Computer Networks
Lab Assignment

Spriha Anvi
21BPS1191

Server Side

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

#define SERVER_PORT 8080

void get_network_class(const struct in_addr *ip, char *network_class,
char *subnet_mask) {
    uint32_t address = ntohl(ip->s_addr);

    if ((address & 0x80000000) == 0) {
        *network_class = 'A';
        strcpy(subnet_mask, "255.0.0.0");
    } else if ((address & 0xC0000000) == 0x80000000) {
        *network_class = 'B';
        strcpy(subnet_mask, "255.255.0.0");
    } else if ((address & 0xE0000000) == 0xC0000000) {
        *network_class = 'C';
        strcpy(subnet_mask, "255.255.255.0");
    } else if ((address & 0xF0000000) == 0xE0000000) {
        *network_class = 'D';
        strcpy(subnet_mask, "Reserved for multicast");
    } else {
        *network_class = 'E';
        strcpy(subnet_mask, "Reserved for experimental");
```

```
    }
}

int main() {
    int serverSocket, newSocket;
    struct sockaddr_in serverAddr, clientAddr;
    socklen_t addrSize = sizeof(clientAddr);
    char clientIP[INET_ADDRSTRLEN];

    // Create socket
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket < 0) {
        perror("Error in socket creation");
        exit(EXIT_FAILURE);
    }

    // Set up server address
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(SERVER_PORT);

    // Bind the socket to the specified IP and port
    if (bind(serverSocket, (struct sockaddr *)&serverAddr,
sizeof(serverAddr)) < 0) {
        perror("Binding failed");
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(serverSocket, 5) < 0) {
        perror("Error in listening");
        exit(EXIT_FAILURE);
    }

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

    while (1) {
        // Accept a new connection
```

```
        newSocket = accept(serverSocket, (struct sockaddr *)&clientAddr,
&addrSize);
        if (newSocket < 0) {
            perror("Error in accepting connection");
            exit(EXIT_FAILURE);
        }

        // Get client IP address
        inet_ntop(AF_INET, &(clientAddr.sin_addr), clientIP,
sizeof(clientIP));
        printf("Client connected from %s\n", clientIP);

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

        // Process IP address and get class and subnet mask
        struct in_addr ip;
        if (inet_pton(AF_INET, clientIPAddr, &ip) <= 0) {
            perror("Invalid IP address");
            close(newSocket);
            continue;
        }

        char networkClass;
        char subnetMask[INET_ADDRSTRLEN];
        get_network_class(&ip, &networkClass, subnetMask);

        // Send the result back to the client
        char serverResponse[256];
        snprintf(serverResponse, sizeof(serverResponse), "Network Class:
%c, Subnet Mask: %s", networkClass, subnetMask);
        send(newSocket, serverResponse, strlen(serverResponse), 0);
        printf("Sent response to client\n");

        // Close the new socket
        close(newSocket);
```

```
    }

    // Close the server socket
    close(serverSocket);

    return 0;
}
```

Client Side

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

#define SERVER_IP "127.0.0.1"
#define SERVER_PORT 8080

int main() {
    int clientSocket;
    struct sockaddr_in serverAddr;
    char ipAddress[16]; // Assuming IPv4 address

    // Create socket
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket < 0) {
        perror("Error in socket creation");
        exit(EXIT_FAILURE);
    }

    // Set up server address
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(SERVER_PORT);
    if (inet_pton(AF_INET, SERVER_IP, &(serverAddr.sin_addr)) <= 0) {
        perror("Invalid address/Address not supported");
        exit(EXIT_FAILURE);
    }
```

```
    // Connect to server
    if (connect(clientSocket, (struct sockaddr*)&serverAddr,
sizeof(serverAddr)) < 0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }

    // Get client IP address
    printf("Enter IP address: ");
    fgets(ipAddress, sizeof(ipAddress), stdin);
    ipAddress[strcspn(ipAddress, "\n")] = '\0';

    // Send IP address to server
    send(clientSocket, ipAddress, strlen(ipAddress), 0);
    printf("IP address sent to server.\n");

    // Receive class and subnet mask from server
    char serverResponse[256];
    memset(serverResponse, 0, sizeof(serverResponse));
    recv(clientSocket, serverResponse, sizeof(serverResponse), 0);
    printf("Server response: %s\n", serverResponse);

    // Close the socket
    //pclose(clientSocket);

    return 0;
}
```

Output

```
student@hostserver42:~/Desktop$ gcc server.c
student@hostserver42:~/Desktop$ ./a.out
Server listening on port 8080
Client connected from 127.0.0.1
Received IP address from client: 192.255.0.1
Sent response to client
```

```
student@hostserver42:~$ cd Desktop/
student@hostserver42:~/Desktop$ gcc client.c
student@hostserver42:~/Desktop$ ./a.out
Enter IP address: 192.255.0.1
IP address sent to server.
Server response: Network Class: C, Subnet Mask: 255.255.255.0
student@hostserver42:~/Desktop$
```

```
student@hostserver42:~/Desktop$ ./a.out
Server listening on port 8080
Client connected from 127.0.0.1
Received IP address from client: 25.21.0.1
Sent response to client
```

```
student@hostserver42:~/Desktop$ gcc client.c
student@hostserver42:~/Desktop$ ./a.out
Enter IP address: 25.21.0.1
IP address sent to server.
Server response: Network Class: A, Subnet Mask: 255.0.0.0
student@hostserver42:~/Desktop$
```