1.1) Write the following function’s body. A nested dictionary is passed as parameter. You need

to print all keys with their depth.

Sample Input:

a = {

“key1”: 1,

“”key2”: {

“key3”: 1,

“key4”: {

“key5”: 4

“user”: person\_b,

}

}

}

Sample Output:

key1 1

key2 1

key3 2

key4 2

key5 3

def print\_depth(data):

# Write function body

You may write additional function.

1.2) Write a new function with same functionality from Question 1.1, but it should be able to

handle any Python object in addition to a dictionary.

Sample Input:

class Person(object):

def \_\_init\_\_(self, first\_name, last\_name, father):

self.first\_name = first\_name

self.last\_name = last\_name

self.father = father

person\_a = Person(“User”, “1”, none)

person\_b = Person(“User”, “2”, person\_a)

a = {

“key1”: 1,

”key2”: {

“key3”: 1,

“key4”: {

“key5”: 4,

“user”: person\_b,

}

},

}

Sample Output:

key1 1

key2 1

key3 2

key4 2

key5 3

user: 3

first\_name: 4

last\_name: 4

father: 4

first\_name: 5

last\_name: 5

father: 5

def print\_depth\_extended(data):

# Write function body

You may write additional function.

2) Write following functions body. 2 Nodes are passed as parameter. You need to find Least

Common Ancestor and print its value. Node structure are as following:

class Node{

value;

parent;

}

1

2

3

7

6

5

4

8

9

Ancestor Definition:

1) Any node falls under parent chain till root node.

2) A node is an ancestor of itself.

For example: if we consider Node 7 it’s ancestors will be 1, 3, and 7.

All nodes values are unique for this tree.

You function needs to find least common ancestor (closest common ancestor). For an example

for the tree image,

if 6 and 7 passed to lca it should return 3

if 3 and 7 passed to lca it should return 3

def lca(node1, node2):

# Write function body

You may write additional function.

Explain the Runtime and Memory requirements for your solution.

3) Given an input string (s) and a pattern (p), implement wildcard pattern matching with support

for '?' and '\*'.

'?' Matches any single character.

'\*' Matches any sequence of characters (including the empty sequence).

The matching should cover the entire input string (not partial).

Note:

s could be empty and contains only lowercase letters a-z.

p could be empty and contains only lowercase letters a-z, and characters like ? or \*.

Example:

Input:

s = "aa"

p = "\*"

Output: true

Input:

s = "cb"

p = "?a"

Output: false

Input:

s = "adceb"

p = "\*a\*b"

Output: true

def isMatch(self, s: str, p: str) -> bool:

# Write function body

Explain the Runtime and Memory requirements for your solution