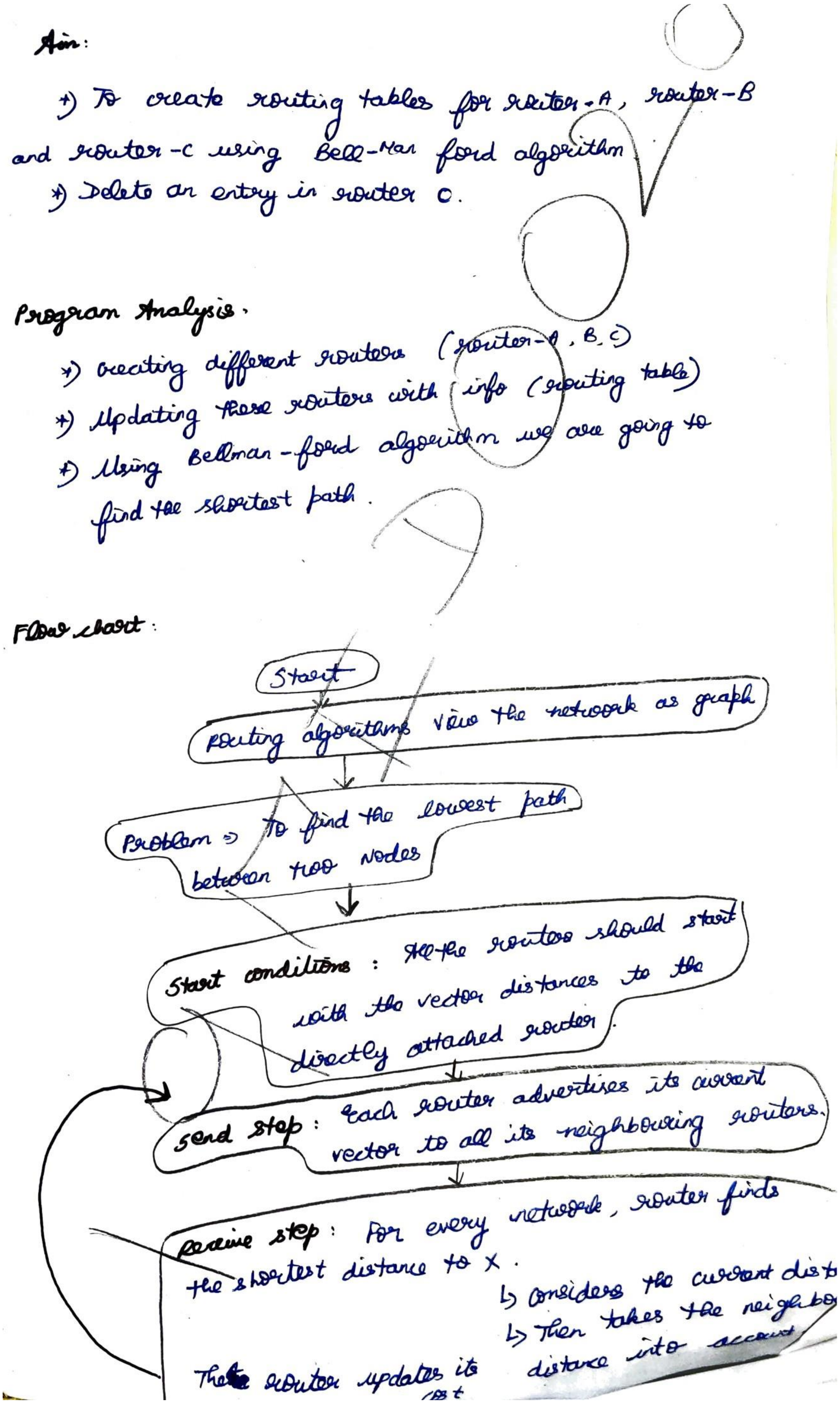
DCN Lab-Fat **Prashanth.S**
(19MID0020)

Question-1



Aim:

*) To create routing tables for router-A, router-B and router-C using Bell-Man ford algorithm

*) Delete an entry in router C.

