```
unit ruleAtoF;

  interface

    uses
          globals,cmmnds1,pusherr,pushStack;

           procedure rulea(parm1,parm2,con:longint);
           procedure ruleb(parm1,parm2,parm3:longint);
           procedure rulec(parm,con:longint);
           procedure ruled(parm1,parm2,parm3:longint);
           procedure rulee(parm1,parm2,parm3,con:longint);
           procedure rulef(xa,xb:longint);

implementation

procedure rulea(parm1,parm2,con:longint);
(*****************************************)
(*                                       *)
(*    parm1 <= parm2 + con               *)
(*                                       *)
(*****************************************)
var savez:longint;
begin
  if errcode = 0 then
      begin
          rule[5]:='a';
          savez:=z;
          if max[parm2] < infinity then
                  begin
                    z:=max[parm2]+con;
                    if z < max[parm1] then pushmax(parm1);
                  end;
          z:=min[parm1]-con;
          if z > min[parm2] then pushmin(parm2);
          z:=savez;
          rule[5]:=blk;
        end;
end;

procedure ruleb(parm1,parm2,parm3:longint);
(***********************************************)
(*                                             *)
(*        parm1 <= parm2*parm3                 *)
(*                                             *)
(***********************************************)
var savez:longint;
begin
  if errcode = 0 then
        begin
          rule[5]:='b';
          savez:=z;
          if (max[parm2] > 0) and (max[parm2] < infinity) then
                         begin
                           z:=(min[parm1]+max[parm2]-1) div
max[parm2];
                           if z > min[parm3] then pushmin(parm3);
                         end;
          if (max[parm3] > 0) and (max[parm3] < infinity) then
                         begin
                           z:=(min[parm1]+max[parm3]-1) div
max[parm3];
                           if z > min[parm2] then pushmin(parm2);
                         end;
          rz:=max[parm3];
          rz:=rz*max[parm2];
          if rz < infinity then
             begin
               z:=max[parm3]*max[parm2];
               if z < max[parm1] then pushmax(parm1);
             end;
          z:=savez;
          rule[5]:=blk;
          end;
end;

procedure rulec(parm,con:longint);
(*******************************************)
(*                                         *)
(*            parm <= spectr +con          *)
(*                                         *)
(*         parm is chr or mindeg           *)
(*         and con is 1 or 0, resp.        *)
(*                                         *)
(*******************************************)
begin
  if errcode = 0 then
       begin
          rule[5]:='c';
          if lammax < infinity then
              begin
                 z:=trunk(lammax)+con;
                 if max[nodes] = 0 then z:=0;
                 if z < max[parm] then pushmax(parm);
               end;
          rz:=min[parm]-con;
          pushlammin;
          rule[5]:=blk;
         end;
end;

procedure ruled(parm1,parm2,parm3:longint);
(***************************************************)
```

```
(*                                                 *)
(*        parm1<=parm2+parm3                       *)
(*                                                 *)
(***************************************************)
begin
  if errcode = 0 then
       begin
         rule[5]:='d';
         z:=max[parm2]+max[parm3];
         if z < max[parm1] then pushmax(parm1);
         z:=min[parm1]-max[parm3];
         if z > min[parm2] then pushmin(parm2);
         z:=min[parm1]-max[parm2];
         if z > min[parm3] then pushmin(parm3);
         rule[5]:=blk;
       end;
end;

procedure rulee(parm1,parm2,parm3,con:longint);
(************************************************)
(*                                              *)
(*        parm1>=parm2+parm3+con                *)
(*                                              *)
(************************************************)
begin
  if errcode = 0 then
      begin
        rule[5]:='e';
        z:=min[parm2]+min[parm3]+con;
        if z > min[parm1] then pushmin(parm1);
        if max[parm1] < infinity then
                begin
                  z:=max[parm1]-min[parm3]-con;
                  if max[parm1] = 0 then z:=0;
                  if z < max[parm2] then pushmax(parm2);
                  z:=max[parm1]-min[parm2]-con;
                  if max[parm1] = 0 then z:=0;
                  if z < max[parm3] then pushmax(parm3);
                end;
        rule[5]:=blk;
      end;
end;

procedure rulef(xa,xb:longint);
(*************************************************)
(*                                               *)
(*   computes z:=max(ncov*nind)               *)
(*       (only called from other rules when      *)
(*        proper conditions are met)             *)
(*                                               *)
(*************************************************)
var alp,bet:longint;
```

```
begin
  if errcode = 0 then
    begin
      k:=max[nodes];
      alp:=-1;
      bet:=-1;
      if (2*min[xa] <= k) and (k <= 2*max[xa]) and
         (2*min[xb] <= k) and (k <= 2*max[xb]) then bet:=k div 2
        else
          if (2*max[xb] < k) or (2*min[xa] > k) then
                   if min[xa]+max[xb] <= k then bet:=max[xb]
                                                  else alp:
=min[xa]
                else
                  if max[xa]+min[xb] <= k then alp:=max[xa]
                                                 else bet:
=min[xb];
       if alp > -1 then bet:=k-alp
                   else alp:=k-bet;
       rz:=alp;
       rz:=rz*bet;
       if rz < infinity then z:=alp*bet
                        else z:=infinity;
     end;
end;


end.
```