Assignment No: 03

Name: Omkar Kulkarni
Class: SY-1
Batch: C
PRN: B25CE2012

Title:

Load Balancing:
For example, imagine you have a set of servers that handle requests for a web application. The key to load balancing is using the hash value of a client's IP address or a request ID to determine which server should handle the request. The hash function is typically designed so that the data is evenly distributed across the servers, ensuring that no single server is overloaded.

Write a program of a load balancing system using hashing, where a basic hash table is used for mapping incoming requests to a set of servers.

Program:

```
#include <iostream>
using namespace std;

class LoadBalancer {
    int n_servers;          // Total number of servers
    int hash_table[100];    // Server storage (0 = empty)

public:
    // Constructor to initialize server table
    LoadBalancer(int n) {
        n_servers = n;
        for (int i = 0; i < n_servers; i++) {
            hash_table[i] = 0;
        }
    }

    // Basic hash function
    int hashFunction(int val) {
        return val % n_servers;
    }
    // Convert IP string into integer sum (e.g., 192.168.1.5 -> 366)
    int processIP(char ip[]) {
        int sum = 0, num = 0;
        for (int i = 0; ip[i] != '\0'; i++) {
            if (ip[i] == '.') {
                sum += num;    // Add current part when dot found
                num = 0;       // Reset for next number
            } else {
                num = num * 10 + (ip[i] - '0'); // Build number
            }
        }
        sum += num; // Add the last part
        return sum;
    }

    // Insert IP request using linear probing
    void insert(char ip[]) {
        int val = processIP(ip);       // Convert IP -> sum of parts
        int idx = hashFunction(val);   // Hash value
        int start = idx;               // Remember start position

        // Collision handling using linear probing
        while (hash_table[idx] != 0) {
            idx = (idx + 1) % n_servers;
            if (idx == start) {
                cout << "All servers are busy! Cannot assign " << ip
<< endl;
                return;
            }
        }

        // Assign request to server
        hash_table[idx] = val;
        cout << "Request " << ip << " (sum = " << val << ") is
handled by Server " << idx << endl;
    }

    // Display load summary of all servers
    void display() {
        cout << "\nServer Load Summary:\n";
        for (int i = 0; i < n_servers; i++) {
            cout << "Server " << i << " <- ";
            if (hash_table[i] != 0)
                cout << hash_table[i];
            else
                cout << "<empty>";
            cout << endl;
        }
    }
};

int main() {
    int n, r;
    cout << "Enter number of servers: ";
    cin >> n;

    cout << "Enter number of IP requests: ";
    cin >> r;

    LoadBalancer lb(n); // Create load balancer object

    // Take multiple requests
    for (int i = 0; i < r; i++) {
        char ip[50];
        cout << "Enter Request " << i + 1 << " IP: ";
        cin >> ip;
        lb.insert(ip);
    }

    // Show final distribution
    lb.display();
    return 0;
}
```

Sample Output:

```
Enter number of servers: 5
Enter number of IP requests: 4
Enter Request 1 IP: 192.168.1.5
Request 192.168.1.5 (sum = 366) is handled by Server 1
Enter Request 2 IP: 192.168.1.10
```

```
Request 192.168.1.10 (sum = 371) is handled by Server 1
Enter Request 3 IP: 10.0.0.1
Request 10.0.0.1 (sum = 11) is handled by Server 1
Enter Request 4 IP: 172.16.0.2
Request 172.16.0.2 (sum = 190) is handled by Server 0

Server Load Summary:
Server 0 <- 190
Server 1 <- 11
Server 2 <- <empty>
Server 3 <- <empty>
Server 4 <- <empty>
```