Program Analysis.

*) Creating different routers (router-A, B, C)

*) Updating these routers with info (routing table)

*) Using Bellman-ford algorithm we are going to find the shortest path.

Flow chart:

Start

Routing algorithms view the network as graph

Problem ⇒ To find the lowest path between two Nodes

Start conditions : ~~the~~ the routers should start with the vector distances to the directly attached router.

Send step : Each router advertises its current vector to all its neighbouring routers.

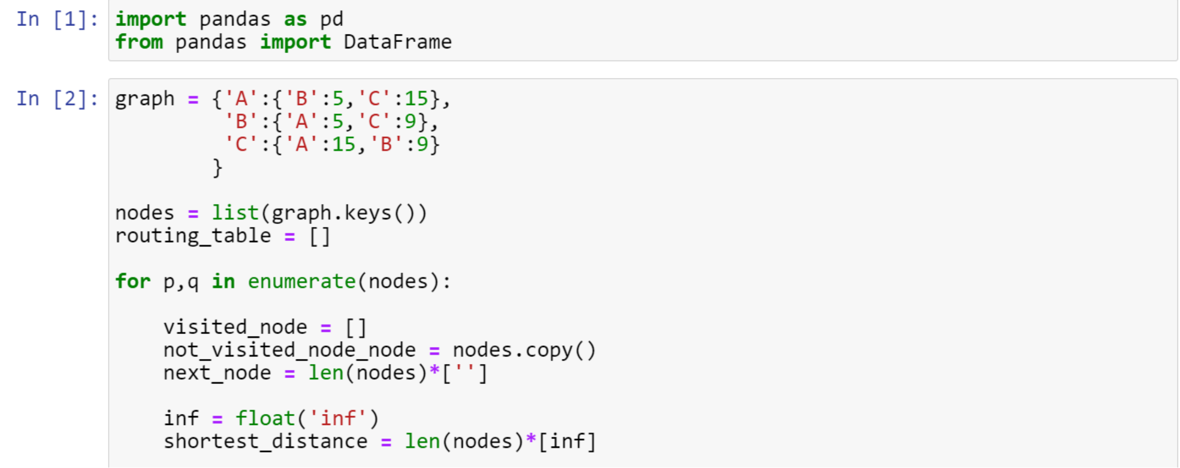
Receive step : For every network, router finds the shortest distance to X.

↳ Considers the current dist

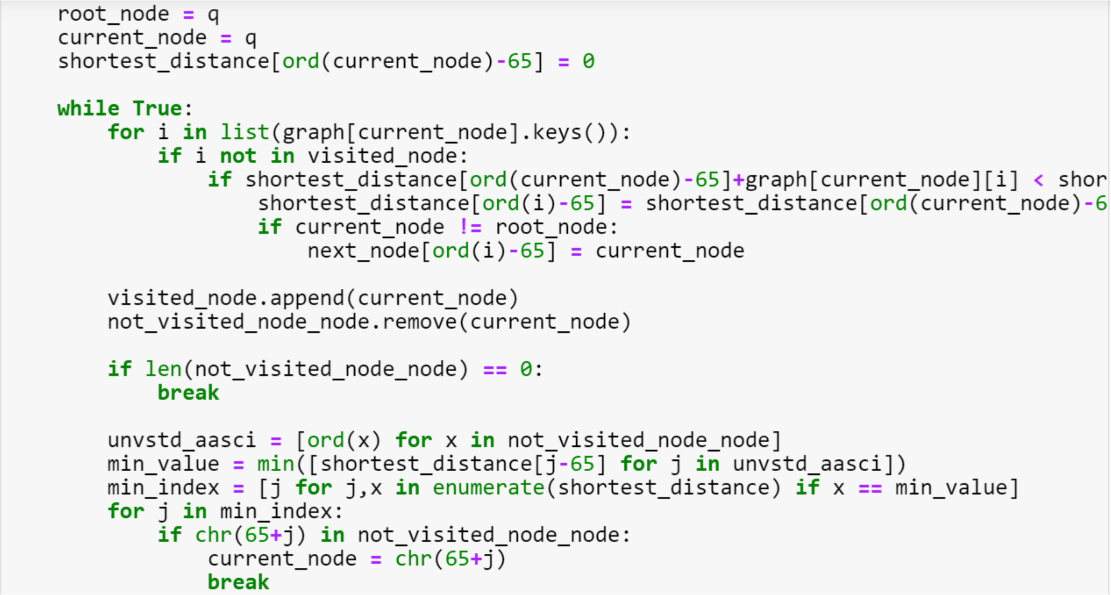
↳ Then takes the neighbo distance into account

