```
  procedure Enter(var P: Ptr);
  begin  Read(Input, Ch); Write(Output, Ch);
    if Ch <> '.' then
      begin  New(P);
        P↑.Info := Ch; Enter(P↑.LLink); Enter(P↑.RLink)

      end
    else  P := nil
  end { Enter };

  procedure WriteNodes
       (procedure TreeOperation(Start: Ptr); Root: Ptr;
        Title: packed array [M..N: Positive] of Char);
    var
      C: Positive;
  begin
    Writeln(Output);
    for C := M to N do Write(Output, Title[C]);
    Writeln(Output); Writeln(Output);
    TreeOperation(Root); Writeln(Output)
  end { WriteNodes };

begin { Traversal2 }
  Enter(Root); Writeln(Output);
  WriteNodes(PreOrder, Root,
             'Nodes listed in preorder:');
  WriteNodes(InOrder, Root, 'Nodes listed inorder:');
  WriteNodes(PostOrder, Root,
             'Nodes listed in postorder:')
end { Traversal2 } .
```

Produces as results:

```
abc..de..fg...hi..jkl..m..n..

Nodes listed in preorder:

abcdefghijklmn

Nodes listed inorder:

cbedgfaihlkmjn

Nodes listed in postorder:

cegfdbilmknjha
```