```
  unit rules100;
     interface
         uses
                globals,cmmnds1,pusherr,pushStack,
                 ruleAtoF;



        procedure r051; procedure r052; procedure r053; procedure
r054; procedure r055;
           procedure r056; procedure r057; procedure r058;
procedure r059; procedure r060;
           procedure r061; procedure r062; procedure r063;
procedure r064; procedure r065;
           procedure r066; procedure r067; procedure r068;
procedure r069; procedure r070;
           procedure r071; procedure r072; procedure r073;
procedure r075; procedure r076;
           procedure r077; procedure r078; procedure r079;
procedure r080; procedure r081;
           procedure r082; procedure r083; procedure r084;
procedure r085; procedure r086;
           procedure r087; procedure r088; procedure r089;
procedure r090; procedure r091;
           procedure r092; procedure r093; procedure r094;
procedure r095; procedure r096;
           procedure r097; procedure r098; procedure r099;
procedure r100;

implementation


procedure r051;
(*************************)
(*                       *)
(*   chr/2 <= arbor      *)
(*                       *)
(*************************)
begin if activerule[51] then
        begin
          rule:=' 51/ ';
          z:=2*max[arbor];
          if z < max[chr] then pushmax(chr);
          z:=(min[chr]+1) div 2;
          if z > min[arbor] then pushmin(arbor);
        end;
end;

procedure r052;
(*********************************************************)
(*                                                       *)
```

```
(*            arbor <= 1 + spectr/2                          *)
(*                                                           *)
(*    if (connected and not a cycle and                      *)
(*     not  Kp with p odd) or (clique <= maxdeg and          *)
(*         (maxdeg >= 4 and reg) then                        *)
(*           arbor <= (1+maxdeg)/2                           *)
(*                                                           *)
(***********************************************************)
begin if activerule[52] then
  begin
    rule:=' 52/ ';
    if lammax < infinity then
              begin
                z:=1+trunk(lammax/2);
                if z < max[arbor] then pushmax(arbor);
              end;
    rz:=2*min[arbor]-2;
    if rz > lammin then pushlammin;
    if ((min[connct]=1) and (max[cycle]=0) and
       ((max[compl]=0) or (pReven=eq) or ((max[nodes]=min[nodes])
and
       (not(odd(max[nodes])))))) or
       ((max[clique] <= min[maxdeg]) and (min[maxdeg] > 3)
        and ((min[mindeg]=max[maxdeg])or (min[reg]=0))) then
        begin
          z:=2*min[arbor]-1;
          if z > min[maxdeg] then pushmin(maxdeg);
          if max[maxdeg] < infinity then
                begin
                  z:=(1+max[maxdeg]) div 2;
                  if z < max[arbor] then pushmax(arbor);
                end;
        end;
  end;
end;

procedure r053;
(*********************************************************)
(*                                                       *)
(*   arbor <= chr-trunc(chr/(1 + P/(l*chr)))             *)
(*          where l=(girth-1) div 2                      *)
(*                                                       *)
(*********************************************************)
begin if (activerule[53]) and (min[girth] < infinity) then
begin
  rule:=' 53/ ';
  z1:=(min[girth]-1) div 2;
  z:=max[chr];
  k:=min[arbor]-1;
  if z < infinity then
    begin
      if max[nodes] < infinity then
```

```
            begin
              z:=z-trunk(z/(1+max[nodes]/(z1*z)));
              if z < max[arbor] then pushmax(arbor);
              if k > 1 then
                 begin
                   z:=round(k*max[nodes]/(max[nodes]-z1*k)+hf);
                   if z > min[chr] then pushmin(chr);
                   z:=(k*z1*max[chr]-1) div (max[chr]-k)+1;
                   if z > min[nodes] then pushmin(nodes);
                   z:=(2*max[nodes]*(max[chr]-k)) div
(k*max[chr])+2;
                   if z < max[girth] then pushmax(girth);
                 end;
            end
         else
           begin
             rulea(arbor,chr,0);
             z:=(k*z1*max[chr]-1) div (max[chr]-k)+1;
             if z > min[nodes] then pushmin(nodes);
           end;
    end
  else
    if max[nodes] < infinity then
           begin
             z:=max[nodes]-z1*k;
             if z > 0 then
               begin
                 z:=(k*max[nodes]-1) div z+1;
                 if z > min[chr] then pushmin(chr);
               end;
           end
        else
          begin
            z:=k+1;
            if z > min[chr] then pushmin(chr);
          end;
  end;
end;
procedure r054;
(*********************************)
(*                               *)
(*   planar ==> mindeg <= 5      *)
(*              clique <= 4      *)
(*              arbor <= 3       *)
(*              Xnum = 0         *)
(*                               *)
(*********************************)
begin if (activerule[54]) and (max[plnar] = 1) then
begin
  rule:=' 54/ ';
  z:=0;
  if min[plnar] =1 then
```

```
     begin
       if z < max[xnum] then pushmax(xnum);
       z:=5;
       if z < max[mindeg] then pushmax(mindeg);
       z:=4;
       if z < max[clique] then pushmax(clique);
       z:=3;
       if z < max[arbor] then pushmax(arbor);
     end
    else
      if (min[mindeg] > 5) or
         (min[clique] > 4) or
         (min[arbor] > 3) or
         (min[xnum] > 0) then
              if z < max[plnar] then pushmax(plnar);
  end;
end;

procedure r055;
(*******************************)
(*                             *)
(*   2E >= maxdeg+(P-1)mindeg  *)
(*                             *)
(*******************************)
begin if activerule[55] then
     begin
       rule:=' 55/ ';
       z:=(min[maxdeg]+(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]-(min[nodes]-1)*min[mindeg];
                    if z < max[maxdeg] then pushmax(maxdeg);
                    z:=(2*max[edges]-min[maxdeg]) div
(min[nodes]-1);
                    z1:=(2*max[edges]) div min[nodes];
                    if z1 < z then z:=z1;
                    if z < max[mindeg] then pushmax(mindeg);
                    z:=(2*max[edges]-min[maxdeg]) div
min[mindeg]+1;
                    if z < max[nodes] then pushmax(nodes);
                  end;
      end;
end;

procedure r056;
(*************************************)
(*                                   *)
(*       2E <= (P-1)maxdeg+mindeg    *)
(*                                   *)
(*************************************)
begin if activerule[56] then
```

```
   begin
     rule:=' 56/ ';
     if (max[nodes] < infinity) and (max[maxdeg] < infinity) then
         begin
           z:=2*min[edges]-(max[nodes]-1)*max[maxdeg];
           if z > min[mindeg] then pushmin(mindeg);
           if max[mindeg] < infinity then
                  begin
                    z:=((max[nodes]-1)*max[maxdeg]+max[mindeg])
div 2;
                    if z < max[edges] then pushmax(edges);
                    z:=(2*min[edges]-max[mindeg]-1) div
(max[nodes]-1)+1;
                    if z > min[maxdeg] then pushmin(maxdeg);
                  end;
         end;
     if max[maxdeg] < infinity then
            begin
