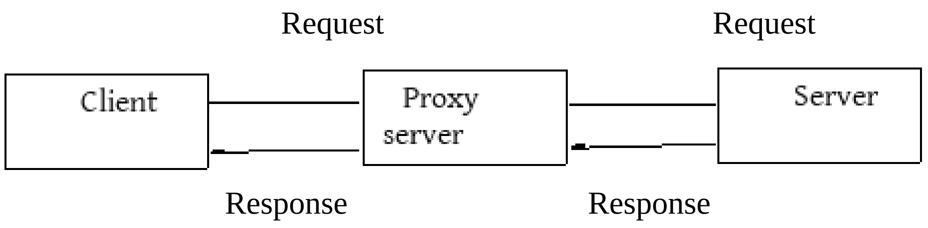
1. Write a socket program to create a proxy server. Create client, server and and proxy server programs separately.



2. Create two files at server side with the following fields:

Department.txt

Name	Employee_Count
Metallurgy	3
Chemistry	3
Robotics	1
HR	2
Communication	4

Employee.txt

Name	Salary	Department_Name
Rohit	10000	Metallurgy
Sandeep	8000	Chemistry
Sumit	12000	Robotics
Ankur	11000	Chemistry
Gagan	19000	Communication
Sangam	15000	Chemistry
Soumitra	6000	Metallurgy
Subhashis	9000	HR

A user at the client side should be given the following options (at terminal) which in turn issues the corresponding query to the server:
a. Display the record of employee with highest salary.
b. Display the record of employees(name, salary and department_name) with particular Department_Name (taken as input from terminal).
c. Display the record of employees (name, salary and department_name) who belong to the department with highest Employee_Count.
d. Display record of the employee (name, salary and department_name) with second highest salary.
The server should process the query and give proper result to the client which then displays the result on it’s terminal.
Note : During evaluation, the content of the files may be changed.

3. Create a server client program where the following tasks are carried out by the server:

The server will start in *passive* mode listening on a specified port for a transmission from a client. Separately, the client will be started and will contact the server on a given IP address and port number that must be entered via the command line. The client will pass the server a string consisting of a sequence of characters. If the the string contains anything but numbers, the server will respond with "Sorry, cannot compute!" and exit. If the string contains all numbers, the individual digits will be added together and returned as a string (see below for an example). If the server sends a "Sorry" response to the client it will immediately exit. If the server receives a string of numbers, it will (1) add the digits together, (2) send the value back to the client, and (3) will not exit unless the response is a single digit. This process will be repeated until there is only one digit remaining. Note: the server will send a new packet each time Step (2) is executed, and the client will expect to receive a packet until there is only a single digit. See below for the exact output.

Example:

Client Input/Output for Non-Numeric Example

```
•	machine2> client 128.111.49.44 32000
Enter string: I don't like addition!!
From server: Sorry, cannot compute!
Machine2>
```

Client Input/Output for Numeric Example

```
machine2> client 128.111.49.44 32000
Enter string: 123456789101234567891012345678910
From server: 138
From server: 12
From server: 3
machine2>
```