**ASSIGNMENT 01**

**NAME: SAMIHA SOFRANA AZIM**

**ID: 17101290**

**Level01:**

import queue

def minEdgeBFS(edges, u, v, n):

visited = [0] \* n

distance = [0] \* n

Q = queue.Queue()

distance[u] = 0

Q.put(u)

visited[u] = True

while (not Q.empty()):

x = Q.get()

for i in range(len(edges[x])):

if (visited[edges[x][i]]):

continue

distance[edges[x][i]] = distance[x] + 1

Q.put(edges[x][i])

visited[edges[x][i]] = 1

return distance[v]

def addEdge(edges, u, v):

edges[u].append(v)

edges[v].append(u)

# Test Code

if \_\_name\_\_ == '\_\_main\_\_':

n = 9

edges = [[] for i in range(n)]

addEdge(edges, 0, 1)

addEdge(edges, 0, 2)

addEdge(edges, 0,3)

addEdge(edges, 1, 3)

addEdge(edges, 1,4)

addEdge(edges, 2, 3)

addEdge(edges, 3,5)

addEdge(edges, 3, 6)

addEdge(edges, 4,8)

addEdge(edges, 4, 7)

addEdge(edges, 5, 6)

addEdge(edges, 6, 7)

addEdge(edges, 7, 8)

u = 0

v = 6

print(minEdgeBFS (edges, u, v, n))

**Level2:**

import queue

def minEdgeBFS(edges, u, v, n):

visited = [0] \* n

distance = [0] \* n

Q = queue.Queue()

distance[u] = 0

Q.put(u)

visited[u] = True

while (not Q.empty()):

x = Q.get()

for i in range(len(edges[x])):

if (visited[edges[x][i]]):

continue

distance[edges[x][i]] = distance[x] + 1

Q.put(edges[x][i])

visited[edges[x][i]] = 1

return distance[v]

def addEdge(edges, u, v):

edges[u].append(v)

edges[v].append(u)

# Test Code

if \_\_name\_\_ == '\_\_main\_\_':

n = 9

edges = [[] for i in range(n)]

addEdge(edges, 0, 1)

addEdge(edges, 0, 2)

addEdge(edges, 0, 3)

addEdge(edges, 1, 3)

addEdge(edges, 1, 4)

addEdge(edges, 2, 3)

addEdge(edges, 3, 5)

addEdge(edges, 4, 8)

addEdge(edges, 4, 7)

addEdge(edges, 5, 6)

addEdge(edges, 6, 7)

addEdge(edges, 7, 8)

u = 5

v = 7

print(minEdgeBFS(edges, u, v, n))

a=minEdgeBFS(edges,u,v,n))

u = 3

v = 7

print(minEdgeBFS(edges, u, v, n))

b=minEdgeBFS(edges,u,v,n))

if a<b:

print(“Nora”)

else:

print(“Lara”)

**Level3:**

  import queue

  def minEdgeBFS(edges, u, v, n):

    visited = [0] \* n

    distance = [0] \* n

    Q = queue.Queue()

    distance[u] = 0

    Q.put(u)

    visited[u] = True

    while (not Q.empty()):

      x = Q.get()

      for i in range(len(edges[x])):

        if (visited[edges[x][i]]):

          continue

        distance[edges[x][i]] = distance[x] + 1

        Q.put(edges[x][i])

        visited[edges[x][i]] = 1

    return distance[v]

  def addEdge(edges, u, v):

    edges[u].append(v)

    edges[v].append(u)

  # Test Code

  if \_\_name\_\_ == '\_\_main\_\_':

    n = 10

    edges = [[] for i in range(n)]

    addEdge(edges, 0, 1)

    addEdge(edges, 0, 2)

    addEdge(edges, 0, 3)

    addEdge(edges, 1, 3)

    addEdge(edges, 1, 4)

    addEdge(edges, 2, 3)

    addEdge(edges, 3, 5)

    addEdge(edges, 4, 7)

    addEdge(edges, 4, 8)

    addEdge(edges, 5, 6)

    addEdge(edges, 6, 7)

    addEdge(edges, 6, 9)

    addEdge(edges, 7, 8)

    addEdge(edges, 8, 9)

    u = 0

    v = 9

    print(minEdgeBFS(edges, u, v, n))

k1=(minEdgeBFS(edges, u, v, n))

    u = 1

    v = 9

    print(minEdgeBFS(edges, u, v, n))

k2=(minEdgeBFS(edges, u, v, n))

    u = 3

    v = 9

    print(minEdgeBFS(edges, u, v, n))

k3=(minEdgeBFS(edges, u, v, n))

    u = 5

    v = 9

    print(minEdgeBFS(edges, u, v, n))

k4=(minEdgeBFS(edges, u, v, n))

    u = 7

    v = 9

    print(minEdgeBFS(edges, u, v, n))

k5=(minEdgeBFS(edges, u, v, n))

# Sort

import sys

A = [k1,k2,k3,k4,k5]

  for i in range(len(A)):

    min\_idx = i

    for j in range(i+1, len(A)):

        if A[min\_idx] > A[j]:

            min\_idx = j

    A[i], A[min\_idx] = A[min\_idx], A[i]

# Test code

print(“Sorted Array”)

for i in range(len(A)):

    print("%d" %A[i]),

print(A[0])