              if max[mindeg] > max[maxdeg] then z:=max[maxdeg]
                                           else z:=max[mindeg];
              z:=(2*min[edges]-z-1) div max[maxdeg]+2;
              if z > min[nodes] then pushmin(nodes);
            end;
    end;
end;

procedure r057;
(*****************************************)
(*                                       *)
(*  d-regular and d is odd  ==> P is even  *)
(*                                       *)
(*****************************************)
begin if (activerule[57]) and (max[reg] = 1) then
begin
  rule:=' 57/ ';
  if min[reg]=1 then
    begin
      if (odd(min[mindeg])) and (min[mindeg]=max[mindeg]) then
             begin
               z:=min[nodes]+1;
               if not(odd(z)) then pushmin(nodes);
               z:=max[nodes]-1;
               if (not(odd(z))) and (max[nodes] < infinity) then
pushmax(nodes);
               pReven:=eq;
               pRodd:=ne;
             end
       else
         if ((max[nodes]=min[nodes]) and (odd(max[nodes]))) or
(pRodd=eq)  then
                  begin
                    z:=min[mindeg]+1;
```

```
                    if not(odd(z)) then pushmin(mindeg);
                    z:=max[maxdeg]-1;
                    if not(odd(z)) then pushmax(maxdeg);
                    pRodd:=eq;
                    pReven:=ne;
                  end;
      end
    else
      if (((max[nodes]=min[nodes]) and (odd(max[nodes]))) or
(pRodd=eq)) and
         (((odd(min[mindeg])) and (min[mindeg]=max[mindeg])) or
         ((odd(min[maxdeg])) and (min[maxdeg] = max[maxdeg])))
then
          begin
            z:=0;
            pushmax(reg);
          end;
  end;
end;

procedure r058;
(*********************************)
(*                               *)
(*   clique > P/(P-spectr) - 1/3   *)
(*                               *)
(*********************************)
begin if activerule[58] then
    begin
      rule:=' 58/ ';
      z:=max[nodes];
      if z < infinity then
         begin
           z:=round(z/(z-lammin)-1/3+hf);
           if z > min[clique] then pushmin(clique);
           z:=max[clique];
           if z < infinity then
             begin
               rz:=max[nodes]*(z-2/3)/(z+1/3);
               if rz < lammax then pushlammax;
             end;
          end;
      z:=max[clique];
      if z < infinity then
            begin
              z:=round(lammin*(z+1/3)/(z-2/3)+hf);
              if z > min[nodes] then pushmin(nodes);
            end;
    end;
end;

procedure r059;
(************************************************)
```

```
(*                                                  *)
(*  Xnum <= (P/2)((P-1)/2)((P-2)/2)((P-3)/2)/4      *)
(*                                                  *)
(*    equality if complete and P <= 10              *)
(*                                                  *)
(****************************************************)
begin if activerule[59] then
    begin
      rule:=' 59/ ';
      if max[nodes] < infinity then
          begin
            z1:=max[nodes];
            z:=z1 div 2;
            z:=z*((z1-1) div 2);
            if z < infinity then z:=z*((z1-2) div 2);
            if z < infinity then z:=z*((z1-3) div 2) div 4;
            if z < max[xnum] then pushmax(xnum);
          end;
       if (min[compl]=1) and (max[nodes] <= 10)  then
             begin
               z1:=min[nodes];
               z:=z1 div 2;
               z:=z*((z1-1) div 2);
               if z < infinity then z:=z*((z1-2) div 2);
               if z < infinity then z:=z*((z1-3) div 2) div 4;
               if z > min[xnum] then pushmin(xnum);
             end;
      end;
end;

procedure r060;
(*********************************************************)
(*                                                       *)
(*      if girth < infinity and (Nconn > 0 or mindeg > 1) *)
(*         then    genus >= E(1-2/girth)/2 - P/2 + Ncomp  *)
(*                                                       *)
(*********************************************************)
begin if (activerule[60]) and (min[girth] < infinity) then
 begin
   rule:=' 60/ ';
   rz:=min[edges]*(1-2/min[girth])-max[nodes]+2*min[ncomp];
   if  (min[nconn] > 0) or (min[mindeg] > 1) then
     begin
       if max[nodes] < infinity then
        begin
          z:=round(min[edges]*(1-2/min[girth])/2-max[nodes]/2
+min[ncomp]+hf);
          if z > min[genus] then pushmin(genus);
          if max[genus] < infinity then
             begin
               z:=2*max[genus]+max[nodes]-2*min[ncomp];
               z:=z*min[girth] div (min[girth]-2);
```

```
               if z < max[edges] then pushmax(edges);
               z:=min[edges]-2*max[genus]-max[nodes]+2
*min[ncomp];
               if (z > 0) and (max[girth] < infinity) then
                  begin
                    z:=(2*min[edges]) div z;
                    if z < max[girth] then pushmax(girth);
                   end;
             end;
        end;
      if (max[genus] < infinity) and (max[girth] < infinity) then
          begin
            rz:=rz+max[nodes]-2*max[genus];
            z:=round(rz+hf);
            if z > min[nodes] then pushmin(nodes);
            if max[nodes] < infinity then
                  begin
                    z:=trunk((max[nodes]-rz+2*min[ncomp])/2);
                    if z < max[ncomp] then pushmax(ncomp);
                  end;
          end;
     end
   else
     if 2*max[genus] < rz then
         if max[girth] < infinity then
                  begin
                    z:=1;
                    if z < max[mindeg] then pushmax(mindeg);
                    z:=0;
                    if z < max[nconn] then pushmax(nconn);
                  end
                else
                  if (min[nconn] > 0) or (min[mindeg] > 1) then
                         begin
                           z:=infinity;
                           pushmin(girth);
                         end;
 end;
end;

procedure r061;
(*************************************************)
(*                                               *)
(*     if genus <= 1 then eccov <= ncov*nind     *)
(*                                               *)
(*************************************************)
begin
  if (activerule[61]) and (max[nind] < infinity) and (max[ncov]
< infinity)
     and (min[genus] < 2) then
    begin
      rule:=' 61/ ';
```

```
      rulef(ncov,nind);
      if max[genus] <= 1 then
            begin
              if z < max[eccov] then pushmax(eccov);
              z:=(min[eccov]-1) div max[nind]+1;
              if z > min[ncov] then pushmin(ncov);
              z:=(min[eccov]-1) div max[ncov]+1;
              if z > min[nind] then pushmin(nind);
            end
          else
            if min[eccov] > z then
                        begin
                          z:=2;
                          pushmin(genus);
                        end;
    end;
end;

procedure r062;
(*************************************)
(*                                   *)
(*   mindeg >= P/2 then Nconn>=nind  *)
(*                                   *)
(*************************************)
begin if activerule[62] then
   begin
     rule:=' 62/ ';
     z:=min[mindeg];
     if 2*z >= max[nodes] then
                begin
                  rulea(nind,nconn,0);
                  nconnRnind:=ge;
