PROGRAM : Write a program of depth first search

domains

X=symbol

Y=symbol\*

predicateschild(X,X)

childnode(X,X,Y)

path(X,X,Y)

clauseschild(a,b). /\*b is child of a\*/

child(a,c). /\*c is child of a\*/

child(a,d). /\*d is child of a\*/

child(b,e). /\*b is child of b\*/

child(b,f). /\*f is child of b\*/

child(c,g). /\*g is child ofc\*/

path(A,G,[A|Z]):-/\*to find the path from root to leaf\*/

childnode(A,G,Z).

childnode(A,G,[G]):-/\*to determine whether a node is child of other\*/

child(A,G).

childnode(A,G,[X|L]):-

child(A,X),

childnode(X,G,L).

goal:-path(a,e,L).

L=[“a”,”b”,”e”]

PROGRAM : Program to count number of elements in a list.

domains

x=integer

list=integer\*

predicates

count(list,x)

clauses

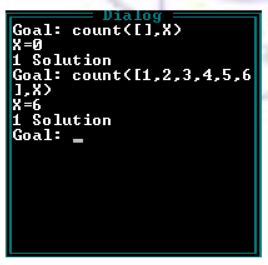
count([],0).

count([\_|T],N):-

count(T,N1),

N=N1+1.

Output:-



PROGRAM : Program to reverse the list .

domains

x=integer

list=integer\*

predicates

append(x,list,list)

rev(list,list)

clauses

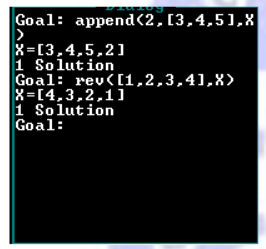
append(X,[],[X]).

append(X,[H|T],[H|T1]):-append(X,T,T1).

rev([],[]).

rev([H|T,rev):-rev(T,L),append(H,L,rev).

Output:-



PROGRAM : Program to count number of elements in a list.

domains

x=integer

list=integer\*

predicates

count(list,x)

clauses

count([],0).

count([\_|T],N):-count(T,N1),N=N1+1.

Output:-

