# Assignment No. 10

**a) Write a client and server programs for interaction between server and client processes using TCP sockets. On successful connection, client sends an IPv4 address to the server. Sever identifies the network id of the IP based on its class and replies the network id back to the client.**

```
//Server
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
    int server_socket, new_socket;
    struct sockaddr_in server_addr, client_addr;
    int addrlen = sizeof(server_addr);

    // Create socket
    if ((server_socket = socket(AF_INET, SOCK_STREAM,
0)) == -1) {
        perror("socket");
        return 1;
    }

    // Prepare the sockaddr_in structure
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind
    if (bind(server_socket, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("bind");
        return 1;
    }
```

```
    // Listen
    if (listen(server_socket, 5) == -1) {
        perror("listen");
        return 1;
    }

    printf("Server listening on port %d...\n", PORT);

    // Accept incoming connection
    if ((new_socket = accept(server_socket, (struct
sockaddr *)&client_addr, (socklen_t*)&addrlen)) == -
1) {
        perror("accept");
        return 1;
    }

    char client_ip[INET_ADDRSTRLEN];
    inet_ntop(AF_INET, &client_addr.sin_addr,
client_ip, INET_ADDRSTRLEN);
    printf("Client connected: %s\n", client_ip);

    // Receive IP address from client
    char ip_address[INET_ADDRSTRLEN];
    recv(new_socket, ip_address, sizeof(ip_address),
0);
    printf("Received IP address from client: %s\n",
ip_address);

    // Identify network id based on IP class
    char network_id[20];
    char ip_class = ip_address[0];
    if (ip_class >= 'A' && ip_class <= 'C') {
        strcpy(network_id, "Private Network");
    } else {
        strcpy(network_id, "Public Network");
    }

    // Reply network id back to client
    send(new_socket, network_id, strlen(network_id),
0);
    printf("Replied network ID to client: %s\n",
network_id);

    close(new_socket);
```

```
    close(server_socket);

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 10thAssigna 10thAssigna.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssigna
Server listening on port 8080...
Client connected: 127.0.0.1
Received IP address from client: 192.168.1.1
Replied network ID to client: Public Network
```

```
//Client
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
    int client_socket;
    struct sockaddr_in server_addr;

    // Create socket
    if ((client_socket = socket(AF_INET, SOCK_STREAM,
0)) == -1) {
        perror("socket");
        return 1;
    }

    // Prepare the sockaddr_in structure
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to
binary form
    if (inet_pton(AF_INET, "127.0.0.1",
&server_addr.sin_addr) <= 0) {
        perror("inet_pton");
        return 1;
    }

    // Connect to server
```

```
    if (connect(client_socket, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("connect");
        return 1;
    }

    printf("Connected to server...\n");

    // Send IP address to server
    char ip_address[] = "192.168.1.1"; // Replace
with your IP address
    send(client_socket, ip_address,
strlen(ip_address), 0);
    printf("Sent IP address to server: %s\n",
ip_address);

    // Receive network id from server
    char network_id[20];
    recv(client_socket, network_id,
sizeof(network_id), 0);
    printf("Received network ID from server: %s\n",
network_id);

    close(client_socket);

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssignar
Connected to server...
Sent IP address to server: 192.168.1.1
Received network ID from server: Public Network
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

**b) Write a client and server programs using TCP sockets, where a server maintains a list of country-capital pair in its system. On successful connection, client sends the name of a country and in response serve replies the name of the capital of the country if it is present in the list; otherwise, sends 'NOT FOUND' back to the client.**

```
//Server
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

typedef struct {
    char country[50];
    char capital[50];
} CountryCapitalPair;

CountryCapitalPair countryCapitalPairs[] = {
    {"India", "New Delhi"},
    {"USA", "Washington, D.C."},
    {"Japan", "Tokyo"},
    {"Germany", "Berlin"},
    // Add more country-capital pairs as needed
};

int main() {
    int server_socket, new_socket;
    struct sockaddr_in server_addr, client_addr;
    int addrlen = sizeof(server_addr);

    // Create socket
    if ((server_socket = socket(AF_INET, SOCK_STREAM,
0)) == -1) {
        perror("socket");
        return 1;
    }

    // Prepare the sockaddr_in structure
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
```

```
    server_addr.sin_port = htons(PORT);

    // Bind
    if (bind(server_socket, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("bind");
        return 1;
    }

    // Listen
    if (listen(server_socket, 5) == -1) {
        perror("listen");
        return 1;
    }

    printf("Server listening on port %d...\n", PORT);

    // Accept incoming connection
    if ((new_socket = accept(server_socket, (struct
sockaddr *)&client_addr, (socklen_t*)&addrlen)) == -
1) {
        perror("accept");
        return 1;
    }

    printf("Client connected.\n");

