Assignment No: 03

Name: Tanvi Sanjay Dongare
Class: SY-1
Batch: C
PRN: B25CE2010

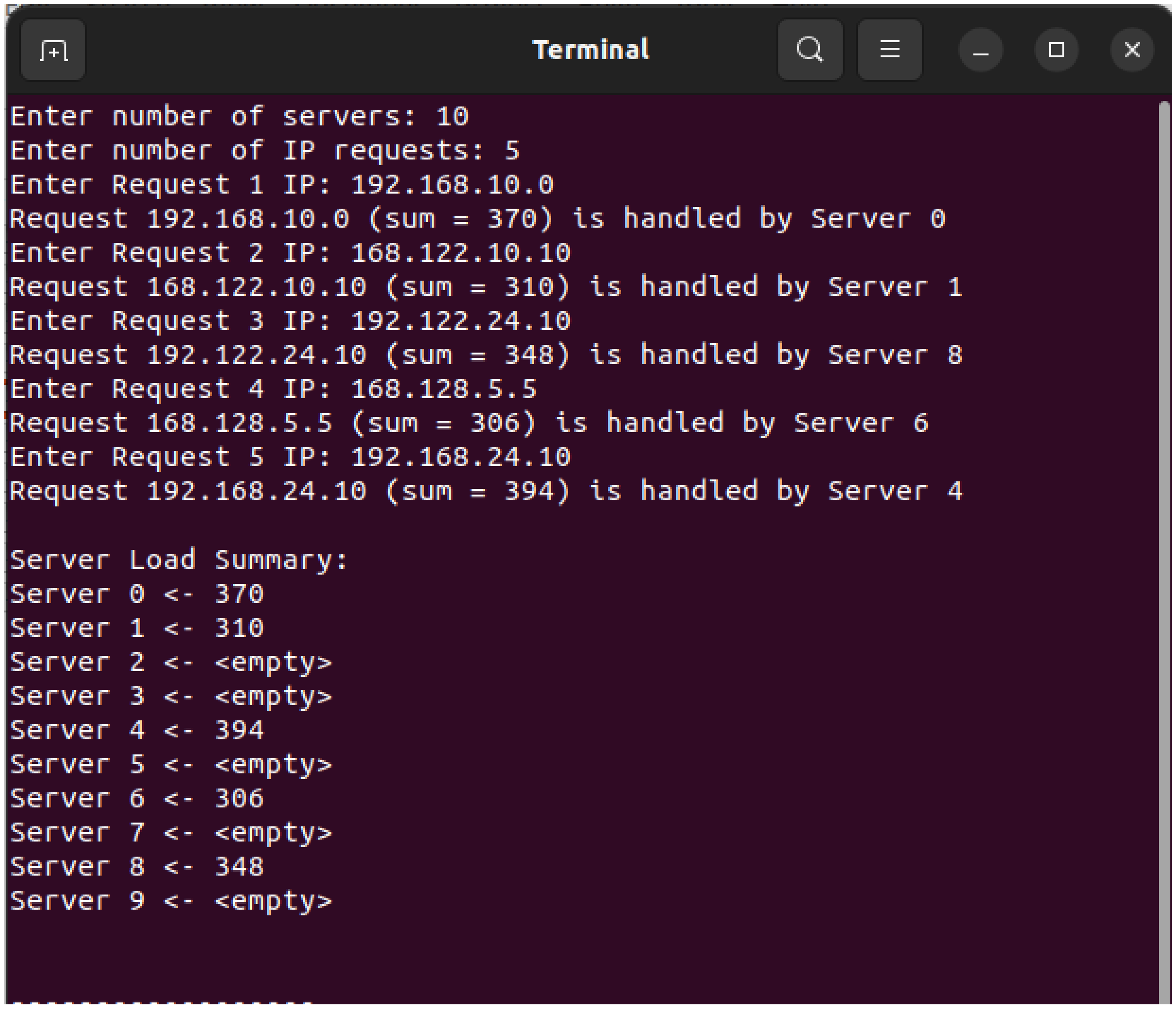
Title:
Load Balancing:
For example, imagine you have a set of servers that handle requests for a web application. The key to load balancing is using the hash value of a client's IP address or a request ID to determine which server should handle the request. The hash function is typically designed so 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 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;
    // Traverse character by character
    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 with 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;
}
```

Output:



```
Enter number of servers: 10
Enter number of IP requests: 5
Enter Request 1 IP: 192.168.10.0
Request 192.168.10.0 (sum = 370) is handled by Server 0
Enter Request 2 IP: 168.122.10.10
Request 168.122.10.10 (sum = 310) is handled by Server 1
Enter Request 3 IP: 192.122.24.10
Request 192.122.24.10 (sum = 348) is handled by Server 8
Enter Request 4 IP: 168.128.5.5
Request 168.128.5.5 (sum = 306) is handled by Server 6
Enter Request 5 IP: 192.168.24.10
Request 192.168.24.10 (sum = 394) is handled by Server 4

Server Load Summary:
Server 0 <- 370
Server 1 <- 310
Server 2 <- <empty>
Server 3 <- <empty>
Server 4 <- 394
Server 5 <- <empty>
Server 6 <- 306
Server 7 <- <empty>
Server 8 <- 348
Server 9 <- <empty>
```