```
unit rules300;

   interface
        uses
           globals,cmmnds1,pusherr,pushStack,ruleAtoF;

    procedure r251; procedure r252; procedure r253; procedure
r254; procedure r255;
     procedure r256; procedure r257; procedure r258; procedure
r259; procedure r260;
     procedure r261; procedure r262; procedure r263; procedure
r264; procedure r265;
     procedure r266; procedure r267; procedure r268; procedure
r269; procedure r270;
     procedure r271; procedure r272; procedure r273; procedure
r274; procedure r275;
     procedure r276; procedure r277; procedure r278; procedure
r279; procedure r280;
     procedure r281; procedure r282; procedure r283; procedure
r284; procedure r285;
     procedure r286; procedure r287; procedure r288; procedure
r289; procedure r290;
    procedure r291; procedure r292; procedure r293; procedure
r294; procedure r295;
     procedure r296; procedure r297; procedure r298; procedure
r299; procedure r300;




implementation



procedure r251;
(*************************************************)
(*                                               *)
(*    if girth is defined then                   *)
(*      E >= (g-1)*(arb-1)**2+(arb-1)            *)
(*                                               *)
(*************************************************)
begin
  if (activerule[251]) and (min[girth] < infinity) then
    begin
      rule:='251/ ';
      if max[edges] < infinity then
        begin
          k:=min[girth]-1;
          z:=trunk(1+(sqrt(4*max[edges]*k+1)-1)/(2*k));
          if z < max[arbor] then pushmax(arbor);
```

```
        end;
      if min[arbor] > 2 then
         begin
           k:=min[arbor]-1;
           z:=(min[girth]-1)*k*k+k;
           if z > min[edges] then pushmin(edges);
           if max[edges] < infinity then
              begin
                z:=(max[edges]-k) div (k*k)+1;
                if z < max[girth] then pushmax(girth);
              end;
         end;
    end;
end;

procedure r252;
(*************************************************)
(*                                               *)
(*  if Nconn >= 2 and girth >= 4 then            *)
(*    eind >= maxdeg*[(girth-4)/4]+k             *)
(*            where                              *)
(*                  0 if g=1 mod 4               *)
(*            k =   1 if g=2 mod 4 or            *)
(*                      (g=3 mod 4 and maxdeg=2) *)
(*                  2 if g=0 or (3 mod 4 and     *)
(*                           not odd cycle)      *)
(*                                               *)
(*************************************************)
begin
  if (activerule[252]) and (max[nconn] > 1) and (max[girth] > 3)
then
    begin
      rule:='252/ ';
      if max[eind] < infinity then
        begin
          z:=4*(max[eind] div min[maxdeg])+4;
          if z < 3 then z:=3;
          if min[nconn] >= 2 then
              begin
                if z < max[girth] then pushmax(girth);
              end
           else
             if min[girth] > z then
                 begin
                   z:=1;
                   pushmax(nconn);
                end;
        end;
      if min[nconn] >= 2 then
         begin
           k:=min[girth] div 4;
           k:=min[girth]-4*k-1;
```

```
            if k=-1 then k:=2
               else if (k=2) and (max[maxdeg]=2) then k:=1;
            z1:=(min[girth]-1) div 4;
            z:=min[maxdeg]*z1+k;
            if z > min[eind] then pushmin(eind);
            if (max[eind] < infinity) and (z1 > 0) then
              begin
                z:=(max[eind]-k) div z1;
                if z < max[maxdeg] then pushmax(maxdeg);
             end;
          end;
    end;
end;

procedure r253;
(*************************************************)
(*                                               *)
(*if Nconn >= 1 then                             *)
(*  maxdeg <= (P-1)/([(girth-4)/2]*(Nconn-1)+1) *)
(*                                               *)
(*************************************************)
begin
  if (activerule[253]) and (min[nconn] > 0) then
    begin
      rule:='253/ ';
      k1:=(min[girth]-3) div 2;
      if k1 > 0 then
        begin
          k:=k1*(min[nconn]-1)+1;
          if max[nodes] < infinity then
            begin
              z:=(max[nodes]-1) div k;
              if z < max[maxdeg] then pushmax(maxdeg);
              z:=((((max[nodes]-1) div min[maxdeg])-1) div k1)+1;
              if z < max[nconn] then pushmax(nconn);
            end;
          z:=min[maxdeg]*k+1;
          if z > min[nodes] then pushmin(nodes);
        end;
    end;
end;

procedure r254;
(*************************************************)
(*                                               *)
(*  if nind=2 then                               *)
(*                    2    if mindeg=1 or P <= 4 *)
(*  nccov <=          3         if 5 <= P <= 10   *)
(*              (mindeg+11)/4  otherwise         *)
(*                                               *)
(*************************************************)
begin
```

```
  if (activerule[254]) and (min[nind] < 3) then
    begin
      rule:='254/ ';
      z:=3;
      if max[nind]=2 then
         begin
           if max[nodes] <= 10 then
               begin
                 if (max[mindeg]=1) or (max[nodes] <= 4) then z:=
2;
                 if z < max[nccov] then pushmax(nccov);
               end
             else
               begin
                 if max[mindeg] < infinity then
                        begin
                          z:=(max[mindeg]+11) div 4;
                          if z < max[nccov] then pushmax(nccov);
                        end;
                 z:=4*min[nccov]-11;
                 if z > min[mindeg] then pushmin(mindeg);
               end;
           end
        else
          if min[nccov] > (max[mindeg]+11) div 4 then
pushmin(nind);
    end;
end;

procedure r255;
(*************************************************)
(*                                               *)
(*  if connected  then                           *)
(*     E >= P+8*thick-13                         *)
(*                                               *)
(*************************************************)
begin
  if (activerule[255]) and (max[connct] = 1) then
    begin
      rule:='255/ ';
      if min[connct] = 1 then
          begin
            z:=min[nodes]+8*min[thick]-13;
            if z > min[edges] then pushmin(edges);
            if max[edges] < infinity then
               begin
                 z:=max[edges]-8*min[thick]+13;
                 if z < max[nodes] then pushmax(nodes);
                 z:=(max[edges]-min[nodes]+13) div 8;
                 if z < max[thick] then pushmax(thick);
               end;
          end
```

```
        else
          if max[edges] < min[nodes]+8*min[thick]-13 then
                         begin
                           z:=0;
                           pushmax(connct);
                         end;
    end;
end;

procedure r256;
(*************************************************)
(*                                               *)
(*  if maxdeg >= 3 then                          *)
(*       diam>= minimum k such that              *)
(*         (deg-1)**k >= (P*(deg-2)+2)/deg       *)
(*                                               *)
(*************************************************)
begin
  if (activerule[256]) and (max[maxdeg] < infinity) and
     (max[maxdeg]=min[maxdeg]) and (max[maxdeg] > 2)  then
    begin
      rule:='256/ ';
      rz:=min[nodes];
      rz:=(rz*(max[maxdeg]-2)+2)/max[maxdeg];
      if rz < infinity then
         begin
           z:=1;
           rhb:=max[maxdeg]-1;
           while rhb < rz do
                begin
                  z:=z+1;
                  rhb:=rhb*(max[maxdeg]-1);
                end;
           if z > min[diam] then pushmin(diam);
         end;
    end;
end;

procedure r257;
(*************************************************)
(*                                               *)
(*  if diam >= 3 then                            *)
(*        eind >= Nconn*CL((diam-2)/2) + K       *)
(*            where                              *)
(*                K = 1  if diam is odd or Nc=1  *)
(*                  = 2  if diam  even and Nc>1  *)
