```
        (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];
```

```
int bytes_read;
socklen_t addr_len = sizeof(address);

// Create socket
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
    perror("Socket failed");
    exit(EXIT_FAILURE);
}
// Bind to the port
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;  // Listen on all interfaces
address.sin_port = htons(PORT);
if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
    perror("Bind failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}
// Listen for connections
if (listen(server_fd, 3) < 0) {
    perror("Listen failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}
printf("TCP Echo Server running on port %d...\n", PORT);
// Accept client connection
if ((new_socket = accept(server_fd, (struct sockaddr *)&address, &addr_len)) < 0) {
    perror("Accept failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}

printf("Client connected.\n");
// Handle client communication
```

```
    while ((bytes_read = read(new_socket, buffer, BUFFER_SIZE)) > 0) {
        buffer[bytes_read] = '\0';
        printf("Received: %s", buffer);
        // Send the same data back to the client
        send(new_socket, buffer, bytes_read, 0);
    }
    printf("Client disconnected.\n");
    close(new_socket);
    close(server_fd);
    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 8080
#define BUFFER_SIZE 1024
int main() {
    int sock;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];

    // Create socket
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    // Set up server address
```

```
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    if (inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr) <= 0) {
        perror("Invalid address / Address not supported");
        exit(EXIT_FAILURE);
    }
    // Connect to server
    if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }
    printf("Connected to TCP Echo Server.\n");
    // Send and receive messages
    while (1) {
        printf("Enter message (type 'exit' to quit): ");
        fgets(buffer, BUFFER_SIZE, stdin);
        if (strncmp(buffer, "exit", 4) == 0)
            break;
        send(sock, buffer, strlen(buffer), 0);
       int bytes_received = read(sock, buffer, BUFFER_SIZE);
        buffer[bytes_received] = '\0';
        printf("Echo from server: %s", buffer);
    }

    close(sock);
    return 0;
}
```

Output:

Client:

Connected to TCP Echo Server.

Enter message (type 'exit' to quit'): Hello

Echo from server: Hello

Enter message (type 'exit' to quit'): How are you?

Echo from server: How are you?

Server

TCP Echo Server running on port 8080...

Client connected.

Received: Hello

Received: How are you?

Client disconnected.

Ex:No : 29

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

TCP Chat 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];
   socklen_t addr_len = sizeof(address);
   // Create socket
   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
      perror("Socket failed");
      exit(EXIT_FAILURE);
   }
   // Bind
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons(PORT);
```