The ~~te~~ router updates its cost

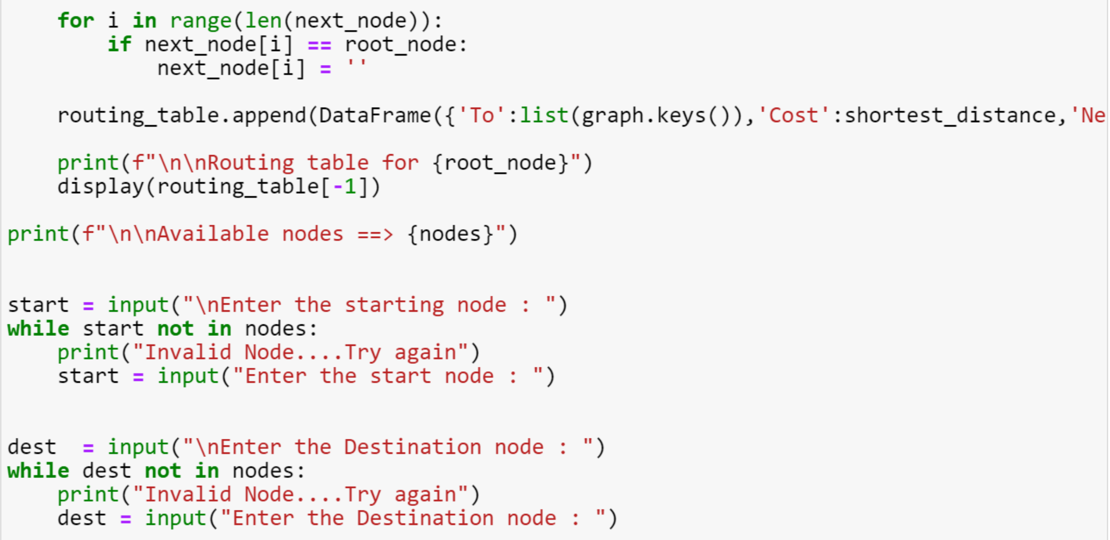
Code Snippet



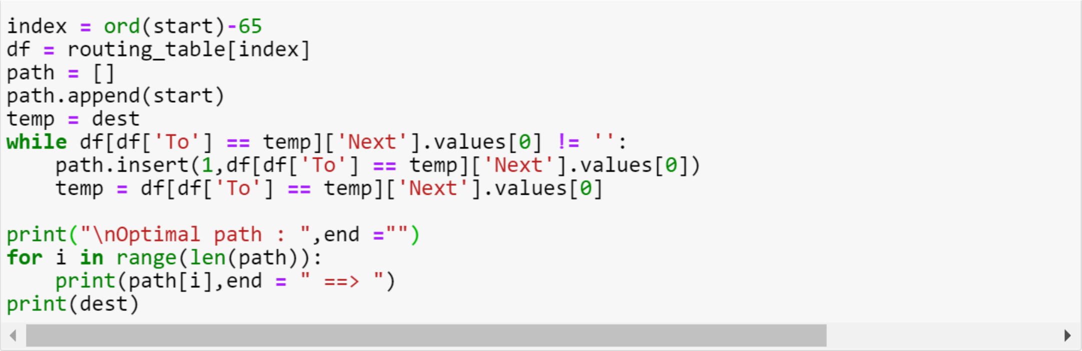
```
In [1]: import pandas as pd
        from pandas import DataFrame

In [2]: graph = {'A':{'B':5,'C':15},
                 'B':{'A':5,'C':9},
                 'C':{'A':15,'B':9}
                }

        nodes = list(graph.keys())
        routing_table = []

        for p,q in enumerate(nodes):

            visited_node = []
            not_visited_node_node = nodes.copy()
            next_node = len(nodes)*['']

            inf = float('inf')
            shortest_distance = len(nodes)*[inf]
```



```
    root_node = q
    current_node = q
    shortest_distance[ord(current_node)-65] = 0

    while True:
        for i in list(graph[current_node].keys()):
            if i not in visited_node:
                if shortest_distance[ord(current_node)-65]+graph[current_node][i] < shor
                    shortest_distance[ord(i)-65] = shortest_distance[ord(current_node)-6
                    if current_node != root_node:
                        next_node[ord(i)-65] = current_node

        visited_node.append(current_node)
        not_visited_node_node.remove(current_node)

        if len(not_visited_node_node) == 0:
            break

        unvstd_aasci = [ord(x) for x in not_visited_node_node]