(*                                               *)
(*************************************************)
begin
  if (activerule[257]) and (min[nconn] > 0) and (max[diam] >= 3)
then
    begin
```

```
      rule:='257/ ';
      if max[eind] < infinity then
        begin
          z:=2*((max[eind]-1) div min[nconn])+2;
          if z < 2 then z:=2;
          if z < max[diam] then pushmax(diam);
        end;
      if min[diam] >= 3 then
        begin
          k:=1;
          if (min[nconn] > 1) and (max[diam]=min[diam]) and
             (not(odd(min[diam]))) then k:=2;
          z:=min[nconn]*((min[diam]-1) div 2)+k;
          if z > min[eind] then pushmin(eind);
          if max[eind] < infinity then
             begin
               z:=(max[eind]-k) div ((min[diam]-1) div 2);
               if z < max[nconn] then pushmax(nconn);
             end;
         end;
    end;
end;

procedure r258;
(***********************************************)
(*                                             *)
(*  if (Nconn > 0 and P > 2) or (mindeg > 1)    *)
(*     and (thick <> 3 or P <> 9,10)            *)
(*    then genus >= thick + (E-4*P-1)/6         *)
(*                                             *)
(***********************************************)
begin
  if (activerule[258]) and (((min[nconn] > 0) and (min[nodes] >
2)) or
     (min[mindeg] > 1))
      and ((min[thick] <> 3) or ((max[nodes] <> 9) and
(max[nodes] <> 10))) then
    begin
      rule:='258/ ';
      if max[nodes] < infinity then
        begin
          z:=6*min[thick]+min[edges]-4*max[nodes]+4;
          if z > 5 then
               begin
                 z:=z div 6;
                 if z > min[genus] then pushmin(genus);
               end;
          if max[genus] < infinity then
             begin
               z:=(6*max[genus]-min[edges]+4*max[nodes]+1) div 6;
               if z < max[thick] then pushmax(thick);
               z:=6*(max[genus]-min[thick])+4*max[nodes]+1;
```

```
              if z < max[edges] then pushmax(edges);
            end;
        end;
      if max[genus] < infinity then
          begin
            z:=(min[edges]-6*(max[genus]-min[thick])+2) div 4;
            if z > min[nodes] then pushmin(nodes);
          end;
    end;
end;

procedure r259;
(*********************************)
(*                               *)
(* if girth > 1+2log(P) then     *)
(*            chr <= 3           *)
(*                               *)
(*********************************)
begin
  if (activerule[259]) and (max[nodes] < infinity) and
(max[chr] >= 4) then
    begin
      rule:='259/ ';
      rz:=max[nodes];
      rhb:=1+2*log2(rz);
      if min[girth] > rhb then
         begin
           z:=3;
           pushmax(chr);
         end
       else
         if min[chr] > 3 then
                      begin
                        z:=trunk(rhb);
                        pushmax(girth);
                      end;
    end;
end;

procedure r260;
(*********************************************)
(*                                           *)
(*   let   B=E-P+Ncomp                       *)
(*   genus <= B/2-B/(4*log(B))               *)
(*                                           *)
(*********************************************)
begin
  if activerule[260] then
    begin
      rule:='260/ ';
      if min[genus] = 0 then z1:=0
          else
```

```
            z1:=2*min[genus]+round((sqrt(32*min[genus]+1)+1)/8
+hf);
      if (max[edges] < infinity) and (max[ncomp] < infinity) then
         begin
           z:=max[edges]-min[nodes]+max[ncomp];
           if z > 1 then
             begin
               rz:=z;
               rz:=log2(rz);
               z:=trunk(z/2-z/(4*rz));
             end
            else z:=0;
           if z < max[genus] then pushmax(genus);
           z:=max[edges]+max[ncomp]-z1;
           if z < max[nodes] then pushmax(nodes);
         end;
       if max[edges] < infinity then
         begin
           z:=min[nodes]-max[edges]+z1;
           if z > min[ncomp] then pushmin(ncomp);
         end;
       if max[ncomp] < infinity then
           begin
             z:=min[nodes]-max[ncomp]+z1;
             if z > min[edges] then pushmin(edges);
           end;
    end;
end;

procedure r261;
(*********************************************)
(*                                           *)
(*   if genus <= 2 and girth >= 5            *)
(*         then chr <= 4                     *)
(*                                           *)
(*********************************************)
begin
  if (activerule[261]) and (max[chr] > 4) and (min[genus] <= 2)
     and (max[girth] >= 5) then
    begin
      rule:='261/ ';
      z:=4;
      if (max[genus] <= 2) and (min[girth] > 4)  then
pushmax(chr)
            else
              if min[chr] > 4 then
                 if max[genus] <= 2 then pushmax(girth)
                     else
                       if min[girth] > 4 then
                              begin
                                z:=3;
                                pushmin(genus);
```

```
                              end;
    end;
end;

procedure r262;
(*************************************************)
(*                                               *)
(*  if (genus = 0 and girth >= 4) or             *)
(*     ( "   <= 1 and  "    >= 6) or             *)
(*     ( "   <= 2 and  "    >= 7) or             *)
(*     ( girth >= 9)  then                       *)
(*                      chr <= 3                 *)
(*                                               *)
(*************************************************)
begin
  if (activerule[262]) and (max[chr] > 3) then
    begin
      rule:='262/ ';
      k1:=max[genus];
      k2:=min[girth];
      z:=3;
      if ((k1 = 0) and (k2 > 3)) or
         ((k1 < 2) and (k2 > 5)) or
         ((k1 < 3) and (k2 > 6)) or
         (k2 > 8) then pushmax(chr)
        else
          if min[chr] > 3 then
             begin
               if k1 = 0 then z:=3
                 else if k1 = 1 then z:=5
                   else if k1 = 2 then z:=6
                     else z:=8;
               if z < max[girth] then pushmax(girth);
               if k2 > 6 then z:=3
                 else if k2 > 5 then z:=2
                   else if k2 > 3 then z:=1
                          else z:=0;
               if z > min[genus] then pushmin(genus);
              end;
    end;
end;

procedure r263;
(*****************************************)
(*                                       *)
(*  maxdeg >= (p-1)**(1/radius)          *)
(*                                       *)
(*****************************************)
var rz1:real;
begin
  if (activerule[263]) and (max[radius] < infinity) then
    begin
```

```
      rule:='263/ ';
      rz1:=min[nodes]-1;
      z:=round(root(rz1,max[radius])+hf);
      if z > min[maxdeg] then pushmin(maxdeg);
    end;
end;

procedure r264;
(*********************************************)
(*                                           *)
(*   if G is connected then diam <= 3*dom-1  *)
(*                                           *)
(*********************************************)
begin
  if (activerule[264]) and (max[connct] > 0) then
    begin
      rule:='264/ ';
      if min[connct] = 1 then
            begin
              if max[dom] < infinity then
                  begin
                    z:=3*max[dom]-1;
                    if z < max[diam] then pushmax(diam);
                   end;
              z:=1+min[diam] div 3;
              if z > min[dom] then pushmin(dom);
             end
          else
            if min[diam] > 3*max[dom] -1 then
                begin
                  z:=0;
                  pushmax(connct);
                 end;
    end;
end;

procedure r265;
(*********************************************************)
(*                                                       *)
(*       if connected and maxdeg > 2 then                *)
(*         P <= 1+mindeg*((maxdeg-1)**diam -1)/(maxdeg-2)  *)
(*                                                       *)
(*********************************************************)
var rz:real;
begin
  if (activerule[265]) and (max[maxdeg] > 2) and (min[diam]
< infinity)
      and (max[connct] = 1) then