    // Receive country name from client
    char country_name[50];
    recv(new_socket, country_name,
sizeof(country_name), 0);
    printf("Received country name from client: %s\n",
country_name);

    // Search for country-capital pair
    char capital_name[50] = "NOT FOUND";
    for (int i = 0; i < sizeof(countryCapitalPairs) /
sizeof(countryCapitalPairs[0]); i++) {
        if (strcmp(countryCapitalPairs[i].country,
country_name) == 0) {
            strcpy(capital_name,
countryCapitalPairs[i].capital);
            break;
        }
```

```
    }

    // Reply capital name back to client
    send(new_socket, capital_name,
strlen(capital_name), 0);
    printf("Replied capital name to client: %s\n",
capital_name);

    close(new_socket);
    close(server_socket);

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 10thAssignb.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 10thAssignb 10thAssignb.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssignb
Server listening on port 8080...
Client connected.
Received country name from client: India
Replied capital name to client: New Delhi
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

```
//Client
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
    int client_socket;
    struct sockaddr_in server_addr;

    // Create socket
    if ((client_socket = socket(AF_INET, SOCK_STREAM,
0)) == -1) {
        perror("socket");
        return 1;
    }

    // Prepare the sockaddr_in structure
    server_addr.sin_family = AF_INET;
```

```
    server_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to
binary form
    if (inet_pton(AF_INET, "127.0.0.1",
&server_addr.sin_addr) <= 0) {
        perror("inet_pton");
        return 1;
    }

    // Connect to server
    if (connect(client_socket, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("connect");
        return 1;
    }

    printf("Connected to server...\n");

    char country_name[] = "India"; ]
    send(client_socket, country_name,
strlen(country_name), 0);
    printf("Sent country name to server: %s\n",
country_name);
    char capital_name[50];
    recv(client_socket, capital_name,
sizeof(capital_name), 0);
    printf("Received capital name from server: %s\n",
capital_name);

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

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 10thAssignbr.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 10thAssignbr 10thAssignbr.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssignbr
Connected to server...
Sent country name to server: India
Received capital name from server: New Delhi
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

**c) Write a client and server programs using TCP sockets, where a concurrent TCP server maintains user credentials at its end. On successful connection, client enters username, and then password. Server verifies the username and password and replies either 'Success', 'Invalid user', or 'Invalid password' depending on the result of verification.**

```
//Server

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

typedef struct {
    char username[50];
    char password[50];
} UserCredentials;

UserCredentials validUsers[] = {
    {"alice", "12345"},
    {"bob", "qwerty"},
    // Add more valid user credentials as needed
};

int verifyCredentials(const char *username, const
char *password) {
    for (int i = 0; i < sizeof(validUsers) /
sizeof(validUsers[0]); i++) {
        if (strcmp(validUsers[i].username, username)
== 0) {
            if (strcmp(validUsers[i].password,
password) == 0) {
                return 1; // Success
            } else {
                return 0; // Invalid password
            }
        }
    }
    return -1; // Invalid user
```

```
}

int main() {
    int server_socket, new_socket;
    struct sockaddr_in server_addr, client_addr;
    int addrlen = sizeof(server_addr);

    // Create socket
    if ((server_socket = socket(AF_INET, SOCK_STREAM,
0)) == -1) {
        perror("socket");
        return 1;
    }

    // Prepare the sockaddr_in structure
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind
    if (bind(server_socket, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("bind");
        return 1;
    }

    // Listen
    if (listen(server_socket, 5) == -1) {
        perror("listen");
        return 1;
    }

    printf("Server listening on port %d...\n", PORT);

    while (1) {
        // Accept incoming connection
        if ((new_socket = accept(server_socket,
(struct sockaddr *)&client_addr,
(socklen_t*)&addrlen)) == -1) {
            perror("accept");
            return 1;
        }

        printf("Client connected.\n");
```

```
        // Receive username from client
        char username[50];
        recv(new_socket, username, sizeof(username),
0);
        printf("Received username from client: %s\n",
username);

        // Receive password from client
        char password[50];
        recv(new_socket, password, sizeof(password),
0);
        printf("Received password from client: %s\n",
password);

        // Verify user credentials
        int verification_result =
verifyCredentials(username, password);
        char response[20];
        if (verification_result == 1) {
            strcpy(response, "Success");
        } else if (verification_result == 0) {
            strcpy(response, "Invalid password");
        } else {
            strcpy(response, "Invalid user");
        }

        // Reply verification result back to client
        send(new_socket, response, strlen(response),
0);
        printf("Replied verification result to
client: %s\n", response);

        close(new_socket);
    }

    close(server_socket);