        min_value = min([shortest_distance[j-65] for j in unvstd_aasci])
        min_index = [j for j,x in enumerate(shortest_distance) if x == min_value]
        for j in min_index:
            if chr(65+j) in not_visited_node_node:
                current_node = chr(65+j)
                break
```

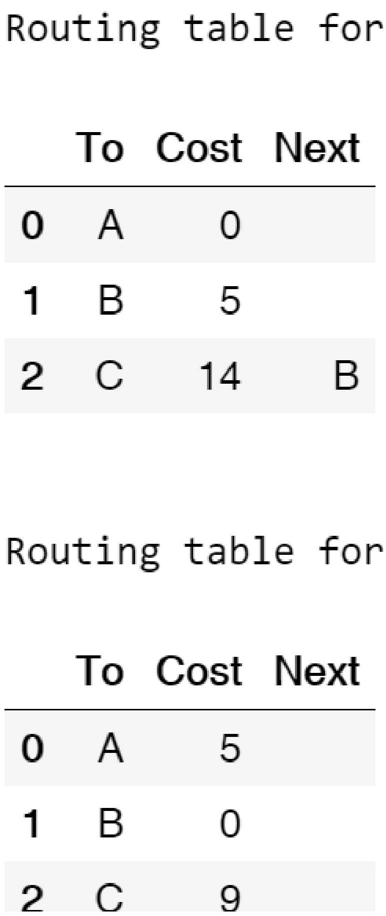


```
    for i in range(len(next_node)):
        if next_node[i] == root_node:
            next_node[i] = ''

    routing_table.append(DataFrame({'To':list(graph.keys()),'Cost':shortest_distance,'Ne

    print(f"\n\nRouting table for {root_node}")
    display(routing_table[-1])

print(f"\n\nAvailable nodes ==> {nodes}")


start = input("\nEnter the starting node : ")
while start not in nodes:
    print("Invalid Node....Try again")
    start = input("Enter the start node : ")


dest  = input("\nEnter the Destination node : ")
while dest not in nodes:
    print("Invalid Node....Try again")
    dest = input("Enter the Destination node : ")
```



```python
index = ord(start)-65
df = routing_table[index]
path = []
path.append(start)
temp = dest
while df[df['To'] == temp]['Next'].values[0] != '':
    path.insert(1,df[df['To'] == temp]['Next'].values[0])
    temp = df[df['To'] == temp]['Next'].values[0]

print("\nOptimal path : ",end ="")
for i in range(len(path)):
    print(path[i],end = " ==> ")
print(dest)
```

