pushmax(clique);
                if max[nodes]=min[nodes] then
                          if odd(max[nodes]) then
                                              begin
                                                z:=3;
                                                pushmin(chr);
                                               end
                                          else pushmax(chr);
               if max[chr]=2 then
                  begin
                    if odd(min[nodes]) then
                       begin
                         z:=min[nodes]+1;
                         pushmin(nodes);
                       end;
                     if (max[nodes] < infinity) and
(odd(max[nodes])) then
                       begin
                         z:=max[nodes]-1;
                         pushmax(nodes);
                       end;
                   end
                  else
                   if min[chr]=3 then
                      begin
                        if not(odd(min[nodes])) then
                             begin
                               z:=min[nodes]+1;
                               pushmin(nodes);
                             end;
                        if (max[nodes] < infinity) and
(not(odd(max[nodes])))
```

```
                        then
                          begin
                            z:=max[nodes]-1;
                            pushmax(nodes);
                          end;
                  end;
            if max[clique]=2 then
                   begin
                     z:=4;
                     pushmin(nodes);
                   end
                  else
                    if min[clique]=3 then
                            begin
                              z:=3;
                              pushmax(nodes);
                             end;
            if max[nodes] < infinity then
                    begin
                      z:=(max[nodes]+1) div 2;
                      if z < max[ncov] then pushmax(ncov);
                      if z < max[ecov] then pushmax(ecov);
                      z:=max[nodes] div 2;
                      if z < max[eind] then pushmax(eind);
                      if z < max[nind] then pushmax(nind);
                     end;
            z:=(min[nodes]+1) div 2;
            pushmin(ncov);
            pushmin(ecov);
            z:=min[nodes] div 2;
            pushmin(eind);
            pushmin(nind);
            if max[nodes]=3 then
                    begin
                      z:=1;
                      pushmax(nccov);
                    end
                  else if min[nodes] > 3 then
rulea(ncov,nccov,0);
            rulea(nccov,ncov,0);
            rulea(radius,eind,0);
            rulea(eind,radius,0);
            rulea(circ,nodes,0);
            rulea(nodes,girth,0);
            rulea(echr,chr,0);
            rulea(chr,echr,0);
            z:=2*min[ncov]-1;
            pushmin(nodes);
            z:=2*max[ncov];
            if z < max[nodes] then pushmax(nodes);
            z:=2*min[eind];
            pushmin(nodes);
```

```
                z:=2*max[eind]+1;
                if z < max[nodes] then pushmax(nodes);
                girthRcirc:=eq;
                circRnodes:=eq;
              end
            else
              begin
                z:=0;
                if (max[plnar]=0) or
                   (max[reg]=0) or
                   (min[forest]=1) or
                   (max[nodes] < 3) or
                   (max[edges] < min[nodes]) or
                   (min[edges] > max[nodes]) or
                   (max[nconn] < 2) or (min[nconn] > 2) or
                   (max[ncov] < (min[nodes]+1) div 2) or
                   (min[ncov] > (max[nodes]+1) div 2) or
                   (max[nind] < (min[nodes] div 2)) or
                   (min[nind] > (max[nodes] div 2)) or
                   (min[nccov] > max[ecov]) or
                   ((max[nccov] < min[ecov]) and (min[nodes] > 3))
or
                   (min[radius] > max[eind]) or
                   (max[radius] < min[eind]) or
                   (min[xnum] > 0) or
                   (max[bwidth]=1) or
                   (min[bwidth] > 2) or
                   ((min[clique]>2) and (min[nodes] > 3)) or
                   (min[clique]>3) or (min[chr]>3) or
                   (max[arbor]<2) or (min[arbor]>2) or
                   (max[girth]<min[circ]) or
                   (max[chr] < min[echr]) or (min[chr] > max[echr])
or
                   (max[circ]<min[nodes]) or
                   (girthRcirc=lt) or (girthRcirc=ne) or
                   (circRnodes=lt) or (circRnodes=ne) then
                   pushmax(cycle);
              end;
  end;
end;

procedure r050;
(*********************************************************)
(*                                                       *)
(*      Forest <=> E=P-Ncomp                             *)
(*             <=> arbor=1                               *)
(*             <=> girth=circ='undefined'                *)
(*                                                       *)
(*********************************************************)
begin if activerule[50] then
      begin
        rule:=' 50/ ';
```

```
        z:=0;
        if (min[forest] = 1) or (max[arbor] = 1) or
           (min[girth] = infinity) or
           (max[edges] = (min[nodes]-max[ncomp])) then
              begin
                z:=1;
                if z > min[forest] then pushmin(forest);
                pushmax(arbor);
                rulee(nodes,edges,ncomp,0);
                z:=infinity;
                if z > min[girth] then pushmin(girth);
              end
           else
             if (max[forest] = 0) or (min[arbor] > 1) or
                (max[circ] < infinity) or
                (max[girth] < infinity) or
                (min[edges] > (max[nodes]-min[ncomp])) then
                  begin
                    pushmax(forest);
                    z:=2;
                    if z > min[arbor] then pushmin(arbor);
                    z:=max[edges]+max[ncomp];
                    if z < infinity then
                           begin
                             z:=z-1;
                             if z < max[nodes] then
pushmax(nodes);
                           end;
                    if max[ncomp] < infinity then
                            begin
                              z:=min[nodes]-max[ncomp]+1;
                              if z > min[edges] then
pushmin(edges);
                            end;
                    if max[edges] < infinity then
                             begin
                               z:=min[nodes]-max[edges]+1;
                               if z > min[ncomp] then
pushmin(ncomp);
                             end;
                    z:=max[nodes];
                    if z < max[circ] then pushmax(circ);
                  end;
         end;
end;



end.
```