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 10thAssignc.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 10thAssignc 10thAssignc.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssignc
Server listening on port 8080...
Client connected.
Received username from client: Anubhavk
Received password from client: A@1234
Replied verification result to client: Invalid user
^C
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

```
//Client

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
    int client_socket;
    struct sockaddr_in server_addr;

    // Create socket
    if ((client_socket = socket(AF_INET, SOCK_STREAM,
0)) == -1) {
        perror("socket");
        return 1;
    }

    // Prepare the sockaddr_in structure
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to
binary form
    if (inet_pton(AF_INET, "127.0.0.1",
&server_addr.sin_addr) <= 0) {
        perror("inet_pton");
        return 1;
    }

    // Connect to server
```

```
    if (connect(client_socket, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("connect");
        return 1;
    }

    printf("Connected to server...\n");

    // Send username to server
    char username[] = "alice"; // Replace with the
username
    send(client_socket, username, strlen(username),
0);
    printf("Sent username to server: %s\n",
username);

    // Send password to server
    char password[] = "12345"; // Replace with the
password
    send(client_socket, password, strlen(password),
0);
    printf("Sent password to server: %s\n",
password);

    // Receive verification result from server
    char verification_result[20];
    recv(client_socket, verification_result,
sizeof(verification_result), 0);
    printf("Received verification result from server:
%s\n", verification_result);

    close(client_socket);

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 10thAssigncr.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 10thAssigncr 10thAssigncr.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssigncr
Connected to server...
Sent username to server: Anubhavk
Sent password to server: A@1234
Received verification result from server: Invalid user
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

**d) Write a client and server programs(using c)for interaction between server and client processes using UDP sockets. Perform the country-capital pair assignment. for UDP Socket.**

```
//server

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

typedef struct {
    char country[50];
    char capital[50];
} CountryCapitalPair;

CountryCapitalPair countryCapitalPairs[] = {
    {"India", "New Delhi"},
    {"USA", "Washington, D.C."},
    {"Japan", "Tokyo"},
    {"Germany", "Berlin"},
};

int main() {
    int socket_fd;
    struct sockaddr_in server_addr, client_addr;
    int addrlen = sizeof(server_addr);
    if ((socket_fd = socket(AF_INET, SOCK_DGRAM, 0))
== -1) {
        perror("socket");
        return 1;
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind
```

```
    if (bind(socket_fd, (struct sockaddr
*)&server_addr, sizeof(server_addr)) == -1) {
        perror("bind");
        return 1;
    }

    printf("Server listening on port %d...\n", PORT);

    while (1) {
        // Receive country name from client
        char country_name[50];
        recvfrom(socket_fd, country_name,
sizeof(country_name), 0, (struct sockaddr
*)&client_addr, (socklen_t*)&addrlen);
        printf("Received country name from client:
%s\n", country_name);

        // Search for country-capital pair
        char capital_name[50] = "NOT FOUND";
        for (int i = 0; i <
sizeof(countryCapitalPairs) /
sizeof(countryCapitalPairs[0]); i++) {
            if
(strcmp(countryCapitalPairs[i].country, country_name)
== 0) {
                strcpy(capital_name,
countryCapitalPairs[i].capital);
                break;
            }
        }

        // Send capital name back to client
        sendto(socket_fd, capital_name,
strlen(capital_name), 0, (struct sockaddr
*)&client_addr, sizeof(client_addr));
        printf("Sent capital name to client: %s\n",
capital_name);
    }

    close(socket_fd);

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 10thAssignd.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 10thAssignd 10thAssignd.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssignd
Server listening on port 8080...
Received country name from client: India
Sent capital name to client: New Delhi
Received country name from client: India
Sent capital name to client: New Delhi
^Z
[1]+  Stopped                 ./10thAssignd
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

```
//Client

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
    int socket_fd;
    struct sockaddr_in server_addr;

    // Create socket
    if ((socket_fd = socket(AF_INET, SOCK_DGRAM, 0))
== -1) {
        perror("socket");
        return 1;
    }

    // Prepare the sockaddr_in structure
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to
binary form
    if (inet_pton(AF_INET, "127.0.0.1",
&server_addr.sin_addr) <= 0) {
        perror("inet_pton");
        return 1;
    }
```

```
    printf("Client running...\n");

    // Send country name to server
    char country_name[] = "India"; // Replace with
the country name
    sendto(socket_fd, country_name,
strlen(country_name), 0, (struct sockaddr
*)&server_addr, sizeof(server_addr));
    printf("Sent country name to server: %s\n",
country_name);

    // Receive capital name from server
    char capital_name[50];
    int addrlen = sizeof(server_addr);
    recvfrom(socket_fd, capital_name,
sizeof(capital_name), 0, (struct sockaddr
*)&server_addr, (socklen_t*)&addrlen);
    printf("Received capital name from server: %s\n",
capital_name);

    close(socket_fd);

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 10thAssigndr.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 10thAssigndr 10thAssigndr.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssigndr
Client running...
Sent country name to server: India
Received capital name from server: New Delhi
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./10thAssigndr
Client running...
Sent country name to server: India
Received capital name from server: New Delhi
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```