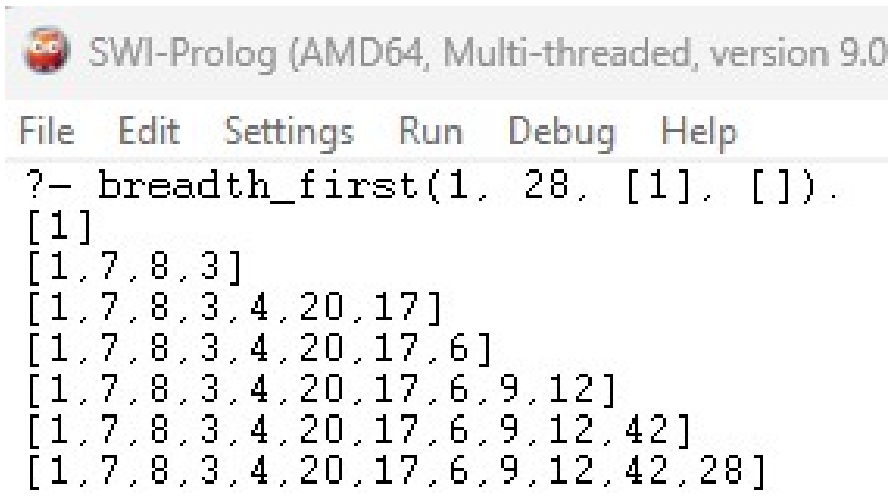
bfs.pl

```
connected(1,7,1).
connected(1,8,1).
connected(1,3,1).
connected(7,4,1).
connected(7,20,1).
connected(7,17,1).
connected(8,6,1).
connected(3,9,1).
connected(3,12,1).
connected(9,19,1).
connected(4,42,1).
connected(20,28,1).
connected(17,10,1).
connected2(X,Y,D) :- connected(X,Y,D).
connected2(X,Y,D) :- connected(Y,X,D).
next_node(Current, Next, Path) :-
    connected2(Current, Next, _),
    not(member(Next, Path)).
breadth_first(Goal, Goal, _,[Goal]).
breadth_first(Start, Goal, Visited, Path) :-
    findall(X,
            (connected2(X,Start,_),not(member(X,Visited)))
            [T|Extend]),
    write(Visited), nl,
    append(Visited, [T|Extend], Visited2),
    append(Path, [T|Extend], [Next|Path2]),
    breadth_first(Next, Goal, Visited2, Path2).
```



```python
def bfs(graph,start,end):
    visited=set()
    queue=[start]
    visited.add(start)
    while queue:
        node=queue.pop()
        print(node,end=" ")
        if node==end:
            break
        for neighbour in graph[node]:
            if neighbour not in visited:
                queue.append(neighbour)
                visited.add(neighbour)
graph={}
n=int(input("Enter number of nodes : "))
for _ in range(n):
    node=input("Enter name of node : ")
    neighbours=input("Enter space seperated neighbour node name : ").split()
    graph[node]=neighbours
start=input("Enter start node : ")
end=input("Enter end node : ")
print("BFS : ")
bfs(graph,start,end)
```

```
Enter number of nodes : 5
Enter name of node : a
Enter space seperated neighbour node name : b c
Enter name of node : b
Enter space seperated neighbour node name : a d
Enter name of node : c
Enter space seperated neighbour node name : a
Enter name of node : d
Enter space seperated neighbour node name : b e
Enter name of node : e
Enter space seperated neighbour node name : d
Enter start node : a
Enter end node : e
BFS :
a c b d e
```