                end
       else
         if  (max[nconn] < min[nind]) or
            (nconnRnind=lt) then
                begin
                  z:=(max[nodes]-1) div 2;
                  if (max[nodes] < infinity) and (z
< max[mindeg]) then
                                pushmax(mindeg);
                  z:=2*min[mindeg]+1;
                  if z > min[nodes] then pushmin(nodes);
                end;
    end;
end;

procedure r063;
(*******************************************)
(*                                         *)
(*  if connected then                      *)
(*      2d-3-(d*d-d-4)/P <= (P*P-2E)/P      *)
```

```
(*      (i.e. d <= (2P+1-sqrt(8E-8P+17))/2    *)
(*  where d=diam                              *)
(*                                            *)
(**********************************************)
var d:longint;
begin if (activerule[63]) and (min[connct]=1) then
       begin
         rule:=' 63/ ';
         if max[nodes] < infinity then
                begin
                  z:=8*min[edges]-8*max[nodes]+17;
                  if z >= 0 then
                          begin
                            z:=trunk((2*max[nodes]+1-sqrt(z))/2);
                            if z < max[diam] then pushmax(diam);
                          end;
                   d:=min[diam];
                   z:=max[nodes];
                   z:=(z*z+d*d-d-4-2*z*d+3*z) div 2;
                   if z < max[edges] then pushmax(edges);
                 end;
          z:=8*min[edges]-8*min[diam]+25;
          if z >= 0 then
              begin
                 z:=round((2*min[diam]-3+sqrt(z))/2+hf);
                 if z > min[nodes] then pushmin(nodes);
               end;
       end;
end;

procedure r064;
(**********************************************)
(*                                            *)
(*  P >= 3 and Nconn >= nind then hamiltonian  *)
(*                                            *)
(**********************************************)
begin if (activerule[64]) and (min[hamil]=0) then
begin
  rule:=' 64/ ';
  z:=1;
  if (min[nodes] > 2) and ((min[nconn] >= max[nind]) or
         (nconnRnind=ge)) then
            begin
              pushmin(hamil);
              nconnRnind:=ge;
              rulea(nind,nconn,0);
            end
      else
        if max[hamil]=0 then
           begin
             z:=2;
             if (min[nconn] >= max[nind]) or (nconnRnind=ge) then
```

```
                                  begin
                                    if z < max[nodes] then
pushmax(nodes);
                                   end
                     else
                       if min[nodes] > 2 then
                              begin
                               rulea(nconn,nind,-1);
                               nconnRnind:=lt;
                              end;
            end;
 end;
end;

procedure r065;
(*****************************************)
(*                                       *)
(*  E >= (P*P-3*P+6)/2 => hamiltonian    *)
(*                                       *)
(*****************************************)
begin if (activerule[65]) and (min[hamil] = 0) then
begin
  rule:=' 65/ ';
  z:=max[nodes];
  z1:=(z*z-3*z+7) div 2;
  if min[edges] >= z1 then
     begin
       z:=1;
       pushmin(hamil);
     end
    else
     if max[hamil]=0 then
        begin
          z:=z1-1;
          if z < max[edges] then pushmax(edges);
          z:=8*min[edges]-11;
          if z > 0 then
             begin
               z:=round((3+sqrt(z))/2+hf);
               if z > min[nodes] then pushmin(nodes);
             end;
        end;
   end;
end;

procedure r066;
(*****************************************)
(*                                       *)
(*  Nconn > 3 and plnar => hamiltonian   *)
(*                                       *)
(*****************************************)
begin if (activerule[66]) and (min[hamil] = 0) and (max[plnar] =
```

```
1)
         and (max[nconn] > 3) then
  begin
    rule:=' 66/ ';
    z:=1;
    if (min[nconn] > 3) and (min[plnar]=1) then pushmin(hamil)
     else
       if max[hamil]=0 then
          if min[plnar]=1 then
               begin
                 z:=3;
                 pushmax(nconn);
               end
             else
               if min[nconn] > 3 then
                  begin
                    z:=0;
                    pushmax(plnar);
                  end;
    end;
end;

procedure r067;
(************************************************)
(*                                              *)
(* connected, not odd cycle, and not complete   *)
(*           ==> chr <= maxdeg                  *)
(*                                              *)
(************************************************)
begin if (activerule[67]) and (max[connct] = 1) and (min[compl] =
0)then
begin
  rule:=' 67/ ';
  if max[chr] > min[maxdeg] then
    begin
      z:=max[cycle]+min[cycle];
      if (z = 2) and (((max[nodes]=min[nodes]) and
         (not(odd(max[nodes])))) or (pReven=eq))
                      then z:=0;
      if (z=0) and (min[connct]=1) and (max[compl]=0) then
rulea(chr,maxdeg,0)
          else
            if min[chr] > max[maxdeg] then
              begin
                if (z=0) and (max[compl]=0) then pushmax(connct)
                  else
                    if (z=0) and (min[connct]=1) then
                        begin
                          z:=1;
                          pushmin(compl);
                        end
                      else
```

```
                        if (min[connct]=1) and (max[compl]=0)
then
                            begin
                              z:=1;
                              pushmin(cycle);
                              z:=min[nodes]+1;
                              if odd(z) then pushmin(nodes);
                              z:=max[nodes]-1;
                              if (max[nodes] < infinity) and
                                 (odd(z)) then pushmax(nodes);
                              pRodd:=eq;
                              pReven:=ne;
                            end;
                    end;
        end;
      end;
end;

procedure r068;
(*****************************************************)
(*                                                   *)
(*        complete ==> regular,                      *)
(*                       if P even then echr=P-1     *)
(*                                 else echr=P       *)
(*                                                   *)
(*****************************************************)
begin if (activerule[68]) and (max[compl] = 1) then
begin
  rule:=' 68/ ';
  if min[compl]=1 then
    begin
      z:=1;
      pushmin(reg);
      if (min[nodes]=max[nodes]) or (pRodd=eq) or (pReven=ne)
then
      if (odd(min[nodes])) or (pRodd=eq) then
         begin
           pRodd:=eq;
           pReven:=ne;
           echrRmaxdeg:=gt;
           rulea(echr,nodes,0);
           rulea(nodes,echr,0);
         end
        else
          begin
            pRodd:=ne;
            pReven:=eq;
            echrRmaxdeg:=eq;
            rulea(echr,nodes,-1);
            rulea(nodes,echr,1);
          end;
        end
```

```
      else
        if (max[reg]=0) or (max[echr] < min[nodes]-1) then
           begin
             z:=0;
             pushmax(compl);
