```
unit pushStack;
     interface
          uses  MemTypes, QuickDraw, OSIntf, ToolIntf, PackIntf,
PrintTraps, PasLibIntf,
               globals,help,text,cmmnds1,pusherr;


           procedure push(x:longint);
           procedure pushmax(param:longint);
           procedure pushmin(param:longint);
           procedure pushlammin;
           procedure pushlammax;


   implementation

procedure dumpRuleStack;
(*********************************)
(*                               *)
(*   Dump the rule stack. First  *)
(*   give user a chance to see it.  *)
(*                               *)
(*********************************)
var i,k:integer;
begin
  primary:=numtables;
  writeln;
  writeln(sysm:1,' The Rule Stack is full and must be emptied.');
  writeln(sysm:1,' To display the current invariant values');
  noescap:=true;
  getAnswer(0);
  write(sysm:1,' To display the rules before purging,');
  getAnswer(1);
  noescap:=false;
  ruletop:=0;
  k:=0;
  if rulestk[1] <> blk5 then
       begin
         ruletop:=1;
         change[1]:=0;
         count[1]:=0;
         rulestk[1]:=' TT- ';
         rulename[1]:=0;
       end;
  for i:=1 to rstkmax do
      if rulestk[i] <> blk5 then k:=k+1
          else
            begin
              ruletop:=ruletop+1;
              if ruletop < i then
```

```
                  begin
                    count[ruletop-1]:=k;
                    rulestk[ruletop]:=blk5;
                    rulename[ruletop]:=rulename[i];
                    change[ruletop]:=change[i];
                  end;
              k:=count[ruletop];
            end;
   count[ruletop]:=k;
   for i:=0 to mxncopiesm1 do undoable[i]:=false;
   xtracopies:=xtracopies+ncopies-1;
   if xtracopies > mxncopiesm1 then xtracopies:=mxncopiesm1;
   ncopies:=1;
end;

procedure push(x:longint);
(*************************************************)
(*                                               *)
(*       pushes an altered invariant onto        *)
(*       the stack.  also places the rule        *)
(*       that caused the change onto the         *)
(*       rule stack.                             *)
(*                                               *)
(*      (if x < 0 then this is a dummy push      *)
(*       just to get a rule number on the rule   *)
(*       stack.)                                 *)
(*                                               *)
(*************************************************)
begin
  if (not trace) and (errcode = 0) then
      begin
        if savesw then
                begin
                  savez:=-1;
                  if copyptr = mxncopiesm1 then copyptr:=0
                                           else copyptr:=copyptr+
1;
                  idnumcopies[copyptr]:=x;
                  undoable[copyptr]:=true;
                  if copyptr = mxncopiesm1 then undoable[0]:
=false
                                           else undoable[copyptr+
1]:=false;
                  numtables:=numtables+1;
                  if ncopies < mxncopies then ncopies:=ncopies+1;
                  if (xtracopies > 0) and
                     (ncopies+xtracopies > mxncopies) then
                          xtracopies:=mxncopies-ncopies;
                  savesw:=false;
                end;
           if cart=lte then i:=1
               else if cart=blk then i:=0
```

```
                  else i:=2;
          if x > 0 then
            if  down[x] = 0 then
                  begin
                    down[x]:=stack;
                    stack:=x;
                        direction[x]:=i;
                  end
                else
                  if i <> direction[x] then direction[x]:=2;
          if ruletop=rstkmax then dumpRuleStack;
          ruletop:=ruletop+1;
          count[ruletop]:=0;
          rulestk[ruletop]:=rule;
          rulename[ruletop]:=x;
          change[ruletop]:=i;
          if perror then pushError(x);
      end;
end;

procedure pushmax(param:longint);
(*******************************************)
(*                                         *)
(*       if upper bound of an invariant    *)
(*       can be improved by 'z', do so.    *)
(*       then place parm on the stack.     *)
(*                                         *)
(*******************************************)
begin
  if (z < max[param]) and (errcode = 0) then
    begin
      if z < min[param] then
         begin
           perror:=true;
           if z < 0 then z:=0;
         end;
      cart:=lte;
      push(param);
      if errcode = 0 then max[param]:=z;
    end;
end;

procedure pushmin(param:longint);
(*******************************************)
(*                                         *)
(*        if lower bound of a parmameter   *)
(*        can be improved, do so. then     *)
(*        place parm on the stack.         *)
(*                                         *)
(*******************************************)
begin
  if (z > min[param]) and (errcode = 0) then
```

```
    begin
      if z > max[param] then
            begin
                  if z > infinity then z:=infinity;
                  if max[param] < infinity then perror:=true
                     else
                       if min[param] < infinity then
                         begin
                           writeln(sysm:1,parameter[param]:6,'
is now larger than "infinity".');
                           writeln(sysm:1,' Results should be
suspect.');
                         end;
               end;
       if min[param] < infinity then
                       begin
                         cart:=blk;
                         push(param);
                         if errcode = 0 then min[param]:=z;
                       end;
      end;
end;

procedure pushlammin;
(*********************************************************)
(*                                                       *)
(*       if lowerbound of spectr can be improved, do so.  *)
(*       then place spectr on the stack.                 *)
(*                                                       *)
(*********************************************************)
begin
  if (rz > lammin) and (errcode = 0) then
     begin
       if rz > infinity then rz:=infinity;
       if rz > lammax then
            begin
              perror:=true;
              if rz > infinity then rz:=infinity;
            end;
       cart:=blk;
       push(spectr);
       if errcode = 0 then lammin:=rz;
     end;
end;

procedure pushlammax;
(*********************************************************)
(*                                                       *)
(*         if upperbound of spectr can be improved,      *)
(*         do so and place spectr on the stack.          *)
(*                                                       *)
(*********************************************************)
```

```
begin
  if (rz < lammax) and (errcode = 0) then
    begin
      if rz < 0 then rz:=0;
      if rz < lammin then
          begin
            perror:=true;
            if rz < 0 then rz:=0;
          end;
      cart:=lte;
      push(spectr);
      if errcode = 0 then lammax:=rz;
    end;
end;


end.
```