Test-case 1

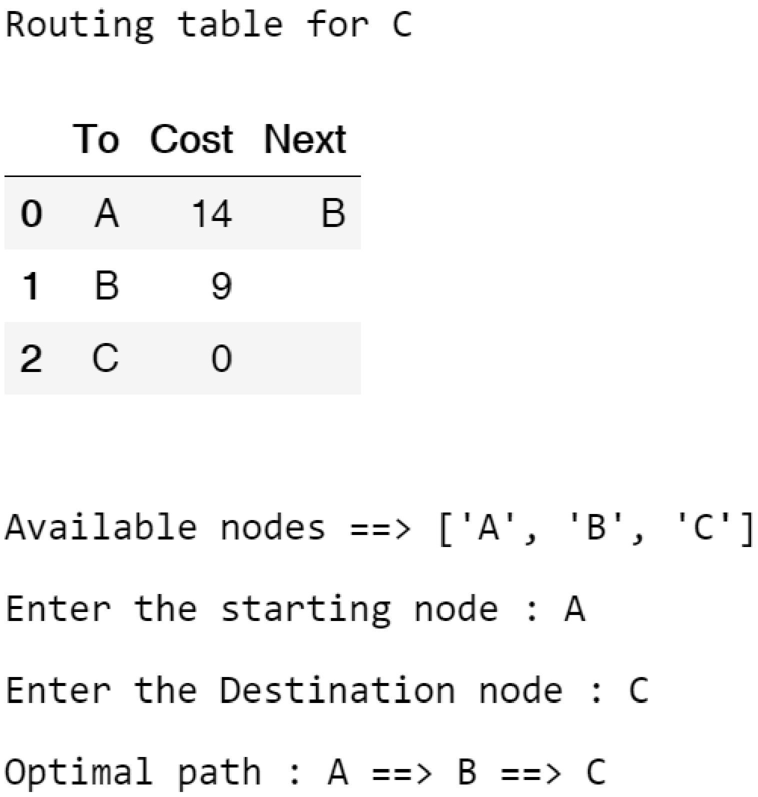


```
Routing table for A
```

	To	Cost	Next
0	A	0	
1	B	5	
2	C	14	B

```
Routing table for B
```

	To	Cost	Next
0	A	5	
1	B	0	
2	C	9	

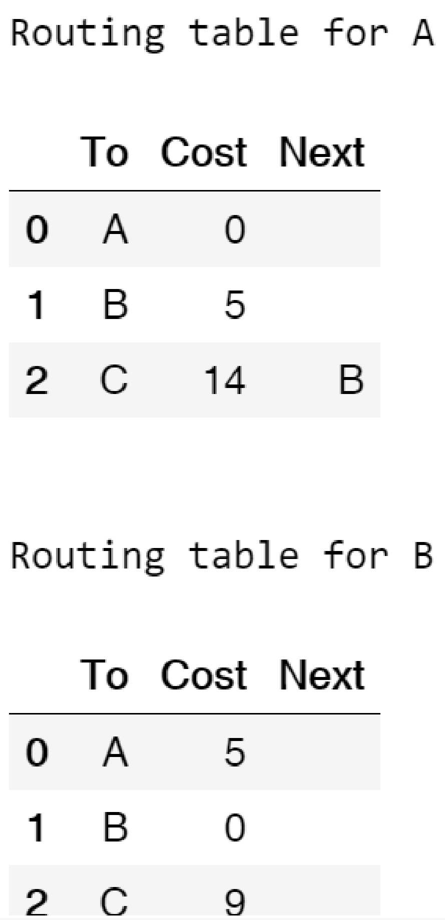


```
Routing table for C
```

	To	Cost	Next
0	A	14	B
1	B	9	
2	C	0	

```
Available nodes ==> ['A', 'B', 'C']

Enter the starting node : A

Enter the Destination node : C

Optimal path : A ==> B ==> C
```