           end;
    end;
end;

procedure r069;
(*********************************)
(*                               *)
(*    chr >= 2E/(2E-spectr**2)   *)
(*                               *)
(*********************************)
begin if activerule[69] then
    begin
      rule:=' 69/ ';
      z:=max[edges];
      if z < infinity then
          begin
            z:=round(2*z/(2*z-lammin*lammin)+hf);
            if z > min[chr] then pushmin(chr);
            z:=2*max[edges];
            if max[chr] < infinity then
                begin
                  rz:=sqrt((z*max[chr]-z)/max[chr]);
                  if rz < lammax then pushlammax;
                end;
          end;
       z:=max[chr];
       if z < infinity then
            begin
              z:=round(z*lammin*lammin/(2*(z-1))+hf);
              if z > min[edges] then pushmin(edges);
            end;
    end;
end;

procedure r070;
(************************************)
(*                                  *)
(*  if genus <= 1 and               *)
(*       girth >= 4 then chr <= 4   *)
(*       girth = 3  then chr <= 7   *)
(*                                  *)
(************************************)
begin
  if (activerule[70]) and (max[chr] > 4) and
     (min[genus] < 2) and (max[girth] > 3) then
     begin
       rule:=' 70/ ';
```

```
       if max[genus] <= 1 then
            begin
              if min[girth] > 3 then z:=4
                    else z:=7;
              if z < max[chr] then pushmax(chr);
              z:=3;
              if min[chr] > 4 then pushmax(girth);
            end
          else
            if ((min[chr] > 4) and (min[girth] > 3)) or
               (min[chr] > 7) then
                 begin
                   z:=2;
                   pushmin(genus);
                 end;
     end;
end;

procedure r071;
(*******************************************)
(*                                         *)
(* if girth < infinity then                *)
(*       circ <= P-(Ncomp-1)*(mindeg+1)    *)
(*       circ <= E-(Ncomp-1)*mindeg        *)
(*       maxdeg > 1                        *)
(*                                         *)
(*******************************************)
begin if (activerule[71]) and (min[girth] < infinity) then
begin
  rule:=' 71/ ';
  z:=infinity;
  if max[girth] < infinity then
        begin
          if min[ncomp]=1 then
             begin
               rulea(circ,nodes,0);
               rulea(circ,edges,0);
             end
           else
             begin
               z:=min[circ]+(min[ncomp]-1)*(min[mindeg]+1);
               if z > min[nodes] then pushmin(nodes);
               if max[nodes] < infinity then
                        begin
                          z:=max[nodes]-(z-min[circ]);
                          if z < max[circ] then pushmax(circ);
                          z:=(max[nodes]-min[circ]) div
(min[ncomp]-1)-1;
                          if z < max[mindeg] then
pushmax(mindeg);
                        end;
                z:=min[circ]+(min[ncomp]-1)*min[mindeg];
```

```
                if z > min[edges] then pushmin(edges);
                if max[edges] < infinity then
                        begin
                          z:=max[edges]-(z-min[circ]);
                          if z < max[circ] then pushmax(circ);
                          z:=(max[edges]-min[circ]) div
(min[ncomp]-1);
                          if z < max[mindeg] then
pushmax(mindeg);
                         end;
             end;
          if max[nodes] < infinity then
                       begin
                         z:=(max[nodes]-min[circ]) div
(min[mindeg]+1)+1;
                         if z < max[ncomp] then pushmax(ncomp);
                       end;
          if max[edges] < infinity then
                        begin
                          z:=(max[edges]-min[circ]) div
min[mindeg] +1;
                          if z < max[ncomp] then pushmax(ncomp);
                         end;
          z:=2;
          if z > min[maxdeg] then pushmin(maxdeg);
        end
      else
        if (min[circ] > max[edges]) or (max[maxdeg] < 2) or
           (min[circ] > max[nodes]) then pushmin(girth);
  end;
end;

procedure r072;
(*****************************)
(*                           *)
(*  hamiltonian <=> circ=P   *)
(*                           *)
(*****************************)
begin if activerule[72] then
begin
  rule:=' 72/ ';
  z:=1;
  if (min[hamil]=1) or ((min[circ]=max[nodes]) and
     (max[nodes] < infinity) and (min[circ]=max[circ])) or
     (circRnodes=eq) then
          begin
            if z > min[hamil] then pushmin(hamil);
            rulea(nodes,circ,0);
            rulea(circ,nodes,0);
            circRnodes:=eq;
          end
        else
```

```
          if (max[hamil]=0) or (max[circ] < min[nodes]) or
             (min[circ] = infinity) or
             (circRnodes=lt) then
                  begin
                    z:=0;
                    if z < max[hamil] then pushmax(hamil);
                    if (max[forest]=0) and (max[nodes]
< infinity) then
                                    begin
                                      rulea(circ,nodes,-1);
                                      circRnodes:=lt;
                                    end;
                  end;
 end;
end;

procedure r073;
(*********************************)
(*                               *)
(* if hamiltonian then           *)
(*           arbor >= 2          *)
(*           Nconn >= 2          *)
(*           nind <= P/2         *)
(*           ncov >=P/2          *)
(*           ecov <= (P+1)/2     *)
(*           eind  >= (P-1)/2    *)
(*           nccov <= (P+1)/2    *)
(*                               *)
(*********************************)
begin if (activerule[73]) and (max[hamil] = 1) then
begin
  rule:=' 73/ ';
  z:=0;
  if min[hamil]=1 then
         begin
           z:=2;
           if z > min[arbor] then pushmin(arbor);
           if z > min[nconn] then pushmin(nconn);
           if max[nodes] < infinity then
                  begin
                    z:=max[nodes] div 2;
                    if z < max[nind] then pushmax(nind);
                    z:=(max[nodes]+1) div 2;
                    if z < max[ecov] then pushmax(ecov);
                    if z < max[nccov] then pushmax(nccov);
                  end;
            z:=(min[nodes]+1) div 2;
            if z > min[ncov] then pushmin(ncov);
            z:=min[nodes] div 2;
            if z > min[eind] then pushmin(eind);
            z:=2*max[ncov];
            z1:=2*max[eind]+1;
```

```
            if z1 < z then z:=z1;
            if z < max[nodes] then pushmax(nodes);
            z:=2*min[nind];
            z1:=2*min[ecov]-1;
            if z < z1 then z:=z1;
            z1:=2*min[nccov]-1;
            if z < z1 then z:=z1;
            if z > min[nodes] then pushmin(nodes);
          end
        else
          if (max[arbor] < 2) or (max[nconn] < 2) or
            (min[nind] > max[nodes] div 2) or
            (max[ncov] < (min[nodes]+1) div 2) or
            (min[ecov] > (max[nodes]+1) div 2) or
            (max[eind] < min[nodes] div 2) or
            (min[nccov] > (max[nodes]+1) div 2) then
pushmax(hamil);
   end;
end;

(*
procedure r074;      *)
(*****************************************)