    begin
      rule:='265/ ';
      if max[maxdeg] < infinity then
           begin
```

```
            k:=max[maxdeg];
            if max[diam] < infinity then
               begin
                 power(k-1,max[diam],z1);
                 if z1 < infinity then
                    begin
                      z1:=(z1-1) div (k-2);
                      z:=1+max[mindeg]*z1;
                      if z < max[nodes] then pushmax(nodes);
                      z:=(min[nodes]-2) div z1+1;
                      if z > min[mindeg] then pushmin(mindeg);
                    end;
               end;
            z:=(min[nodes]-2) div max[mindeg]+1;
            rz:=(k-2)*z+1;
            if rz > 1 then
               begin
                 rhb:=k-1;
                 z:=round(log2(rz)/log2(rhb)+hf);
                 if z > min[diam] then pushmin(diam);
               end;
          end;
      end;
end;

procedure r266;
(*********************************************)
(*                                           *)
(*if 1 < diam < infinity and maxdeg > 2 then   *)
(*   maxdeg >= CL[(p-1)/mindeg]**(1/(diam-1))  *)
(*                                           *)
(*********************************************)
begin
  if (activerule[266]) and (max[diam] < infinity) and
     (max[diam] > 1) and (min[maxdeg] > 2) then
   begin
      rule:='266/ ';
      z:=(min[nodes]-2) div max[mindeg]+1;
      if z > 1 then
           begin
             z:=round(root(z,max[diam]-1)+hf);
             if z > min[maxdeg] then pushmin(maxdeg);
           end;
    end;
end;

procedure r267;
(*************************************)
(*                                   *)
(*  if diam=radius=2 then mindeg > 1    *)
(*                                   *)
(*************************************)
```

```
begin
  if (activerule[267]) and (min[mindeg] < 2) then
    begin
      rule:='267/ ';
      z:=2;
      if (min[radius] = max[diam]) and (min[radius] = 2) then
              pushmin(mindeg);
    end;
end;

procedure r268;
(*************************************)
(*                                   *)
(* if not a forest then              *)
(*      Bwdth >= (girth-1)(arb-2) + 2   *)
(*                                   *)
(*************************************)
begin
  if (activerule[268]) and (min[forest] = 0)
      and (min[girth] < infinity) then
    begin
      rule:='268/ ';
      if max[forest] = 0 then
            begin
              z:=(min[girth]-1)*(min[arbor]-2) + 2;
              if z > min[bwidth] then pushmin(bwidth);
              if max[bwidth] < infinity then
                  begin
                    if min[arbor] > 2 then
                        begin
                          z:=(max[bwidth]-2) div (min[arbor]-2) +
1;
                          if z < max[girth] then pushmax(girth);
                        end;
                    z:=(max[bwidth]-2) div (min[girth]-1) + 2;
                    if z < max[arbor] then pushmax(arbor);
                  end;
            end
          else
            if max[bwidth] < (min[girth]-1)*(min[arbor]-2) + 2
then
                begin
                  z:=1;
                  pushmin(forest);
                end;
    end;
end;

procedure r269;
(*************************************************)
(*                                               *)
(*  if not a forest then                         *)
```

```
(*        Bwidth >= (girth-1)*p/(2nind)-girth + 2  *)
(*                                                 *)
(***************************************************)
begin
  if (activerule[269]) and (min[forest] = 0)
        and (min[girth] < infinity) then
    begin
      rule:='269/ ';
      if max[forest] = 0 then
          begin
            if max[nind] < infinity then
              begin
                z:=(min[girth]-1)*min[nodes] + 2*max[nind]-1;
                z:=z div (2*max[nind]) - min[girth] + 2;
                if z > min[bwidth] then pushmin(bwidth);
                if max[bwidth] < infinity then
                   begin
                     z:=(max[bwidth]+min[girth]-2)*2*max[nind];
                     z:= z div (min[girth]-1);
                     if z < max[nodes] then pushmax(nodes);
                     z:=min[nodes]-2*max[nind];
                     if z > 0 then
                        begin
                          z:=((max[bwidth]-1)*2*max[nind]+z) div
z;
                          if z < max[girth] then pushmax(girth);
                        end;
                    end;
                end;
            if max[bwidth] < infinity then
                begin
                  z:=2*(max[bwidth]+min[girth]-2);
                  z:=(min[nodes]*(min[girth]-1) + z -1) div z;
                  if z > min[nind] then pushmin(nind);
                end;
          end
        else
         if (max[bwidth]+min[girth]-2)*2*max[nind] <
                (min[girth]-1)*min[nodes] then
                  begin
                    z:=0;
                    pushmax(forest);
                   end;
    end;
end;

procedure r270;
(*****************************)
(*                           *)
(* if not a forest then      *)
(*   dom <= p - 2*cir div 3  *)
(*                           *)
```

```
(*****************************)
begin
  if (activerule[270]) and (min[forest] = 0) and (min[circ]
< infinity) then
    begin
      rule:='270/ ';
      z:=min[dom]+(2*min[circ]) div 3;
      if max[forest] = 0 then
        begin
          if z > min[nodes] then pushmin(nodes);
          if max[nodes] < infinity then
                begin
                  z:=max[nodes]-(z-min[dom]);
                  if z < max[dom] then pushmax(dom);
                  z:=3*(max[nodes]-min[dom]) div 2 + 1;
                  if z < max[circ] then pushmax(circ);
                end;
        end
      else if max[nodes] < z then
                begin
                  z:=1;
                  pushmin(forest);
                end;
    end;
end;

procedure r271;
(***********************************************)
(*                                             *)
(*    let k = upper(p/mindeg). Then            *)
(*            circ >= trunc(p/(k-1))           *)
(*                                             *)
(***********************************************)
var k:longint;
begin
  if (activerule[271]) and (min[circ] < infinity) then
    begin
      rule:='271/ ';
      k:=(min[nodes]-1) div min[mindeg] + 1;
      if max[nodes] <= k*min[mindeg] then
           begin
             z:=min[nodes] div (k-1);
             if z > min[circ] then pushmin(circ);
           end;
    end;
end;

procedure r272;
(***********************************************)
(*                                             *)
(*  if diam = 2 then dom <= nconn              *)
(*                                             *)
```

```
(***********************************************)
begin
  if (activerule[272]) and (min[diam] < 3) then
    begin
      rule:='272/ ';
      if max[diam] = 2 then
              begin
                z:=max[nconn];
                if z < max[dom] then pushmax(dom);
                z:=min[dom];
                if z > min[nconn] then pushmin(nconn);
              end
           else
              if min[dom] > max[nconn] then
                     begin
                       z:=3;
                       pushmin(diam);
                     end;
    end;
end;

procedure r273;
(***********************************************)
(*                                             *)
(*      if a tree then rad = upper(diam/2)     *)
(*                                             *)
(***********************************************)
begin
  if (activerule[273]) and (max[tree] = 1) then
    begin
      rule:='273/ ';
      if min[tree] = 1 then
          begin
            if max[diam] < infinity then
                begin
                  z:=(max[diam]+1) div 2;
                  if z < max[radius] then pushmax(radius);
                end;
            z:=2*min[radius]-1;
            if z > min[diam] then pushmin(diam);
          end
        else
          if max[diam] < 2*min[radius]-1 then
               begin
                 z:=0;
                 pushmax(tree);
               end;
    end;
end;

procedure r274;
