Practical 7

Name-Huzaif Baig

Class-A\_4

Batch\_B3

Roll No=40

Sub-Daa Lab

Aim: Implement Hamiltonian Cycle using Backtracking.

Code-# Hamiltonian Cycle problem using backtracking

# Example graph (Adjacency Matrix)

G = [

[0, 1, 1, 0, 0],

[1, 0, 1, 1, 0],

[1, 1, 0, 1, 1],

[0, 1, 1, 0, 1],

[0, 0, 1, 1, 0]

]

n = len(G) # number of vertices

x = [0] \* n # path array to store the cycle

def next\_value(k):

"""

Try to assign a valid vertex to x[k]

"""

while True:

x[k] = (x[k] + 1) % n # cycle through 0 to n-1

if x[k] == 0:

return

if G[x[k - 1]][x[k]] != 0:

for j in range(1, k):

if x[j] == x[k]:

break

else:

if (k < n - 1) or (k == n - 1 and G[x[k]][x[0]] != 0):

return

def hamiltonian(k):

"""

Recursive function to generate Hamiltonian cycles

"""

while True:

next\_value(k)

if x[k] == 0:

return

if k == n - 1:

print("Hamiltonian Cycle:", \*x, x[0])

else:

hamiltonian(k + 1)

def main():

x[0] = 0 # starting from vertex 0

print("Hamiltonian Cycles in the graph:")

hamiltonian(1)

# Entry point for OnlineGDB

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output