(*                                       *)
(*     Bwidth <= P-1-trunc(nind/2)       *)
(*       (removed for R402)              *)
(*                                       *)
(*****************************************)

procedure r075;
(*************************************)
(*                                   *)
(*   let t=trunc(P/nind)             *)
(*                                   *)
(*  then  E >= t(P-(nind(t+1))/2)    *)
(*                                   *)
(*       (Turan's theorem)           *)
(*                                   *)
(*************************************)
begin if (activerule[75]) and (max[nind] < infinity) then
begin
  rule:=' 75/ ';
  z:=min[nodes] div max[nind];
  z:=round(z*(min[nodes]-max[nind]*(z+1)/2)+hf);
  if z > min[edges] then pushmin(edges);
  if max[edges] < infinity then
          begin
            z:=max[nind]+trunk(sqrt(2*max[edges]*max[nind]));
            if z < max[nodes] then pushmax(nodes);
          end;
 end;
end;
```

```
procedure r076;
(*********************************)
(*                               *)
(*  eccov <= nccov +P*(nccov-1)/2   *)
(*                               *)
(*********************************)
begin if activerule[76] then
  begin
    rule:=' 76/ ';
    if(max[nodes] < infinity) and (max[nccov] < infinity) then
        begin
          z:=max[nccov];
          z:=z+(max[nodes]*(z-1)) div 2;
          if z < max[eccov] then pushmax(eccov);
        end;
     z:=max[nodes];
     if z < infinity then
        begin
          z:=(2*min[eccov]+2*z+1) div (z+2);
          if z > min[nccov] then pushmin(nccov);
        end;
      if max[nccov] < infinity then
        begin
          z:=max[nccov]-1;
          if z > 0 then
                 begin
                   z:=(2*min[eccov]-3) div z-1;
                   if z > min[nodes] then pushmin(nodes);
                 end;
        end;
    end;
end;

procedure r077;
(*************************************************************
*)
(*
*)
(*  P >= 6*mindeg and E > (P-mindeg)(P-mindeg-1)/2 +mindeg**2
*)
(*              then   Hamiltonian
*)
(*
*)
(*************************************************************
*)
begin if (activerule[77]) and (min[hamil] = 0) then
begin
  rule:=' 77/ ';
  z:=min[mindeg];
  z:=((max[nodes]-z)*(max[nodes]-z-1)+1) div 2 + z*z;
```

```
  if (min[edges] > z) and (min[nodes] >= 6*max[mindeg]) then
       begin
         z:=1;
         pushmin(hamil);
       end
     else
       if max[hamil] = 0 then
          begin
            if min[nodes] >= 6*max[mindeg] then
                 begin
                   if z < max[edges] then pushmax(edges);
                 end
               else
                if min[edges] > z then
                begin
                  z:=6*max[mindeg]-1;
                  if z < max[nodes] then pushmax(nodes);
                  z:=min[nodes] div 6+1;
                  if z > min[mindeg] then pushmin(mindeg);
                end;
          end;
 end;
end;

procedure r078;
(*************************************)
(*                                   *)
(*   P > 3 and E >= 2P-3 ==>         *)
(*      G has a cycle with a chord.  *)
(*       i.e.  girth <= (circ+2)/2   *)
(*                                   *)
(*************************************)
begin if activerule[78] then
begin
  rule:=' 78/ ';
  if ((min[nodes] > 3) and (min[edges] > 2*max[nodes]-4)) then
          begin
            z:=(max[circ]+2) div 2;
            if (max[circ] < infinity) and (z < max[girth]) then
pushmax(girth);
            z:=2*min[girth]-2;
            if z > min[circ] then pushmin(circ);
            girthRcirc:=lt;
          end
    else
      if (girthRcirc=eq) or (min[girth] > (max[circ]+2) div 2)
then
                  begin
                    if min[nodes] > 3 then
                             begin
                               z:=2*max[nodes]-4;
                               if z < max[edges] then
```

```
pushmax(edges);
                                z:=(min[edges]+5) div 2;
                                if z > min[nodes] then
pushmin(nodes);
                              end;
                    if min[edges] > 2*max[nodes]-4 then
                                begin
                                  z:=3;
                                  pushmax(nodes);
                                end;
                  end;
 end;
end;

procedure r079;
(*********************************************)
(*                                           *)
(*  if not a forest(girth defined) then      *)
(*         nind,radius >= trunc(girth/2)     *)
(*               eind  >= trunc(circ/2)      *)
(*                                           *)
(*********************************************)
begin if (activerule[79]) and (min[forest] = 0) then
   begin
     rule:=' 79/ ';
     z:=1;
     if max[forest]=0 then
             begin
               z:=min[girth] div 2;
               if z > min[nind] then pushmin(nind);
               if z > min[radius] then pushmin(radius);
               z:=min[circ] div 2;
               if z > min[eind] then pushmin(eind);
               if max[eind] < infinity then
                 begin
                   z:=2*max[eind]+1;
                   if z < max[circ] then pushmax(circ);
                 end;
               if max[radius] < max[nind] then z:=2*max[radius]+1
                                          else z:=2*max[nind]+1;
               if z < max[girth] then pushmax(girth);
             end
          else
            if (max[nind] < min[girth] div 2) or
                (max[radius] < min[girth] div 2) or
                (max[eind] < min[circ] div 2) then
pushmin(forest);
  end;
end;

procedure r080;
(*****************************************************)
```

```
(*                                                          *)
(*  if 3 < girth < infinity then not complete               *)
(*  if girth=infinity and P > 2 then not complete           *)
(*                                                          *)
(************************************************************)
begin if (activerule[80]) and (max[compl] = 1) then
     begin
       rule:=' 80/ ';
       z:=0;
       if ((min[girth] > 3) and (max[girth] < infinity)) or
          ((min[girth]=infinity) and (min[nodes] > 2)) then
pushmax(compl);
     end;
end;

procedure r081;
(************************************************************)
(*                                                          *)
(*   Reg then nind <= P/2-(clique-1)*(clique-2)/(2*mindeg)  *)
(*                                                          *)
(************************************************************)
begin if (activerule[81]) and (max[reg]=1) then
        begin
          rule:=' 81/ ';
          if min[reg]=1 then
             begin
               z:=max[nodes];
               if z < infinity then
                 begin
                   k:=min[clique];
                   z:=z-2*min[nind];
                   if z > 0 then
                      begin
                        z:=((k-1)*(k-2)+z-1) div z;
                        if z > min[mindeg] then pushmin(mindeg);
                      end;
                   z:=max[nodes];
                   k:=max[mindeg];
                   if k < infinity then
                       begin