Test-case 2

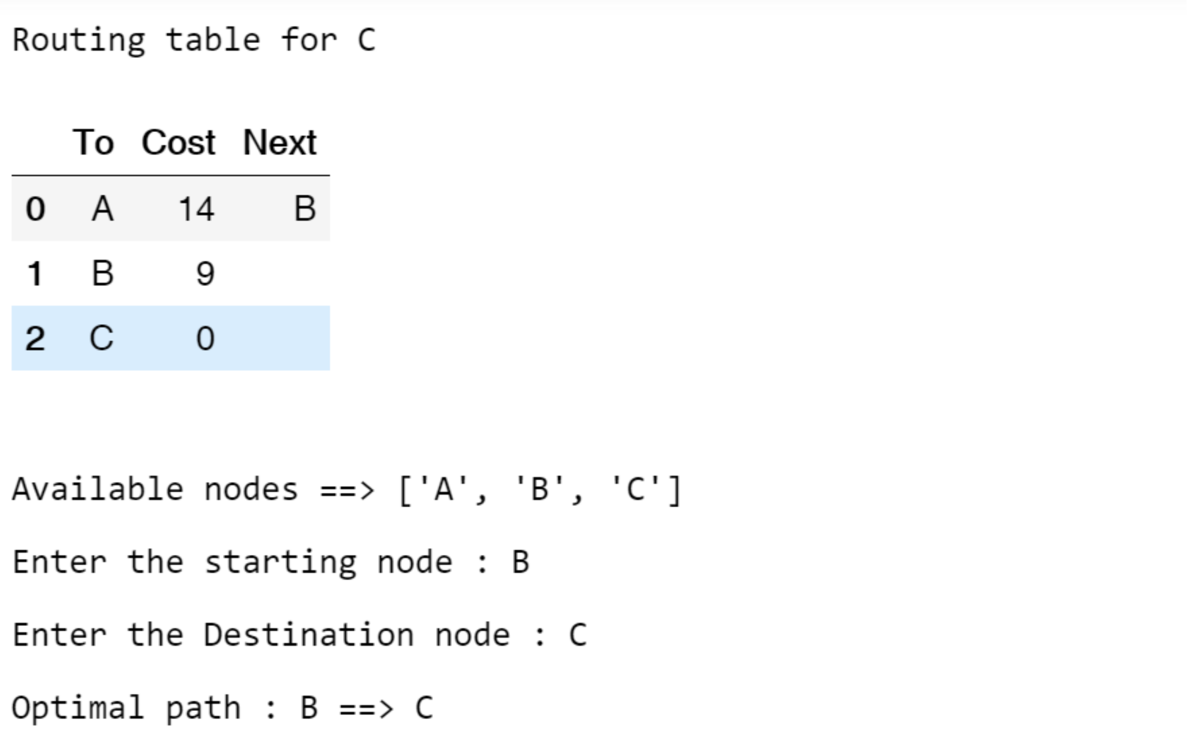


```
Routing table for A
```

	To	Cost	Next
0	A	0	
1	B	5	
2	C	14	B

```
Routing table for B
```

	To	Cost	Next
0	A	5	
1	B	0	
2	C	9	

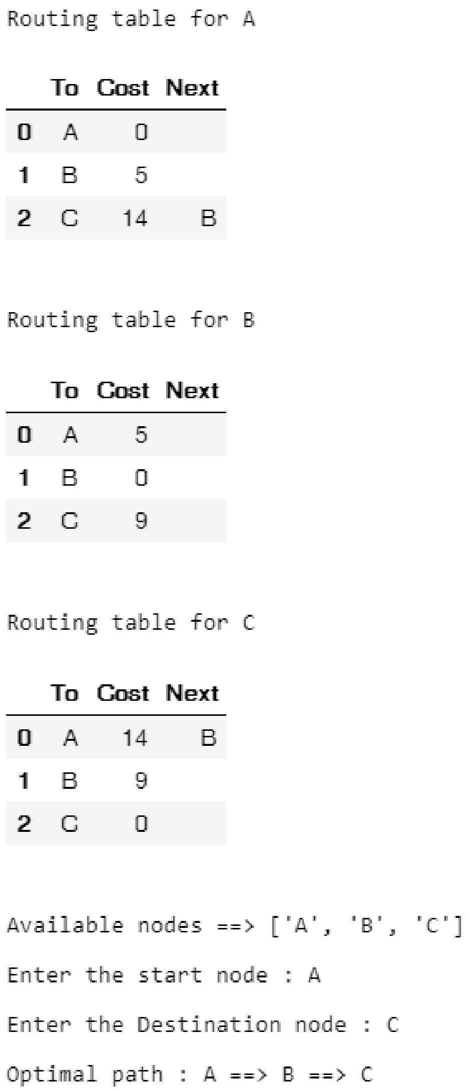


```
Routing table for C
```

	To	Cost	Next
0	A	14	B
1	B	9	
2	C	0	

```
Available nodes ==> ['A', 'B', 'C']

Enter the starting node : B

Enter the Destination node : C

Optimal path : B ==> C
```

Output



```
Routing table for A
```

	To	Cost	Next
0	A	0	
1	B	5	
2	C	14	B

```
Routing table for B
```

	To	Cost	Next
0	A	5	
1	B	0	
2	C	9	

```
Routing table for C
```

	To	Cost	Next
0	A	14	B
1	B	9	
2	C	0	

```
Available nodes ==> ['A', 'B', 'C']

Enter the start node : A

Enter the Destination node : C

Optimal path : A ==> B ==> C
```