(***********************************************)
```

```
(*                                                 *)
(* if Ham. then p >= (maxdeg-1)*(girth-2)+2     *)
(*                                                 *)
(*************************************************)
begin
  if (activerule[274]) and (max[hamil] = 1) and (min[girth]
< infinity) then
    begin
      rule:='274/ ';
      z:=(min[maxdeg]-1)*(min[girth]-2)+2;
      if z > min[circ]+1 then z:=min[circ]+1;
      if z > min[nodes] then pushmin(nodes);
      if min[hamil] = 1 then
        begin
          z:=(min[maxdeg]-1)*(min[girth]-2)+2;
          if z > min[nodes] then pushmin(nodes);
          if max[nodes] < infinity then
             begin
               z:=(max[nodes]-2) div (min[girth]-2) + 1;
               if z < max[maxdeg] then pushmax(maxdeg);
               if min[maxdeg] > 1 then
                   begin
                     z:=(max[nodes]-2) div (min[maxdeg]-1) + 2;
                     if z < max[girth] then pushmax(girth);
                   end;
             end;
         end
       else
         if max[nodes] < (min[maxdeg]-1)*(min[girth]-2)+2 then
               begin
                 z:=0;
                 pushmax(hamil);
               end;
    end;
end;

procedure r275;
(*****************************)
(*                           *)
(*  maxdeg <= nind*(chr-1)     *)
(*                           *)
(*****************************)
begin
  if activerule[275] then
    begin
      rule:='275/ ';
      if max[chr] < infinity then
        begin
          if max[nind] < infinity then
             begin
               z:=max[nind]*(max[chr]-1);
               if z < max[maxdeg] then pushmax(maxdeg);
```

```
            end;
          z:=(min[maxdeg]-1) div (max[chr]-1) + 1;
          if z > min[nind] then pushmin(nind);
        end;
      if max[nind] < infinity then
         begin
           z:=(min[maxdeg]-1) div max[nind] + 2;
           if z > min[chr] then pushmin(chr);
         end;
    end;
end;

procedure r276;
(*********************************************)
(*                                           *)
(*   clique >= (p-mindeg-1)/(nccov-1)        *)
(*                                           *)
(*********************************************)
begin
  if activerule[276] then
    begin
      rule:='276/ ';
      if (max[clique] < infinity) and (max[nccov] < infinity)
          and (max[mindeg] < infinity) then
           begin
             z:=max[clique]*(max[nccov]-1)+max[mindeg]+1;
             if z < max[nodes] then pushmax(nodes);
           end;
      if (max[clique] < infinity) and (max[nccov] < infinity)
then
           begin
             z:=min[nodes]-1-max[clique]*(max[nccov]-1);
             if z > min[mindeg] then pushmin(mindeg);
           end;
      if (max[clique] < infinity) and (max[mindeg] < infinity)
then
           begin
             z:=(min[nodes]-max[mindeg]+max[clique]-2) div
max[clique] + 1;
             if z > min[nccov] then pushmin(nccov);
           end;
      if (max[mindeg] < infinity) and (max[nccov] < infinity)
          and (max[nccov] > 1) then
           begin
             z:=(min[nodes]-max[mindeg]-2) div (max[nccov]-1) +
1;
             if z > min[clique] then pushmin(clique);
           end;
    end;
end;

procedure r277;
```

```
(*************************************************)
(*                                               *)
(* if p >= 3 and nconn <= 1 then econn <= maxdeg/2  *)
(*                                               *)
(*************************************************)
begin
  if (activerule[277]) and (min[nconn] < 2) and (min[nodes] >= 3)
then
    begin
      rule:='277/ ';
      if max[nconn] < 2 then
           begin
             if max[maxdeg] < infinity then
                  begin
                    z:= max[maxdeg] div 2;
                    if z < max[econn] then pushmax(econn);
                   end;
             z:=2*min[econn];
             if z > min[maxdeg] then pushmin(maxdeg);
            end
         else
           if min[econn] > max[maxdeg] div 2 then
                begin
                  z:=2;
                  pushmin(nconn);
                end;
    end;
end;

procedure r278;
(*********************************************************)
(*                                                       *)
(*  if connected and maxdeg > 2 then                     *)
(*      p <= 1 + maxdeg[(maxdeg-1)**rad - 1)/(maxdeg-2)]    *)
(*                                                       *)
(*********************************************************)
var rz1:real;
begin
  if (activerule[278]) and (max[radius] > 1) and (max[connct] =
1)
      and (max[maxdeg] < infinity) and (max[maxdeg] > 2) then
     begin
       rule:='278/ ';
       if max[radius] < infinity then
                  begin
                    power(max[maxdeg]-1,max[radius],z);
                    if z < infinity then
                       begin
                         z:=1+max[maxdeg]*(z-1) div
(max[maxdeg]-2);
                         if z < max[nodes] then pushmax(nodes);
                       end;
```

```
                    end;
        rz:=(min[nodes]-1)*(max[maxdeg]-2)/max[maxdeg]+1;
        rz:=log2(rz);
        rz1:=max[maxdeg]-1;
        rz1:=log2(rz1);
        z:=round(rz/rz1+hf);
        if z > min[radius] then pushmin(radius);
    end;
end;

procedure r279;
(*****************************)
(*                           *)
(* if connected then         *)
(*   p >= maxdeg+2*rad-2     *)
(*                           *)
(*****************************)
begin
  if (activerule[279]) and (max[connct] = 1) then
    begin
      rule:='279/ ';
      z:=min[maxdeg]+2*min[radius]-2;
      if min[connct]=1 then
        begin
          if z > min[nodes] then pushmin(nodes);
          z1:=max[nodes];
          if z1 < infinity then
             begin
               z:=z1-2*min[radius]+2;
               if z < max[maxdeg] then pushmax(maxdeg);
               z:=(z1-min[maxdeg]+2) div 2;
               if z < max[radius] then pushmax(radius);
              end;
         end
        else
           if max[nodes] < z then
               begin
                 z:=0;
                 pushmax(connct);
               end;
    end;
end;

procedure r280;
(*********************************************)
(*                                           *)
(*  if a forest then Bw <= (p-2*(Nc-1))/2    *)
(*                                           *)
(*********************************************)
begin
  if (activerule[280]) and (max[forest]=1) then
    begin
```

```
      rule:='280/ ';
      z:=2*(min[bwidth]+min[ncomp]-1);
      if min[forest]=1 then
        begin
          if z > min[nodes] then pushmin(nodes);
          if max[nodes] < infinity then
             begin
               z:=(max[nodes]-2*(min[ncomp]-1)) div 2;
               if z < max[bwidth] then pushmax(bwidth);
               z:=(max[nodes]-2*min[bwidth]) div 2+1;
               if z < max[ncomp] then pushmax(ncomp);
             end;
         end
       else
         if max[nodes] < z then
           begin
             z:=0;
             pushmax(forest);
           end;
    end;
end;

procedure r281;
(*************************************************)
(*                                               *)
(*    nconn > 1, P <=3mindeg, E <=((P+1)mindeg-1)/2 *)
(*             then hamiltonian                  *)
(*                                               *)
(*************************************************)
begin
  if (activerule[281]) and (min[hamil] = 0) and (max[nconn] > 1)
then
    begin
      rule:='281/ ';
      z:=max[nodes];
      if min[nconn] >= 2 then
        begin
          if (max[nodes] <= 3*min[mindeg]) and (max[edges]
< infinity) then
              begin
                z:=(2*max[edges]) div min[mindeg]-1;
                if z < max[circ] then z:=max[circ];
                if z < max[nodes] then pushmax(nodes);
              end;
          if max[hamil] = 0 then
              begin
                z:=(2*min[edges]) div max[mindeg];
                if z > 3*min[mindeg]+1 then z:=3*min[mindeg]+1;
                if z > min[nodes] then pushmin(nodes);
              end;
        end;
      z:=max[nodes];
```