                         z:=(z*k-(min[clique]-1)*(min[clique]-2))
div (2*k);
                         if z < max[nind] then pushmax(nind);
                         z:=max[nodes];
                         z:=trunk((3+sqrt(4*k*(z-2*min[nind])+
1))/2);
                         if z < max[clique] then pushmax(clique);
                       end;
                 end;
               k:=max[maxdeg];
               if k < infinity then
                    begin
```

```
                    z:=min[clique];
                    z:=2*min[nind]+((z-1)*(z-2)+k-1) div k;
                    if z > min[nodes] then pushmin(nodes);
                  end;
            end
          else
            begin
              z:=max[mindeg];
              if (z < infinity) and (max[nodes] < infinity) then
                 begin
                   k:=min[clique];
                   z1:=(max[nodes]*z-(k-1)*(k-2)) div (2*z);
                   if min[nind] > z1 then
                         begin
                           z:=0;
                           pushmax(reg);
                         end;
                 end;
             end;
        end;
end;

procedure r082;
(*********************************************)
(*                                           *)
(*   if mindeg >= P div 2 then econn=mindeg  *)
(*                                           *)
(*********************************************)
begin if activerule[82] then
begin
  rule:=' 82/ ';
  z:=(max[nodes]-2) div 2;
  if max[econn] > z then z:=max[econn];
  if z < max[mindeg] then pushmax(mindeg);
  if max[econn] < min[mindeg] then
          begin
            z:=2*min[mindeg]+2;
            if z > min[nodes] then pushmin(nodes);
          end;
end;
end;

procedure r083;
(*****************************************)
(*                                       *)
(* if genus > 0 then                     *)
(*     arbor <= (9+sqrt(1+48*genus))/4   *)
(*                                       *)
(*****************************************)
begin if (activerule[83]) and (max[genus] > 0) then
begin
  rule:=' 83/ ';
```

```
  z:=trunk((9+sqrt(1+48*max[genus]))/4);
  if min[genus] > 0 then
       begin
         if (max[genus] < infinity) and (z < max[arbor]) then
pushmax(arbor);
         z:=min[arbor];
         z:=(2*z*z-9*z+15) div 6;
         if z > min[genus] then pushmin(genus);
       end
     else
       if min[arbor] > z then
            begin
              z:=0;
              pushmax(genus);
            end;
end;
end;

procedure r084;
(*************************************************)
(*                                               *)
(*   if clique = 2 then  arbor <= 2+sqrt(genus)  *)
(*                                               *)
(*************************************************)
begin if (activerule[84]) and (min[clique] <= 2) then
begin
  rule:=' 84/ ';
  z:=3;
  if max[clique] < 3 then
         begin
           z:=2+trunk(sqrt(max[genus]));
           if (max[genus] < infinity) and (z < max[arbor]) then
pushmax(arbor);
           if min[arbor] >= 3 then
              begin
                 z:=min[arbor]-2;
                 z:=z*z;
                 if z > min[genus] then pushmin(genus);
               end;
         end
       else
         if min[arbor] > 2+trunk(sqrt(max[genus])) then
pushmin(clique);
end;
end;

procedure r085;
(*************************************************)
(*                                               *)
(*  if clique = 2 then  chr <= 3+2*sqrt(genus)   *)
(*                                               *)
(*************************************************)
```

```
begin if (activerule[85]) and (min[clique]=2) then
begin
  rule:=' 85/ ';
  if max[clique]=2 then
       begin
         if max[genus] < infinity then
            begin
              z:=3+trunk(2*sqrt(max[genus]));
              if z < max[chr] then pushmax(chr);
            end;
         z:=min[chr]-3;
         if z > 0 then
            begin
              z:=(z*z+3) div 4;
              if z > min[genus] then pushmin(genus);
            end;
        end
     else
       if min[chr] > 3+trunk(2*sqrt(max[genus])) then
                  begin
                    z:=3;
                    pushmin(clique);
                   end;
end;
end;

procedure r086;
(*****************************************************************
******)
(*
*)
(*            min(P div 2,mind)    if mind<=maxd-2,P<maxd+mind
*)
(*            P*mind/(maxd+mind)   if mind<=maxd-2 and
P>=maxd+mind    *)
(*            P*maxd/(2*maxd+2)    if maxd=mind=EVEN or
mind=maxd-1=ODD*)
(*            (P*mind+1)/(2mind+2) if mind=maxd-1=EVEN
*)
(* eind >=   P/2                  if mind=maxd=ODD and P=mind+1
*)
(*            (P-u(mind-1))/2-k    if mind=maxd=ODD and P>mind+1
*)
(*                                   where 0<2k+1<=mind,0<r<2mind+
4    *)
(*                                          and
*)
(*                                   P=u(mind+1)**2+(2k+1)(mind+
2)+r   *)
(*
*)
(*****************************************************************
```

```
******)
var u,r,k: longint;
begin if activerule[86] then
  begin
    rule:=' 86/ ';
    z:=0;
    if max[mindeg] <= min[maxdeg]-2 then
        begin
          if max[nodes]<min[mindeg]+min[maxdeg] then
               begin
                 z:=min[nodes] div 2;
                 if z > min[mindeg] then z:=min[mindeg];
               end
             else z:=round(min[nodes]/(1
+max[maxdeg]/min[mindeg])+hf);
        end
       else
         if (min[mindeg]=max[mindeg]) and
(min[maxdeg]=max[maxdeg])
           and (min[mindeg]=max[maxdeg]-1) then
           begin
             if odd(min[mindeg]) then
                   z:=round(min[nodes]/(2+2/min[maxdeg])+hf)
              else z:=(min[nodes]*min[mindeg]) div (2
*min[mindeg]+2)+1;
           end
         else
           if min[mindeg]=max[maxdeg] then
             if not(odd(min[mindeg])) then
                z:=(min[nodes]*min[maxdeg]-1) div (2*min[maxdeg]+
2)+1
              else
                if (max[nodes]=min[mindeg]+1) or (mindegRpminus1
=eq) then
                    begin
                      z:=(min[nodes]+1) div 2;
                      mindegRpminus1:=eq;
                    end
                  else
                    if (min[nodes]=max[nodes]) and
                       (min[nodes]>min[mindeg]+1) then
                         begin
                           z:=min[mindeg]+1;
                           z:=z*z;
                           u:=(min[nodes]-min[mindeg]-3) div z;
                           r:=min[nodes]-min[mindeg]-3-u*z;
                           r:=r+min[mindeg]+3;
                           k:=(r div (min[mindeg]+2)-1) div 2;
                           z:=(min[nodes]-u*(min[mindeg]-1)+1)
div 2-k;
                         end;
       if z > min[eind] then pushmin(eind);
```

```
     end;
end;

procedure r087;
(*****************************************)
(*                                       *)
(* if genus > 0 then                     *)
(*    mindeg <= (5+sqrt(1+48genus))/2     *)
(*                                       *)
(*****************************************)
begin if (activerule[87]) and (max[genus] > 0) and (max[genus]
< infinity) then
begin
  rule:=' 87/ ';
  z:=trunk((5+sqrt(1+48*max[genus]))/2);
  if min[genus] > 0 then
        begin
          if z < max[mindeg] then pushmax(mindeg);
