```
User: Danilov N.U.

Request id: pr1-320 Printer: pr1

Fri Dec 28 19:34:06 EST 1990
```

38

```
rep.exi.ptr: (p:ptr to art.
                       ide: (txtp:ptr_to_txt)
           Pl=^list:
           Ptr_to_C1=^Cluster;
           List=record
                 it:Ptr_to_cl;
                 nx:Pl
                 end.
           Cluster=record
                     Case me:metasymbol of
                        Alt: (v: (first, second);
                             cptPtr_to_cl);
                        Rep: (n:0..maxv:
                             nextipl);
                        Exi: (e:boolean;
                             epiPtr_to_C1);
                        Con: (p1,p2:Ptr_to_c1);
                        Ptr: (pp:Ptr_to_C1)
                     end;
           intr=0..max;
      Var gramm:Ptr_to_Gr;
          Text:Ptr_to_txt;
          Result:Ptr_to_Cl;
          isintr;
      Function Tree make(var i:intr;T:Ptr_to_txt;g:Ptr_to_Gr;Root:Ptr_to_Cl):bo
lean;
         Var j:intr;
             pt:pl;
              eq:boolean;
         begin
         writeln('Tree_make');
         writeln('i:=',i);
   {
          writeln('current metasymbol is: ',g'.me);
             with t^,g^,root^ do
                Case g^.m of
                alt:begin
                       j:=i;
                       Root^.me:=Alt:
                       new(Root^.cp);
                       if Tree_make(i,t,g^.p1,root^.cp)
                        then
                        begin
                            Root^.v:=first;
                            Tree_make:=true
                        end
                        else
                        begin
                            if Tree_make(i,t,g^.p2,root^.cp)
                             then
                             begin
                                Root^.v:=second;
                                Tree_make: =true
                             end
                             else
                             begin
                                dispose (root^.cp);
                                Tree_make: =false;
                                write(^g, 'Alternativ error in position',i)
                             end (else)
                         end (else)
                    end; (Alt)
                Rep: begin
                writeln('replicator.');
                        Root^.me:=Rep;
                        Root^.n:=0;
                        new(root^.next);
                       pt:=root^.next;
                        while Tree_make(i,t,g^.p,pt^.it) do
```

```
begin
          root^.n:=root^.n+1;
          writeln('n:=',root^.n);
          new (pt^. no).
          pt:=pt^.nx
       end; {whilm'
       pt^.nx:=nil;
       Tree_make:=true;
    end: {rep}
exi:begin
       Root^.me:=exi;
       new(root^.ep);
       if Tree makers +
          then
           root^.e:=true
          else
          begin
             root^.e:=false;
             dispose (root^.ep)
          end; {else}
       Tree_make:=true
    end: {exi}
Con: begin
       Root^.me:=Con;
       new(root^.p1);
       new(root^.p2);
       if Tree_make(i,t,g^.p1,root^.p1) and
          Tree make(i,t,q^.p2,root^.p2)
       then
          Tree_make:=true
       else
       begin
          write(^g,'Concat error in',i)
       end (else)
Ptr:begin
       if Tree_make(i,t,g^.p,root^.pp)
       then
          Tree_make:=true
       else
       begin
          Tree_make:=false;
          dispose (root^.pp)
       end (else)
    end; {Ptr}
Ide: begin
    writeln('Identifier');
    writeln('i:=',i);
    writeln('compare: ');
       root^.me:=Ide;
       eq:=true;
       for j:=1 to g^.txtp^.1 do
       writeln(g^.txtp^.c[j],' with ',t^.c[i+j-1]);
           if (g^.txtp^.c[j]<>t^.c[i+j-1]) or (i+j-1>t^.1)
              then
                 eq:=false;
       end; (of for)
       if eq
           then
              i:=i+g^.txtp^.1;
              Tree_make:=true
           end (then)
           else
           begin
              Tree_make:=false;
              writeln(^g, 'Unknown Identifier at '.i)
```

```
begin
                  end (else)
            end (Ide)
         end: (case)
        writeln('end of make ');
   end; (Tree_make)
Begin (Main)
  new(gramm);
  gramm".mt=Rept
  new(gramm^.p);
   gramm^.p^.mt=Idet
  new(gramm'.p'.txtp);
  oramm^.n^.txtp^.c[1]:='b';
  gramm^.p^.txtp^.1:=1;
  neu(test).
  . . .rotemin's
```

```
text^.c[3]:='b';
   text^.c[4]:='b';
   text^.c[5]:='b';
   text^.c[6]:='b';
   text^.1:=2;
   new(result);
   i : =1;
   clrscri
   writeln(tree_make(i,text,gramm,result));
   if i<>text^.1+1
    then
     rest=false
    else
     res:=true;
    writeln(res);
end. +
```