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

procedure r282;
(*************************************************)
(*                                               *)
(* if P is even and not bipartite then not a cycle  *)
```

```
(*                                                  *)
(****************************************************)
begin
  if (activerule[282]) and (max[cycle] = 1) and (min[bipart] = 0)
then
   begin
     rule:='282/ ';
     z:=max[nodes];
     if ((pReven=eq) or ((z=min[nodes]) and (not(odd(z)))))
        and (max[bipart] = 0) then
            begin
              z:=0;
              pushmax(cycle);
            end;
   end;
end;

procedure r283;
(************************************)
(*                                  *)
(* if cubic and nconn = 3 then      *)
(*      circ >= p**(2/3)+1          *)
(*                                  *)
(************************************)
begin
  if (activerule[283]) and (max[nconn] >= 3) and (min[maxdeg] <=
3) then
    begin
      rule:='283/ ';
      z:=min[nodes];
      z:=round(root(z*z,3)+hf)+1;
      if (min[nconn] = max[maxdeg]) and (max[maxdeg] = 3) then
            begin
              if z > min[circ] then pushmin(circ);
              z:=max[circ];
              if z < infinity then
                  begin
                    rz:=z-1;
                    rz:=rz*rz*rz;
                    z:=trunk(sqrt(rz));
                    if z < max[nodes] then pushmax(nodes);
                  end;
             end
          else
            if (max[circ] < z) and (max[maxdeg] = 3) then
                begin
                  z:=2;
                  pushmax(nconn);
                end
             else
               if (min[nconn] = 3) and (max[circ] < z) then
                   begin
```

```
                    z:=4;
                    pushmin(maxdeg);
                  end;
    end;
end;

procedure r284;
(*************************************************)
(*                                               *)
(*   if connected, clique=2, not odd cycle and   *)
(*      not an even path   then                  *)
(*  nind >= p*[1/(maxdeg(maxdeg+1)) + 1/(1+2e/p)]  *)
(*                                               *)
(*************************************************)
var z1,z2:longint;
begin
  if (activerule[284]) and (max[connct]=1) and (min[clique] = 2)
then
    begin
      rule:='284/ ';
      if (min[connct] = 1) and (max[clique] = 2)
         and (min[maxdeg] > 2) then
         begin
           z:=min[nodes];
           if (max[maxdeg] < infinity) and (max[edges]
< infinity) then
             begin
              z:=round(z/(max[maxdeg]*(max[maxdeg]+1))+z*z/(z+2
*max[edges])+hf);
              if z > min[nind] then pushmin(nind);
             end;
           if (max[nind] < infinity) and (max[maxdeg] < infinity)
then
             begin
               rz:=max[nind]-
min[nodes]/(max[maxdeg]*(max[maxdeg]+1));
               z:=round((min[nodes]/rz -1)*min[nodes]/2+hf);
               if z > min[edges] then pushmin(edges);
             end;
           if (max[edges] < infinity) and (max[nind] < infinity)
then
              begin
                rz:=2*max[edges]/min[nodes]+1;
                rz:=min[nodes]*rz/(max[nind]*rz-min[nodes])+0.25;
                if rz >= 0.0 then
                  begin
                    z:=round(sqrt(rz)-0.5+hf);
                    if z > min[maxdeg] then pushmin(maxdeg);
                  end;
              end;
           z:=max[maxdeg];
           if (z < infinity) and (max[nind] < infinity)
```

```
              and (max[edges] < infinity) then
                begin
                  rlb:=z*z+z;
                  rhb:=(max[nind]*rlb-2*max[edges])/(2*rlb+2);
                  rz:=rhb*rhb+2*max[nind]*max[edges]*rlb/(rlb+1);
                  z:=trunk(rhb+sqrt(rz));
                  if z < max[nodes] then pushmax(nodes);
                end;
           end
          else
            begin
              z1:=max[maxdeg];
              z2:=max[edges];
              if (z1 < infinity) and (z2 < infinity) then
                  begin
                    z:=min[nodes];
                    z:= round(z/(z1*z1+z1)+z/(1+2*z2/z)+hf);
                    if max[nind] < z then
                         begin
                           z:=3;
                           if (min[connct] = 1) and (min[maxdeg] >
2)
                             then pushmin(clique)
                             else
                               begin
                                 z:=0;
                                 if (max[clique] = 2) and
(min[maxdeg] > 2)
                                    then pushmax(connct)
                                    else
                                      begin
                                        z:=2;
                                        if (min[connct]=1) and
(max[clique] = 2)
                                           then pushmax(maxdeg);
                                      end;
                               end;
                          end;
                 end;
            end;
    end;
end;

procedure r285;
(*********************************************)
(*                                           *)
(*   eccov <= nccov + p(maxdeg+1-p/nccov)/2  *)
(*                                           *)
(*********************************************)
var rk :real;
begin
  if (activerule[285]) and (max[nodes] = min[nodes])
```