          if min[mindeg] > 6 then
            begin
              z:=min[mindeg]-3;
              z:=(z*(z+1)+11) div 12;
              if z > min[genus] then pushmin(genus);
            end;
        end
      else
        if min[mindeg] > z then
           begin
             z:=0;
             pushmax(genus);
           end;
 end;
end;

procedure r088;
(*************************************)
(*                                   *)
(*  if genus > 0 and clique =2 then    *)
(*          Econn <= 2+2*sqrt(genus)   *)
(*                                   *)
(*************************************)
begin
  if (activerule[88]) and (max[genus] > 0) and (max[genus]
< infinity)
     and (min[clique] <= 2) then
     begin
       rule:=' 88/ ';
       z:=2+trunk(2*sqrt(max[genus]));
       if max[clique] < 3 then
             begin
               if z < max[econn] then pushmax(econn);
               if (min[genus] > 0) and (min[econn] > 4) then
```

```
                    begin
                      z:=min[econn]-2;
                      z:=(z*z+3) div 4;
                      if z > min[genus] then pushmin(genus);
                    end;
            end
          else
            if min[econn] > z then
                if min[genus] > 0 then
                   begin
                     z:=3;
                     pushmin(clique);
                   end
                  else
                    if min[clique] > 3 then
                       begin
                         z:=0;
                         pushmax(genus);
                       end;
     end;
end;

procedure r089;
(*****************************************)
(*                                       *)
(*  if plnar(genus=0) then              *)
(*      if girth=3   then  Econn<=5      *)
(*      if girth=4   then  Econn<=3      *)
(*      if girth=5   then  Econn<=3      *)
(*      if girth>=6  then  Econn<=2      *)
(*                                       *)
(*****************************************)
begin
  if (activerule[89]) and (min[genus] = 0) then
      begin
        rule:=' 89/ ';
        if max[genus]=0 then
             begin
            z:=0;
                 if max[girth]=3 then z:=5
                    else
                  if (min[girth] > 3) and (max[girth] < 6) then
z:=3
                       else if min[girth] > 5 then z:=2;
                 if (z > 0) and (z < max[econn]) then
pushmax(econn);
                 z:=0;
                 if min[econn] > 3 then z:=3
                      else if min[econn] > 2 then z:=5;
                 if (z > 0) and (z < max[girth]) then
pushmax(girth);
               end
```

```
            else
              begin
                z:=1;
                if (min[econn] > 5) or
                   ((min[econn] > 3) and (min[girth] > 3)) or
                   ((min[econn] > 2) and (min[girth] > 5)) then
pushmin(genus);
              end;
    end;
end;

procedure r090;
(*********************************)
(*                               *)
(*   if genus <= 1 then          *)
(*      girth =3 ==> Econn <= 6  *)
(*      girth =4 ==> Econn <= 4  *)
(*      girth =5 ==> Econn <= 3  *)
(*      girth =6 ==> Econn <= 3  *)
(*      girth>=7 ==> Econn <= 2  *)
(*                               *)
(*********************************)
begin if (activerule[90]) and (min[genus] < 2) then
  begin
    rule:=' 90/ ';
    z:=2;
    if max[genus] <= 1 then
        begin
          if max[girth]=3 then z:=6
             else if (min[girth]=max[girth]) and (min[girth]=4)
then z:=4
              else if (min[girth]>4) and (max[girth]<7) then z:=3
                else if min[girth] <= 6  then z:=max[econn];
          if z < max[econn] then pushmax(econn);
          z:=6;
          if min[econn] > 4 then z:=3
             else if min[econn] > 3 then z:=4
              else if min[econn] <= 2 then z:=max[girth];
          if z < max[girth] then pushmax(girth);
        end
      else
          if (min[econn] > 6) or
             ((min[econn] > 4) and (min[girth] > 3)) or
             ((min[econn] > 3) and (min[girth] > 4)) or
             ((min[econn] > 2) and (min[girth] > 6)) then
pushmin(genus);
   end;
end;


procedure r091;
(***************************************)
```

```
(*                                          *)
(*  P >= girth*(mindeg-1)**z                *)
(*     where  mindeg >= 3                   *)
(*            1 <= z = (girth-1) div 4      *)
(*                                          *)
(********************************************)
begin if (activerule[91]) and (min[mindeg] >= 3) then
  begin
    z1:=(min[girth]-1) div 4;
    if z1 >= 1 then
       begin
         rule:=' 91/ ';
         power(min[mindeg]-1,z1,k);
         z:=k*min[girth];
         if z > min[nodes] then pushmin(nodes);
         if max[nodes] < infinity then
             begin
               z:=max[nodes] div k;
               if z < max[girth] then pushmax(girth);
               z:=trunk(root(max[nodes]/min[girth],z1))+1;
               if z < max[mindeg] then pushmax(mindeg);
             end;
       end;
  end;
end;

procedure r092;
(*****************************************************************
)
(*
*)
(*       if nconn >= 2 then circ >= min(nodes,2*mindeg)
*)
(*
*)
(*****************************************************************
)
begin
  if (activerule[92]) and (max[nconn] > 1) and (min[hamil] = 0)
then
    begin
      rule:=' 92/ ';
      z:=2*min[mindeg];
      if min[nodes] < z then z:=min[nodes];
      if min[nconn] > 1 then
         begin
           if min[circ] < z then pushmin(circ);
           if (max[circ] < min[nodes]) or (max[hamil] = 0) then
                begin
                  z:=max[circ] div 2;
                  if max[mindeg] > z then pushmax(mindeg);
                end
```

```
              else
                if max[circ] < 2*min[mindeg] then
                   begin
                     z:=1;
                     pushmin(hamil);
                   end;
           end
         else
          if max[circ] < z then
            begin
              z:=1;
              pushmax(nconn);
            end;
    end;
end;

procedure r093;
(*******************************************************)
(*                                                     *)
(*  if diam = 2 then                                   *)
(*      if P >= mindeg**3 + mindeg + 2                 *)
(*           or                                        *)
(*         P >= mindeg**3 + 2 and P or mindeg is odd   *)
(*  then                                               *)
(*     E >= trunc(((P-1)(mindeg+1)+1)/2)               *)
(*                                                     *)
(*******************************************************)
var boole:boolean;
begin
  if (activerule[93]) and (min[diam] = max[diam]) and (min[diam]
= 2) then
   begin
     rule:=' 93/ ';
     if max[mindeg] < infinity then
       begin
         rz:=max[mindeg];
         rz:=rz*rz*rz+rz;
         if rz < infinity then
          begin
            z:=trunk(rz)+1;
                if ((((min[nodes]=max[nodes]) and
(odd(min[nodes]))) or (pRodd=eq)) or
               ((min[mindeg]=max[mindeg]) and
(odd(min[mindeg])))) then boole:=true
                     else boole:=false;
            if (min[nodes] > z) or ((min[nodes] > (z-