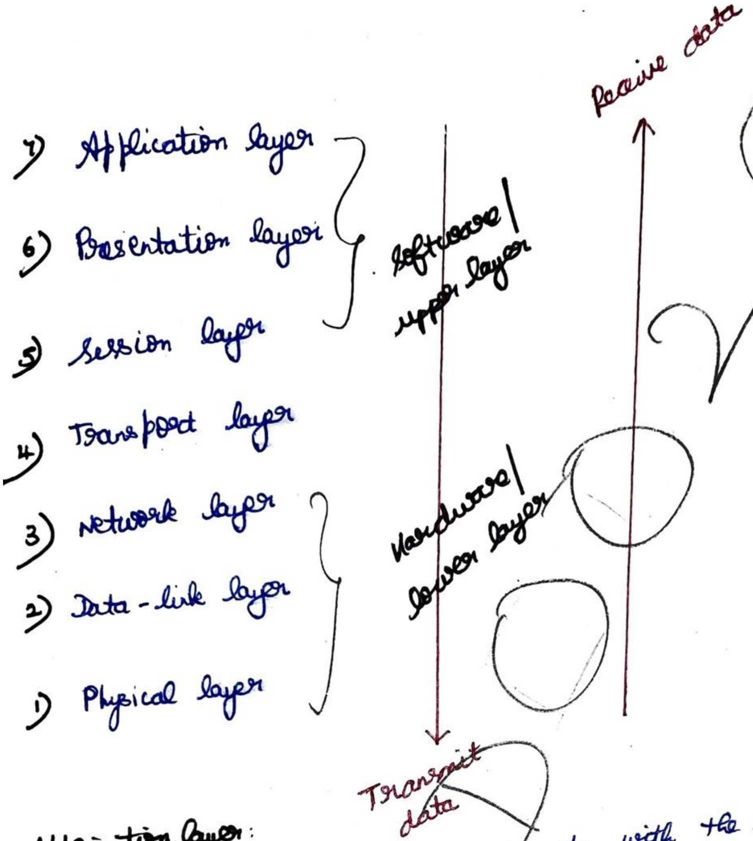
Actual Code

```
import pandas as pd
from pandas import DataFrame

graph = {'A': {'B': 5, 'C': 15},
        'B': {'A': 5, 'C': 9},
        'C': {'A': 15, 'B': 9}
        }

nodes = list(graph.keys())
routing_table = []

for p, q in enumerate(nodes):

    visited_node = []
    not_visited_node_node = nodes.copy()
    next_node = len(nodes) * ['']

    inf = float('inf')
    shortest_distance = len(nodes) * [inf]

    root_node = q
    current_node = q
    shortest_distance[ord(current_node) - 65] = 0

    while True:
        for i in list(graph[current_node].keys()):
            if i not in visited_node:
                if shortest_distance[ord(current_node) - 65] + graph[current_node][i] < shortest_distance[ord(i) - 65]:
                    shortest_distance[ord(
                        i) - 65] = shortest_distance[ord(current_node) - 65] + graph[current_node][i]
                    if current_node != root_node:
                        next_node[ord(i) - 65] = current_node

        visited_node.append(current_node)
        not_visited_node_node.remove(current_node)

        if len(not_visited_node_node) == 0:
            break

        unvstd_aasci = [ord(x) for x in not_visited_node_node]
        min_value = min([shortest_distance[j - 65] for j in unvstd_aasci])
        min_index = [j for j, x in enumerate(
            shortest_distance) if x == min_value]
        for j in min_index:
            if chr(65 + j) in not_visited_node_node:
                current_node = chr(65 + j)
                break

    for i in range(len(next_node)):
        if next_node[i] == root_node:
            next_node[i] = ''

    routing_table.append(DataFrame(
        {'To': list(graph.keys()), 'Cost': shortest_distance, 'Next':
next_node}))

    print(f"\n\nRouting table for {root_node}")
    display(routing_table[-1])

print(f"\n\nAvailable nodes ==> {nodes}")



start = input("\nEnter the starting node : ")
```

```
while start not in nodes:
    print("Invalid Node....Try again")