```
     and (max[nodes] < infinity) then
    begin
      rule:='285/ ';
      rk:=max[nodes];
      rz:=max[nccov];
      if rz < infinity then
          begin
            if max[maxdeg] < infinity then
               begin
                 rz:=rz+rk*(max[maxdeg]*rz+rz-rk)/(2*rz);
                 if rz < infinity then
                     begin
                       z:=trunk(rz);
                       if z < max[eccov] then pushmax(eccov);
                     end;
               end;
            z:=round(2*(min[eccov]-
max[nccov])/rk+rk/max[nccov]+hf)-1;
            if z > min[maxdeg] then pushmin(maxdeg);
          end;
       z:=max[maxdeg];
       if z < infinity then
          begin
            rz:=2*min[eccov]-rk*(z+1);
            rk:=rz*rz+8*rk*rk;
            if rk < infinity then
                 begin
                   k:=trunk(rk);
                   z:=round((rz+sqrt(rk))/4+hf);
                   if z > min[nccov] then pushmin(nccov);
                 end;
          end;
    end;
end;

procedure r286;
(*********************************************)
(*                                           *)
(*  E >= (maxdeg+(chr-1)**2+(p-chr)*mindeg)/2   *)
(*                                           *)
(*********************************************)
var z1:longint;
begin
  if (activerule[286]) and (max[chr] = min[chr])
      and (max[chr] < infinity) then
    begin
      rule:='286/ ';
      z:=max[chr];
      z:=(min[maxdeg]+(z-1)*(z-1)+(min[nodes]-z)*min[mindeg]+1)
div 2;
      if z > min[edges] then pushmin(edges);
      if max[edges] < infinity then
```

```
          begin
            z:=max[chr]-1;
            z:=2*max[edges]-z*z-(min[nodes]-z-1)*min[mindeg];
            if z < max[maxdeg] then pushmax(maxdeg);
            z:=max[chr]-1;
            z1:=2*max[edges]-z*z-min[maxdeg];
            z:=min[nodes]-max[chr];
            if z > 0 then
                 begin
                   z:=z1 div z;
                   if z < max[mindeg] then pushmax(mindeg);
                 end;
            z:=z1 div min[mindeg]+max[chr];
            if z < max[nodes] then pushmax(nodes);
          end;
    end;
end;

procedure r287;
(*********************************************)
(*                                           *)
(*  if plnar then mindeg <= nind + 2         *)
(*                                           *)
(*********************************************)
begin
  if (activerule[287]) and (max[plnar] = 1) then
    begin
      rule:='287/ ';
      if min[plnar] = 1 then
          begin
            z:=max[nind]+2;
            if z < max[mindeg] then pushmax(mindeg);
            z:=min[mindeg]-2;
            if z > min[nind] then pushmin(nind);
          end
        else
          if min[mindeg] > max[nind]+2 then
                begin
                  z:=0;
                  pushmax(plnar);
                end;
    end;
end;

procedure r288;
(*********************************************)
(*                                           *)
(*  if clique=di=2 , reg., not bipartite, and   *)
(*       maxdeg=2k or 3k then                *)
(*            p >= 3maxdeg - k               *)
(*                                           *)
(*********************************************)
```

```
begin
  if (activerule[288]) and (max[maxdeg] = min[maxdeg]) and
(min[clique] = 2)
      and (min[bipart] = 0) and (max[reg] = 1) and (min[diam] <=
2) then
    begin
      rule:='288/ ';
      z:=max[maxdeg];
      if not(odd(z)) then k:=z div 2
            else
              if z=3*(z div 3) then k:=z div 3
                  else k:=0;
      if (k > 0) and (max[diam] = 2) and (min[diam]=2) and
(min[reg] = 1)
          and (max[bipart] = 0) and (max[clique]=2) then
          begin
            z:=3*z-k;
            if z > min[nodes] then pushmin(nodes);
          end
         else
           if (k > 0) and (max[nodes] < 3*z-k) then
                begin
                  z:=1;
                  if (min[diam]=max[diam]) and (min[diam]=2)
                     and (min[reg]=1) and (max[clique]=2) then
                           pushmin(bipart)
                     else
                       if (min[diam]=max[diam]) and (min[diam]=2)
                          and (max[bipart]=0) and (max[clique]=2)
then
                             begin
                              z:=0;
                              pushmax(reg);
                             end
                           else
                             if (min[reg]=1) and (max[bipart]=0)
                                 and (max[clique]=2) then
                                 begin
                                   z:=3;
                                   if min[diam]=2 then
pushmin(diam);
                                  end
                             else
                               if (min[reg]=1) and (max[bipart]=
0)
                                  and (max[diam]=2) and
(min[diam]=2) then
                                      begin
                                        z:=3;
                                        pushmin(clique);
                                      end;
                  end;
```

```
    end;
end;

procedure r289;
(*********************************************)
(*                                           *)
(*  if connected, not ham, (P-2)/2 <= circ, and *)
(*       not a tree then  dom <=(2P-circ)/3     *)
(*                                           *)
(*********************************************)
begin
  if (activerule[289]) and (max[connct] = 1) and (min[tree] = 0)
     and (min[circ] < infinity) and (min[hamil] = 0) then
    begin
      rule:='289/ ';
      z1:=max[nodes];
      if (min[connct]=1) and (max[hamil]=0) and (max[tree] = 0)
then
         begin
           z:=(3*min[dom]+min[circ]+1) div 2;
           if 2*min[circ]+3 < z then z:=2*min[circ]+3;
           if z > min[nodes] then pushmin(nodes);
           if z1 < infinity then
             begin
               z:=2*z1-3*min[dom];
               if z < (z1-3) div 2 then z:=(z1-3) div 2;
               if z < max[circ] then pushmax(circ);
             end;
         end;
       z:=(z1-1) div 2;
       if z1 < infinity then
         if (min[connct]=1) and (max[hamil]=0) and (max[tree] = 0)
            and (z <= min[circ]) then
          begin
            z:=(2*z1-min[circ]) div 3;
            if z < max[dom] then pushmax(dom);
          end
         else
           if 3*min[dom] > 2*z-min[circ] then
              begin
                if (z <= min[circ]) and (max[hamil]=0)
                   and (max[tree] = 0) then
                     begin
                       z:=0;
                       pushmax(connct);
                     end
                  else
                    if (min[connct]=1) and (max[tree] = 0) and
                       (z <= min[circ]) then
                               begin
                                 z:=1;
                                 pushmin(hamil);
```

```
                              end
                        else
                          if (max[hamil] = 0) and (min[connct] =
1) and
                             (z <= min[circ]) then
                                 begin
                                   z:=1;
                                   pushmin(tree);
                                 end;
                end;
    end;
end;

procedure r290;
(*************************************************)
(*                                               *)
(* if connected and dom>= 3 then                 *)
(*      dom <=(2p+1-sqrt(8e+1))/2                *)
(*                                               *)
(*************************************************)
begin
  if (activerule[290]) and (max[connct] = 1) and (max[dom] >= 3)
then
    begin
      rule:='290/ ';
      if (min[connct] = 1) and (min[dom] > 2) then
        begin
          if max[nodes] < infinity then
            begin
              z:=8*min[edges]+1;
              z:=trunk((2*max[nodes]+1-sqrt(z))/2);
              if z < max[dom] then pushmax(dom);
              z:=max[nodes]-min[dom];
              z:=(z+1)*z div 2;
              if z < max[edges] then pushmax(edges);
            end;
           z:=8*min[edges]+1;
           z:=round((2*min[dom]-1+sqrt(z))/2+hf);
           if z < min[nodes] then pushmin(nodes);
         end
       else
         if max[nodes] < infinity then
           begin
             z:=max[nodes]-min[dom];
             if 2*min[edges] > (z+1)*z then
                begin
                  if min[connct] = 1 then
                      begin
                        z:=2;
                        pushmax(dom);
                      end
                   else
```

