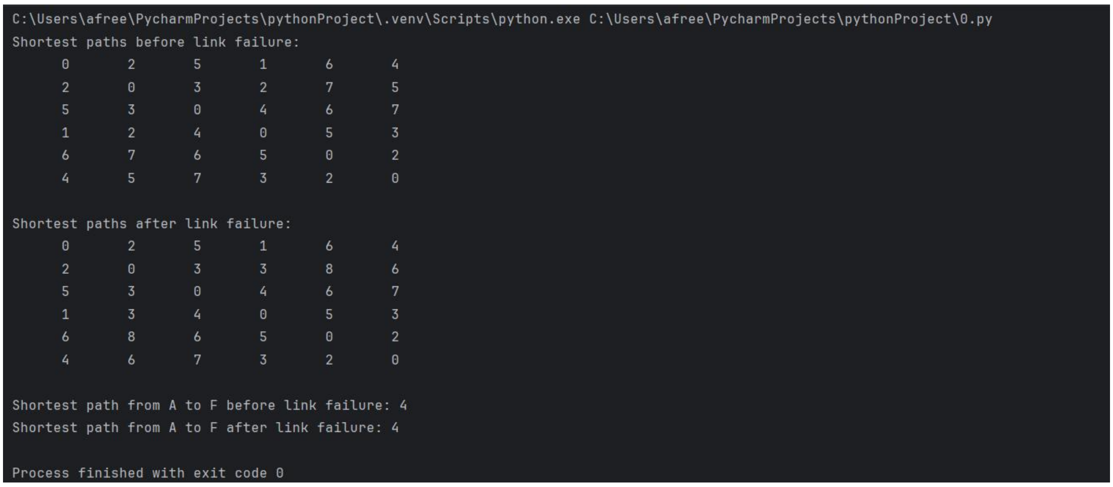
Exercise173:-13.Write a Program to implement Floyd's Algorithm to calculate the shortest paths between all pairs of routers. Simulate a change where the link between Router B and Router D fails. Update the distance matrix accordingly. Display the shortest path from Router A to Router F before and after the link failure.

Program:-

```
import sys
inf = sys.maxsize
dist_matrix = [
    [0, 2, inf, 1, inf, inf],
    [2, 0, 3, 2, inf, inf],
    [inf, 3, 0, 4, 6, inf],
    [1, 2, 4, 0, 5, 3],
    [inf, inf, 6, 5, 0, 2],
    [inf, inf, inf, 3, 2, 0]
]
n = len(dist_matrix)
def floyd_warshall(dist):
    for k in range(n):
        for i in range(n):
            for j in range(n):
                if dist[i][j] > dist[i][k] + dist[k][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]
    return dist
def print_matrix(matrix):
    for row in matrix:
        print(" ".join(f"{(inf if x == sys.maxsize else x):7}" for x in row))
    print()
shortest_paths_before = floyd_warshall([row[:] for row in dist_matrix])
print("Shortest paths before link failure:")
print_matrix(shortest_paths_before)
dist_matrix[1][3] = inf
dist_matrix[3][1] = inf
shortest_paths_after = floyd_warshall([row[:] for row in dist_matrix])
print("Shortest paths after link failure:")
print_matrix(shortest_paths_after)
def shortest_path_distance(matrix, start, end):
    return matrix[start][end]
A, F = 0, 5
shortest_path_before = shortest_path_distance(shortest_paths_before, A, F)
shortest_path_after = shortest_path_distance(shortest_paths_after, A, F)
print(f"Shortest path from A to F before link failure: {shortest_path_before}")
print(f"Shortest path from A to F after link failure: {shortest_path_after}")
```

Output:-



```
C:\Users\afree\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\afree\PycharmProjects\pythonProject\0.py
Shortest paths before link failure:
      0       2       5       1       6       4
      2       0       3       2       7       5
      5       3       0       4       6       7
      1       2       4       0       5       3
      6       7       6       5       0       2
      4       5       7       3       2       0

Shortest paths after link failure:
      0       2       5       1       6       4
      2       0       3       3       8       6
      5       3       0       4       6       7
      1       3       4       0       5       3
      6       8       6       5       0       2
      4       6       7       3       2       0

Shortest path from A to F before link failure: 4
Shortest path from A to F after link failure: 4

Process finished with exit code 0
```

Time complexity:-O(n^3)