    start = input("Enter the start node : ")



dest = input("\nEnter the Destination node : ")
while dest not in nodes:
    print("Invalid Node....Try again")
    dest = input("Enter the Destination node : ")



index = ord(start) - 65
df = routing_table[index]
path = []
path.append(start)
temp = dest
while df[df['To'] == temp]['Next'].values[0] != '':
    path.insert(1, df[df['To'] == temp]['Next'].values[0])
    temp = df[df['To'] == temp]['Next'].values[0]

print("\nOptimal path : ", end="")
for i in range(len(path)):
    print(path[i], end=" ==> ")
print(dest)
```

Question-2

OSI Layers



Application layer:

This is the layer which interacte with the user, and these applications can access the network services.

Presentation layer:

This is the layer where the data is completely usable format and where data encryption occurs.

Session layer:

This is the layer where the connection is responsible for controlling ports and sessions.

Transport layer:

This is the layer where the transmission protocol

TCP ⇒ Transmission Control Protocol

UDP ⇒ User Data-gram Protocol.

Network layer:

This is the layer which decides the physical path should be taken for better broadcasting.

Data-link layer:

This is the layer where the delivery and the error checking mechanism take place. Routers and switches ~~plays~~ are a crutial component.

Physical layer:

This is the layer where the raw bits stream over the physical medium.

Why OSI layer ?

*) OSI layer is very important for trouble shooting the network problems.

*) It creates devices and softwares that can communicate with products from any other vendor.

*) OSI layer is essential for developping secured mindset for cloud adoption

How OSI layer ?

Upper layer *) Mostly implemented in software.
*) Users can interact from one-end to another by using the interaction between the application layer.

Lower layer *) this layer relates to transport the data, Physical and data-link layers aslo implemented in software & hardwa