TASK:3

NAME: MOHAMED ASHRAF

Hash map in python?

hash\_map = {}

hash\_map['apple'] = 1

hash\_map['banana'] = 2

hash\_map['orange'] = 3

print(hash\_map['apple']) # Output: 1

print(hash\_map['orange']) # Output: 3

if 'banana' in hash\_map:

print("Found 'banana' in the hash map!")

else:

print("Key not found.")

hash\_map['apple'] = 5

print(hash\_map['apple']) # Output: 5

del hash\_map['orange']

for key, value in hash\_map.items():

print(f"Key: {key}, Value: {value}")

# Output:

# Key: apple, Value: 5

# Key: banana, Value: 2

Implementation of graph in python?

class Graph:

def \_\_init\_\_(self):

self.graph\_dict = {}

def add\_vertex(self, vertex):

if vertex not in self.graph\_dict:

self.graph\_dict[vertex] = []

def add\_edge(self, vertex1, vertex2):

if vertex1 in self.graph\_dict and vertex2 in self.graph\_dict:

self.graph\_dict[vertex1].append(vertex2)

self.graph\_dict[vertex2].append(vertex1) # For an undirected graph, you can omit this line.

def get\_neighbors(self, vertex):

return self.graph\_dict.get(vertex, [])

def \_\_str\_\_(self):

graph\_str = ""

for vertex, neighbors in self.graph\_dict.items():

graph\_str += f"{vertex}: {neighbors}\n"

return graph\_str

# Example usage

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

my\_graph = Graph()

# Adding vertices

my\_graph.add\_vertex("A")

my\_graph.add\_vertex("B")

my\_graph.add\_vertex("C")

my\_graph.add\_vertex("D")

# Adding edges

my\_graph.add\_edge("A", "B")

my\_graph.add\_edge("A", "C")

my\_graph.add\_edge("B", "D")

my\_graph.add\_edge("C", "D")

print(my\_graph)

# Output:

# A: ['B', 'C']

# B: ['A', 'D']

# C: ['A', 'D']

# D: ['B', 'C']