Implement All Pair Shortest paths problem using Floyd's algorithm.

Code:

```
#include <stdio.h>
#include <limits.h>

#define INF INT_MAX  // Set infinity as the maximum integer value

// Function to implement Floyd-Warshall algorithm
void floydWarshall(int graph[][10], int dist[][10], int n) {
    // Initialize the distance matrix with graph values
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (graph[i][j] != 0) {
                dist[i][j] = graph[i][j];  // Set the direct distances
            } else if (i == j) {
                dist[i][j] = 0;  // Distance from node to itself is 0
            } else {
                dist[i][j] = INF;  // No direct edge means distance is infinity
            }
        }
    }

    // Floyd-Warshall algorithm
    for (int k = 0; k < n; k++) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (dist[i][k] != INF && dist[k][j] != INF && dist[i][j] > dist[i][k] + dist[k][j]) {
                    dist[i][j] = dist[i][k] + dist[k][j];
                }
            }
        }
    }
}

// Function to print the shortest distances
void printSolution(int dist[][10], int n) {
    printf("Shortest distances between every pair of vertices:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (dist[i][j] == INF) {
                printf("%7s ", "INF");
            } else {
```

```
                printf("%7d ", dist[i][j]);
            }
        }
        printf("\n");
    }
}

int main() {
    int n;

    // Input number of vertices
    printf("Enter the number of vertices: ");
    scanf("%d", &n);

    // Declare the graph and the distance matrix
    int graph[10][10], dist[10][10];

    // Input the adjacency matrix
    printf("Enter the adjacency matrix (use 0 for no edge, or a positive integer for the weight):\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    // Call Floyd-Warshall algorithm
    floydWarshall(graph, dist, n);

    // Print the solution (shortest path matrix)
    printSolution(dist, n);

    return 0;
}
```

Output:

```
Enter the number of vertices: 4
Enter the adjacency matrix (use 0 for no edge, or a positive integer for the weight):
0 0 3 0
2 0 0 0
0 7 0 1
6 0 0 0
Shortest distances between every pair of vertices:
      0      10       3       4
      2       0       5       6
      7       7       0       1
      6      16       9       0

Process returned 0 (0x0)   execution time : 39.189 s
Press any key to continue.
```