```
                    begin
                      z:=0;
                      pushmax(connct);
                    end;
                end;
            end;
    end;
end;

procedure r291;
(*********************************************)
(*                                           *)
(*  dom <= P(1+ln(mindeg))/(mindeg+1)        *)
(*                                           *)
(*note: ln(x)=ln(2)*log2(x)~.6931471806*log2(x) *)
(*                                           *)
(*********************************************)
function bound(min:longint):longint;
begin
   if min < 1 then bound:=infinity
      else
        if min = 1 then bound:=z1 div 2
           else
             begin
               rz:=1+0.693147806*log2(min);
               bound:=trunk(z1*rz/(min+1));
             end;
end;

begin
  if activerule[291] then
    begin
      rule:='291/ ';
      z1:=max[nodes];
      if min[mindeg] >= 2 then
         begin
           rz:=1+0.693147806*log2(min[mindeg]);
           if z1 < infinity then
             begin
               z:=bound(min[mindeg]);
               if z < max[dom] then pushmax(dom);
             end;
            z:=round(min[dom]*(min[mindeg]+1)/rz+hf);
            if z > min[nodes] then pushmin(nodes);
         end;
      z:=max[mindeg];
      if z < infinity then
          begin
            if z >= z1 then z:=z1-1;
            while (min[dom] > bound(z)) and (z >= min[mindeg]) do
z:=z-1;
            if z < max[mindeg] then pushmax(mindeg);
```

```
          end;
    end;
end;

procedure r292;
(*************************************************)
(*                                               *)
(*  if connected and not a tree then             *)
(*       g<= 2*diam + 1 with equality iff        *)
(*            i)  G ~ Kp , p>2                   *)
(*           ii)  G ~ Cg , or                    *)
(*          iii)  g=5 and G is a Moore Graph     *)
(*                                               *)
(*************************************************)
var boole:boolean;
var k,z1:longint;
begin
  if (activerule[292]) and (max[girth] < infinity) and
     (max[diam] < infinity)  then
    begin
      rule:='292/ ';
      z:=max[mindeg];
      z1:=min[maxdeg];
      k:=0;
       if (((min[maxdeg] <= 3) and (max[mindeg] >= 2)) or
           ((min[maxdeg] <= 7) and (max[mindeg] >= 7)) or
           ((min[maxdeg] <= 57) and (max[mindeg] >= 57))) then
boole:=true
                  else boole:=false;
      if (max[compl] = 1) or (max[cycle] = 1) or
         ((min[girth] <= 5) and (max[girth] >= 5) and
          (min[diam] <= 2) and (max[reg] = 1) and
          (min[nodes] <= z*z+1) and (max[nodes] >= z1*z1+1) and
boole) then k:=1;
      z:=2*max[diam]+k;
      if z < max[girth] then pushmax(girth);
      z:=(min[girth]-k+1) div 2;
      if z > min[diam] then pushmin(diam);
      if min[girth]=2*max[diam]+1 then
           begin
             z:=1;
             if min[reg] = 0 then pushmin(reg);
             if max[girth] < 5 then pushmin(compl)
                  else
                    if min[girth] > 5 then pushmin(cycle)
                       else
                         begin
                           if min[mindeg] <= 3 then z:=2
                               else if min[mindeg] <= 7 then z:=7
                                      else z:=57;
                           k:=round(sqrt(min[nodes]-1)+hf);
                           if k > z then z:=k;
```

```
                          if z > min[mindeg] then
pushmin(mindeg);
                          if max[maxdeg] >= 57 then z:=57
                               else if max[maxdeg] >= 7 then z:=
7
                                          else z:=3;
                          k:=trunk(sqrt(max[nodes]-1));
                          if k < z then z:=k;
                          if z < max[maxdeg] then
pushmax(maxdeg);
                          z:=max[maxdeg]*max[maxdeg]+1;
                          if z < max[nodes] then pushmax(nodes);
                          z:=min[mindeg]*min[mindeg]+1;
                          if z > min[nodes] then pushmin(nodes);
                        end;
          end;
    end;
end;

procedure r293;
(************************************)
(*                                  *)
(* if connected and maxdeg=2 then   *)
(*  mindeg*diam-mindeg+2 <= P       *)
(*                   <= mindeg*diam+1 *)
(*                                  *)
(************************************)
begin
  if (activerule[293]) and (min[maxdeg] < 3)
      and (min[diam] < infinity) then
    begin
      rule:='293/ ';
      z:=3;
      if max[maxdeg] = 2 then
           begin
             if max[diam] < infinity then
                  begin
                    z:= max[mindeg]*max[diam]+1;
                    if z < max[nodes] then pushmax(nodes);
                    z:=(min[nodes]-2) div max[diam]+1;
                    if z > min[mindeg] then pushmin(mindeg);
                   end;
             z:=(min[nodes]-2) div max[mindeg]+1;
             if z > min[diam] then pushmin(diam);
                 z:=min[mindeg]*(min[diam]-1)+2;
                 if z > min[nodes] then pushmin(nodes);
                 if max[nodes] < infinity then
                       begin
                            z:=(max[nodes]-2) div (min[diam]-1);
                              if z < max[mindeg] then
pushmax(mindeg);
                              z:= (max[nodes]-2) div min[mindeg]+
```

```
1;
                              if z < max[diam] then pushmax(diam);
                          end;
          end
        else
          if (max[diam] < infinity) and (max[mindeg] < infinity)
then
          if (min[nodes] > max[mindeg]*max[diam]+1) or
                (max[nodes] < min[mindeg]*(min[diam]-1)+2) then
pushmin(maxdeg);
    end;
end;

procedure r294;
(*************************************************)
(*                                               *)
(*  eccov <= (p-clique+2)**2 div 4               *)
(*                                               *)
(*************************************************)
begin
  if activerule[294] then
    begin
      rule:='294/ ';
      z1:=round(2*sqrt(min[eccov])+hf);
      if max[nodes] < infinity then
         begin
           z:=max[nodes]-min[clique]+2;
           z:=z*z div 4;
           if z < max[eccov] then pushmax(eccov);
           z:=max[nodes]+2-z1;
           if z < max[clique] then pushmax(clique);
         end;
      z:=min[clique]-2+z1;
      if z > min[nodes] then pushmin(nodes);
    end;
end;

procedure r295;
(*************************************************)
(*                                               *)
(*    e <= max{ eind(2*eind+1),                  *)
(*                eind*p-eind(eind+1)/2 }        *)
(*                                               *)
(*************************************************)
var z2:longint;
begin
 if activerule[295] then
  if (max[nodes] < infinity) and (max[eind] < infinity) then
    begin
      rule:='295/ ';
      z1:=max[eind];
      z2:=z1*max[nodes]-(z1*(z1+1)) div 2;
```

