**LABTASK 2**

**NAME: SARIM AMIR**

**SID: 63686**

**Question 2:**

# question 2

graph={

'5':['3','7'],

'3':['2','4'],

'7':['8'],

'2':[],

'4':['8'],

'8':[]

}

visited=set()

def dfs(visited,graph,node):

if node not in visited:

print("No back tracking")

print(node)

visited.add(node)

for neighbor in graph[node]:

if neighbor in visited:

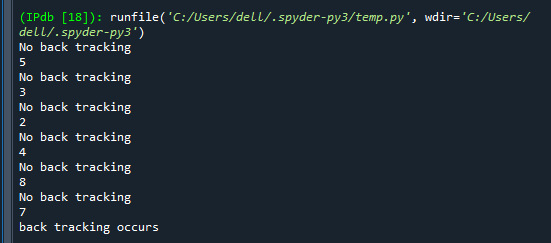
print("back tracking occurs")

break

else:

dfs(visited,graph,neighbor)

dfs(visited,graph,'5')



**Question 1:**

# question 1

king\_graph={

"king":["1 sq forward","1 sq backward","1 sq left","1 sq right"],

"1 sq forward":["1 sq forward","1 sq backward","1 sq left","1 sq right"],

"1 sq backward":["1 sq forward","1 sq backward","1 sq left","1 sq right"],

"1 sq left":["1 sq forward","1 sq backward","1 sq left","1 sq right"],

"1 sq right":["1 sq forward","1 sq backward","1 sq left","1 sq right"]

}

queen\_graph={

"queen":["left","right","forward","backword","diagonal"],

"left":["left","right","forward","backword","diagonal"],

"right":["left","right","forward","backword","diagonal"],

"forward":["left","right","forward","backword","diagonal"],

"backword":["left","right","forward","backword","diagonal"],

"diagonal":["left","right","forward","backword","diagonal"]

}

rook\_graph={

"rook":["moveleft","moveright","moveforward","movebackward"],

"moveleft":["moveleft","moveright","moveforward","movebackward"],

"moveright":["moveleft","moveright","moveforward","movebackward"],

"moveforward":["moveleft","moveright","moveforward","movebackward"],

"movebackward":["moveleft","moveright","moveforward","movebackward"]

}

knight\_graph={

"knight":["2 forward and 90 degree turn","2 backward and 90 degree turn","2 left and 90 degree turn","2 right and 90 degree turn",],

"2 forward and 90 degree turn":["2 forward and 90 degree turn","2 backward and 90 degree turn","2 left and 90 degree turn","2 right and 90 degree turn"],

"2 backward and 90 degree turn":["2 forward and 90 degree turn","2 backward and 90 degree turn","2 left and 90 degree turn","2 right and 90 degree turn"],

"2 left and 90 degree turn":["2 forward and 90 degree turn","2 backward and 90 degree turn","2 left and 90 degree turn","2 right and 90 degree turn"],

"2 right and 90 degree turn":["2 forward and 90 degree turn","2 backward and 90 degree turn","2 left and 90 degree turn","2 right and 90 degree turn"]

}

bishop\_graph={

"bishop":["only diagonal"],

"only diagonal":["only diagonal"]

}

pawn\_graph={

"pawn":["2 sq forward once","1 sq forward","again 1 sq forward"],

"2 sq forward once":["again 1 sq forward"],

"again 1 sq forward":["again 1 sq forward"]

}

visited=[]

queue=[]

def bfs(visited,graph,node):

visited.append(node)

queue.append(node)

while queue:

m=queue.pop(0)

print(m,end=" ")

for neighbor in graph[m]:

if neighbor not in visited:

visited.append(neighbor)

queue.append(neighbor)

print("Rook Moves:")

bfs(visited,rook\_graph,'rook')

print("\n\nKing Moves:")

bfs(visited,king\_graph,'king')

print("\n\nPawn Moves:")

bfs(visited,pawn\_graph,'pawn')

print("\n\nQueen Moves:")

bfs(visited,queen\_graph,'queen')

print("\n\nKnight Moves:")

bfs(visited,knight\_graph,'knight')

print("\n\nBishop Moves:")

bfs(visited,bishop\_graph,'bishop')

**OUTPUT:**

