Exp :27
Implementation of a DNS server and client in C using UDP sockets.

Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 5353
#define BUFFER_SIZE 1024
struct dns_entry {
   char domain[50];
   char ip[20];
};

int main() {
   int sockfd;
   struct sockaddr_in server_addr, client_addr;
   char buffer[BUFFER_SIZE];
   socklen_t addr_len;
   int n, i;
   // Predefined DNS records
   struct dns_entry dns_table[] = {
      {"example.com", "93.184.216.34"},
      {"google.com",  "142.250.183.110"},
      {"openai.com",  "104.18.21.213"}
   };
   int table_size = sizeof(dns_table) / sizeof(dns_table[0]);
   // Create UDP socket
   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
      perror("Socket creation failed");
      exit(EXIT_FAILURE);
   }
```

```
// Configure server address
memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
server_addr.sin_port = htons(PORT);
// Bind socket to port
if (bind(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
   perror("Bind failed");
   close(sockfd);
   exit(EXIT_FAILURE);
}
printf("DNS Server running on port %d...\n", PORT);
while (1) {
   addr_len = sizeof(client_addr);
   // Receive domain name from client
   n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0,
            (struct sockaddr *)&client_addr, &addr_len);
   buffer[n] = '\0';
   printf("Query received for: %s\n", buffer);
   // Search in DNS table
   char response[BUFFER_SIZE] = "Domain not found";
   for (i = 0; i < table_size; i++) {
      if (strcmp(buffer, dns_table[i].domain) == 0) {
         strcpy(response, dns_table[i].ip);
         break;
      }
   }

   // Send IP address back to client
   sendto(sockfd, response, strlen(response), 0,
        (struct sockaddr *)&client_addr, addr_len);
}
close(sockfd);
```

```
    return 0;
}
```

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define SERVER_IP "127.0.0.1"
#define PORT 5353
#define BUFFER_SIZE 1024
int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    socklen_t addr_len;
    int n;
    // Create UDP socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
    // Configure server address
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);
    printf("Enter domain name: ");
    fgets(buffer, BUFFER_SIZE, stdin);
    buffer[strcspn(buffer, "\n")] = '\0'; // Remove newline

    // Send domain to server
    sendto(sockfd, buffer, strlen(buffer), 0,
```

```
        (struct sockaddr *)&server_addr, sizeof(server_addr));
    // Receive IP from server
    addr_len = sizeof(server_addr);
    n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0,
            (struct sockaddr *)&server_addr, &addr_len);
    buffer[n] = '\0';
    printf("IP Address: %s\n", buffer);
    close(sockfd);
    return 0;
}
```

Output

client

```
Client: Enter domain name: google.com
Client: IP Address: 142.250.183.110
```

server

```
DNS Server running on port 5353...
Query received for: google.com
```

Ex:No : 28

Creating the applications using TCP echo server and client in java/C.

Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024
int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    char buffer[BUFFER_SIZE];
```