```
      z1:=z1*(2*z1+1);
      if z1 < z2 then z:=z2
                 else z:=z1;
      if z < max[edges] then pushmax(edges);
      if z2 < min[edges] then
             begin
               rhb:=(-0.5+sqrt(2*min[edges]+0.25))/2;
               z:=round(rhb+hf);
               if z > min[eind] then pushmin(eind);
             end
           else
             if z1 < min[edges] then
                begin
                  z1:=max[eind];
                  z:=(min[edges]+(z1*(z1+1) div 2)+z1-1) div z1;
                  if z > min[nodes] then pushmin(nodes);
                  z:=2*max[nodes]-1;
                  z1:=z*z-8*min[edges];
                  if z1 >= 0 then
                       begin
                         z:=round((z-sqrt(z1))/2+hf);
                         if z > min[eind] then pushmin(eind);
                       end;
                  end;
    end;
end;

procedure r296;
(*********************************************)
(*                                           *)
(*  if mindeg=maxdeg=3 then nconn=econn      *)
(*                                           *)
(*********************************************)
begin
  if (activerule[296]) and (min[mindeg] <= 3) and (max[mindeg] >=
3)
     and (min[maxdeg] <= 3) and (max[maxdeg] >= 3) then
    begin
      rule:='296/ ';
      if (min[mindeg]=max[maxdeg]) and (min[mindeg]=3) then
           begin
             z:=max[nconn];
             if z < max[econn] then pushmax(econn);
             z:=min[econn];
             if z > min[nconn] then pushmin(nconn);
           end
         else
           if max[nconn] < min[econn] then
                begin
                  if min[mindeg]=3 then
                        begin
                          z:=4;
```

```
                          pushmin(maxdeg);
                        end
                      else
                        if max[maxdeg]=3 then
                             begin
                               z:=2;
                               pushmax(mindeg);
                             end;
                  end;
    end;
end;


procedure r297;
(*************************************************)
(*                                               *)
(* if nconn > 2, genus = 0, not hamil            *)
(*           then p > 10                         *)
(*                                               *)
(*************************************************)
begin
  if (activerule[297]) and
     (min[nodes] < 11) and (min[hamil]=0) and (min[genus] = 0)
     and (max[nconn] > 2) then
    begin
      rule:='297/ ';
      if (min[nconn] > 2) and (max[genus] = 0) and (max[hamil] =
0) then
           begin
             z:=11;
             pushmin(nodes);
           end
        else
         if (max[nodes] < 11) and (min[nconn] > 2) and
(max[genus] = 0) then
             begin
               z:=1;
               pushmin(hamil);
             end
           else
             if (max[hamil] = 0) and (max[nodes] < 11) and
(min[nconn] > 2) then
                    begin
                      z:=1;
                      pushmin(genus);
                    end
                  else
                    if (max[genus] = 0) and
                       (max[hamil] = 0) and (max[nodes] < 11)
then
                            begin
                              z:=2;
```

```
                                pushmax(nconn);
                               end;
    end;
end;

procedure r298;
(*************************************************)
(*                                               *)
(* if cubic,nconn>=2,not plnar,and not bipart  *)
(*               then p >= 8                     *)
(*                                               *)
(*************************************************)
begin
  if (activerule[298]) and (min[nodes] < 8) and (max[nconn] >= 2)
and
     (min[plnar] = 0) and (min[mindeg] <= 3) and (max[mindeg] >=
3)
     and (min[maxdeg] <= 3) and (max[maxdeg] >= 3) and
(min[bipart] = 0) then
    begin
      rule:='298/ ';
      z:=1;
      if (min[mindeg]=max[maxdeg]) and (min[mindeg]=3) and
(min[nconn]>1)
         and (max[plnar]=0) and (max[bipart]=0)  then
          begin
            z:=8;
            pushmin(nodes);
          end
        else
         if (max[nodes] < 8) and (min[mindeg]=max[maxdeg]) and
(min[mindeg]=3)
            and (min[nconn]>1) and (max[plnar]=0) then
pushmin(bipart)
          else
           if (max[bipart]=0) and (max[nodes] < 8) and
              (min[mindeg]=max[maxdeg]) and (min[mindeg]=3) and
              (min[nconn]>1) then pushmin(plnar)
             else
               if (max[plnar]=0) and (max[bipart]=0) and
(max[nodes] < 8)
                  and (max[maxdeg]=3) then
                   begin
                     z:=1;
                     pushmax(nconn);
                   end
                else
                  if (min[nconn]>1) and (max[plnar]=0) and
(max[bipart]=0)
                     and (max[nodes] < 8) then
                     begin
                       if min[mindeg]=3 then
```

```
                          begin
                            z:=4;
                            pushmin(maxdeg);
                          end
                        else
                          if max[maxdeg]=3 then
                            begin
                              z:=2;
                              pushmax(mindeg);
                            end;
                      end;
    end;
end;

procedure r299;
(*************************************************)
(*                                               *)
(*  if cubic and nconn=1 then p >= 10            *)
(*                                               *)
(*************************************************)
begin
  if (activerule[299]) and (min[nodes] < 10) then
    begin
      rule:='299/ ';
      if (min[mindeg]=max[maxdeg]) and (min[mindeg]=3) and
         (min[nconn]=max[nconn]) and (min[nconn]=1) then
          begin
            z:=10;
            pushmin(nodes);
          end
       else
         if (max[nodes] < 10) and (min[mindeg]=max[maxdeg]) and
            (min[mindeg]=3) then
             begin
               if min[nconn]=1 then
                         begin
                           z:=2;
                           pushmin(nconn);
                         end
                       else
                         if max[nconn]=1 then
                               begin
                                 z:=0;
                                 pushmax(nconn);
                               end;
              end
            else
              if (min[nconn]=max[nconn]) and (min[nconn]=1) and
                 (max[nodes] < 10) then
                  begin
                    if min[mindeg]=3 then
                           begin
```

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

procedure r300;
(*********************************************)
(*                                           *)
(* if cubic,nconn=1, not plnar then p>=12      *)
(*                                           *)
(*********************************************)
begin
  if (activerule[300]) and (min[nodes] < 12) then
    begin
      rule:='300/ ';
      if (min[mindeg]=max[maxdeg]) and (min[mindeg]=3) and
         (min[nconn]=max[nconn]) and (min[nconn]=1) and
         (max[plnar]=0) then
            begin
              z:=12;
              if z > min[nodes] then pushmin(nodes);
            end
         else
           if (max[nodes] < 12) and (min[mindeg]=max[maxdeg]) and
              (min[mindeg]=3) and (min[nconn]=max[nconn]) and
              (min[nconn]=1) then
               begin
                 z:=1;
                 if z > min[plnar] then pushmin(plnar);
               end
         else
           if (max[plnar]=0) and (max[nodes] < 12) and
              (min[mindeg]=max[maxdeg]) and (min[mindeg]=3) then
              begin
                if min[nconn]=1 then
                    begin
                      z:=2;
                      pushmin(nconn);
                    end
                  else
                    if max[nconn]=1 then
                             begin
```

```
                              z:=0;
                              pushmax(nconn);
                            end;
                end
              else
                if (min[nconn]=max[nconn]) and (min[nconn]=1) and
                   (max[plnar]=0) and (max[nodes] < 12) then
                     begin
                       if min[mindeg]=3 then
                          begin
                            z:=4;
                            if z > min[maxdeg] then
pushmin(maxdeg);
                          end
                        else
                          if max[maxdeg]=3 then
                             begin
                               z:=2;
                               if z < max[mindeg] then
pushmax(mindeg);
                             end;
                      end;
    end;
end;


end.
```