max[mindeg])) and boole) then
                  begin
                    z:=((min[nodes]-1)*(min[mindeg]+1)+1) div
2;
                    if z > min[edges] then pushmin(edges);
                  end
```

```
                else
                  if max[edges] < ((min[nodes]-1)*(min[mindeg]+
1)+1) div 2 then
                       if z < max[nodes] then pushmax(nodes);
          end;
       end;
   end;
end;

procedure r094;
(*****************************************************)
(*                                                   *)
(* if diam = 2 and nconn <> 2 then                   *)
(*      if P >= nconn**3 + nconn + 2                 *)
(*           or                                      *)
(*         P >= nconn**3 + 2 and P or nconn is odd   *)
(*     then                                          *)
(*        E >= trunc(((P-1)(nconn+1)+1)/2)           *)
(*                                                   *)
(*****************************************************)
var boole:boolean;
begin
  if (activerule[94]) and (min[diam] = max[diam]) and (min[diam]
= 2) then
     begin
       rule:=' 94/ ';
       if  (min[nconn] > 2) or (max[nconn] < 2) then
          begin
            rz:=max[nconn];
            rz:=rz*rz*rz+rz+1;
            if rz < infinity then
              begin
                z:=trunk(rz);
                     if ((((min[nodes]=max[nodes]) and
(odd(min[nodes])))
                   or (pRodd=eq)) or ((min[nconn]=max[nconn]) and
                   (odd(min[nconn])))) then boole:=true
                              else boole:=false;
                if (min[nodes]>z) or ((min[nodes]>(z-max[nconn]))
and boole) then
                  begin
                    z:=((min[nodes]-1)*(min[nconn]+1)+1) div
2;
                    if z > min[edges] then pushmin(edges);
                  end
                else
                  if max[edges]<((min[nodes]-1)*(min[nconn]+
1)+1) div 2 then
                    if z < max[nodes] then pushmax(nodes);
              end;
            end;
        end;
```

```
end;

procedure r095;
(*************************************************)
(*                                               *)
(* if diam = 2 then                              *)
(*     if P >= econn**3 + econn + 2              *)
(*           or                                  *)
(*        P >= econn**3 + 2 and P or econn is odd  *)
(*   then                                        *)
(*     E >= trunc(((P-1)(econn+1)+1)/2)          *)
(*                                               *)
(*************************************************)
var boole:boolean;
begin
  if (activerule[95]) and (max[nodes] < infinity)
      and (max[econn] < infinity) and (max[diam] = min[diam])
      and (min[diam] = 2) then
    begin
      rule:=' 95/ ';
      rz:=max[econn];
      rz:=rz*rz*rz+rz+1;
      if rz < infinity then z:=trunk(rz)
                       else z:=infinity;
       if ((((min[nodes]=max[nodes]) and (odd(min[nodes]))) or
(pRodd=eq)) or
        ((min[econn]=max[econn]) and (odd(min[econn])))) then
boole:=true
                 else boole:=false;
      if (min[nodes]>z) or ((min[nodes]>(z-max[econn])) and
boole) then
          begin
            z:=((min[nodes]-1)*(min[econn]+1)+1) div 2;
            if z > min[edges] then pushmin(edges);
          end
         else
             if max[edges]<((min[nodes]-1)*(min[econn]+1)+1) div
2 then
                if z < max[nodes] then pushmax(nodes);
     end;
end;

procedure r096;
(*********************************)
(*                               *)
(* if  girth is defined then     *)
(*      P >= (arb-1)*(g-1)+1     *)
(*                               *)
(*********************************)
begin
  if (activerule[96]) and (max[girth] < infinity) and
(min[arbor] > 2) then
```

```
     begin
       rule:=' 96/ ';
       z:=(min[arbor]-1)*(min[girth]-1)+1;
       if z > min[nodes] then pushmin(nodes);
       k:=max[nodes];
       if k < infinity then
         begin
           z:=(k-1) div (min[girth]-1)+1;
           if z < max[arbor] then pushmax(arbor);
           if min[arbor] > 1 then
               begin
                 z:=(k-1) div (min[arbor]-1)+1;
                 if z < max[girth] then pushmax(girth);
               end;
          end;
     end;
end;

procedure r097;
(*********************************************)
(*                                           *)
(*   if clique < 3 and chr > 3 then P > 10     *)
(*                                           *)
(*********************************************)
begin
  if (activerule[97]) and (min[nodes] < 11) and (min[clique] < 3)
     and (max[chr] > 3) then
     begin
       rule:=' 97/ ';
       z:=3;
       if (max[clique] < 3) and (min[chr] > 3) then
                begin
                  z:=11;
                  pushmin(nodes);
                end
              else
                if max[nodes] < 11 then
                   if max[clique] < 3 then pushmax(chr)
                                      else if min[chr] >= 4 then
pushmin(clique);
      end;
end;

procedure r098;
(*************************************************************
**)
(*
*)
(*       let g=girth/(girth-2)
*)
(*          if girth >= 4, genus >= 2, and chr >= 1+2g
*)
```

```
(*                 then
*)
(*            chr <= (3+6g+sqrt(57-60g+36g**2+48*g*genus)) div 6
*)
(*
*)
(*************************************************************
**)
var gamma:real;
begin
  if (activerule[98]) and (max[genus] < infinity) and
     (max[genus] > 1) and (min[girth] > 3) then
       begin
         gamma:=min[girth]/(min[girth]-2);
         rule:=' 98/ ';
         rz:=57-60*gamma+36*gamma*gamma+48*gamma*max[genus];
         if rz >= 0 then
             begin
               z:=trunk((3+6*gamma+sqrt(rz))/6);
               if z < 2*gamma then z:=trunk(2*gamma);
               if z < max[chr] then pushmax(chr);
             end;
       end;
end;

procedure r099;
(*************************************)
(*                                   *)
(*    if diam <= 2 then Econn=mindeg     *)
(*                                   *)
(*************************************)
begin
  if (activerule[99]) and (min[diam] < 3) then
     begin
       rule:=' 99/ ';
       z:=3;
       if max[diam] < 3 then rulea(mindeg,econn,0)
              else
                if max[econn] < min[mindeg] then
                      pushmin(diam);
     end;
end;

procedure r100;
(*********************************************)
(*                                           *)
(*   if nind >= eind then  eccov <= ncov*nind   *)
(*                                           *)
(*********************************************)
begin
  if (activerule[100]) and (max[nind] < infinity)
      and (max[ncov] < infinity) then
```

```
    begin
      rule:='100/ ';
      rulef(ncov,nind);
      if min[nind] >= max[eind] then
                 begin
                   if z < max[eccov] then pushmax(eccov);
                   z:=(min[eccov]-1) div max[nind]+1;
                   if z > min[ncov] then pushmin(ncov);
                   z:=(min[eccov]-1) div max[ncov]+1;
                   if z > min[nind] then pushmin(nind);
                 end
               else
                 if min[eccov] > z then rulea(nind,eind,-1);
   end;
end;


end.
```