**Problem Statement**

1. Write a prolog program to implement depth first search.
2. Write a prolog program to implement breadth first search.

**CODE**

1. childnode (a,b)

childnode (a,c)

childnode (c,d)

childnode (c,e)

Path(A,B,[A/L]):-(A,B,L)

child (A,B,[B]-]):-childnode(A,X),child(X,B,L)

**?-path(a,e,l) write(L)**

1. mbps(True,listening):-

bfs([Tree],listening),L,bfs([],[])

bfs([Queue],listening):-

node(empty,(queue)):-(bfs(queue,next);

delete node(Queue,node s),bfs(Queue,listening)

bfs(queue[b/listening]);

hof(empty/queue)->(popper(Queue,[left,right])new queue)

delete first(Queue,Queue node)

append element(Queue,[left,right],new queue)

bfs(New=Queue,List Key)

empty([])

delete first([-1 Jail,T<1])

popper([Head 1],Head)

append element([],[],[])

append element([],List,List)

append element(Head/Jail[],List2,[Head/Request]):-

append element(Jail,List2,result)

**?-mbps(node(k,node(x,node(w,null,null):**

**(null,null)node(y,null,null),L**

**Write(R)**

**OUTPUT**

1. [a, c, e] Yes.
2. [k, x, y, z, w] Yes.