COMPUTER NETWORKS

LAB ASSIGNMENT-08

REDDIPALLI SAI CHARISH

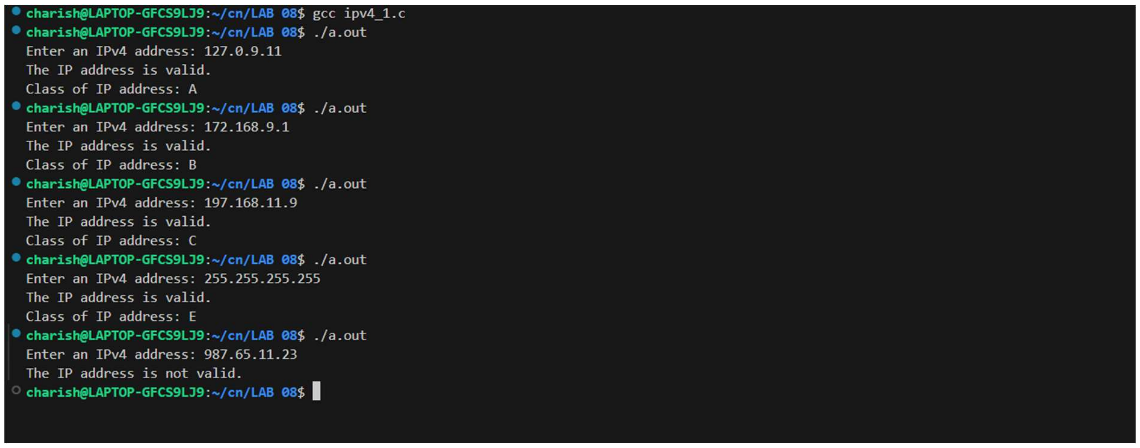
CS22B1095

IPv4:

Classification:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

char find_ip_class(int first_octet) {
   if (first_octet >= 0 && first_octet <= 127)
      return 'A';
   else if (first_octet >= 128 && first_octet <= 191)
      return 'B';
   else if (first_octet >= 192 && first_octet <= 223)
      return 'C';
   else if (first_octet >= 224 && first_octet <= 239)
      return 'D';
   else if (first_octet >= 240 && first_octet <= 255)
      return 'E';
   else
      return 'X';
}
int main() {
   char ip[16];
   printf("Enter an IPv4 address: ");
   scanf("%15s", ip);

   int num, dots = 0;
   char *ptr = strtok(ip, ".");
```

```
    while (ptr != NULL) {
        // for (int i = 0; i < (int)strlen(ptr); i++) {
        //     if (!isdigit(ptr[i])) {
        //         printf("The IP address is not valid.\n");
        //         return 0;
        //     }
        // }
        for (size_t i = 0; i < strlen(ptr); i++) {
            if (!isdigit(ptr[i])) {
                printf("The IP address is not valid.\n");
                return 0;
            }
        }

        num = atoi(ptr);
        if (num < 0 || num > 255) {
            printf("The IP address is not valid.\n");
            return 0;
        }

        ptr = strtok(NULL, ".");
        dots++;
    }

    if (dots != 4) {
        printf("The IP address is not valid.\n");
        return 0;
    }

    int first_octet = atoi(ip);
    char ip_class = find_ip_class(first_octet);

    printf("The IP address is valid.\n");
    printf("Class of IP address: %c\n", ip_class);

    return 0;
}
```



```
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ gcc ipv4_1.c
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./a.out
Enter an IPv4 address: 127.0.9.11
The IP address is valid.
Class of IP address: A
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./a.out
Enter an IPv4 address: 172.168.9.1
The IP address is valid.
Class of IP address: B
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./a.out
Enter an IPv4 address: 197.168.11.9
The IP address is valid.
Class of IP address: C
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./a.out
Enter an IPv4 address: 255.255.255.255
The IP address is valid.
Class of IP address: E
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./a.out
Enter an IPv4 address: 987.65.11.23
The IP address is not valid.
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$
```

2nd question :

Server_code:

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

char find_ip_class(int first_octet) {
    if (first_octet >= 1 && first_octet <= 127)
        return 'A';
    else if (first_octet >= 128 && first_octet <= 191)
        return 'B';
    else if (first_octet >= 192 && first_octet <= 223)
        return 'C';
    else if (first_octet >= 224 && first_octet <= 239)
        return 'D';
    else if (first_octet >= 240 && first_octet <= 255)
        return 'E';
    else
        return 'X';  // Invalid class
}

int main() {
```

```c
int server_socket, new_socket;

struct sockaddr_in server_addr, client_addr;

socklen_t addr_len = sizeof(client_addr);

char client_ip[16];


server_socket = socket(AF_INET, SOCK_STREAM, 0);

if (server_socket == -1) {

    perror("Could not create socket");

    exit(EXIT_FAILURE);

}


server_addr.sin_family = AF_INET;

server_addr.sin_addr.s_addr = INADDR_ANY;

server_addr.sin_port = htons(8080);


if (bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {

    perror("Bind failed");

    close(server_socket);

    exit(EXIT_FAILURE);

}


listen(server_socket, 3);


printf("Server listening on port 8080...\n");


new_socket = accept(server_socket, (struct sockaddr *)&client_addr, &addr_len);

if (new_socket < 0) {

    perror("Accept failed");

    close(server_socket);

    exit(EXIT_FAILURE);

}


recv(new_socket, client_ip, sizeof(client_ip), 0);


int first_octet = atoi(strtok(client_ip, "."));

char ip_class = find_ip_class(first_octet);


char response[32];
```

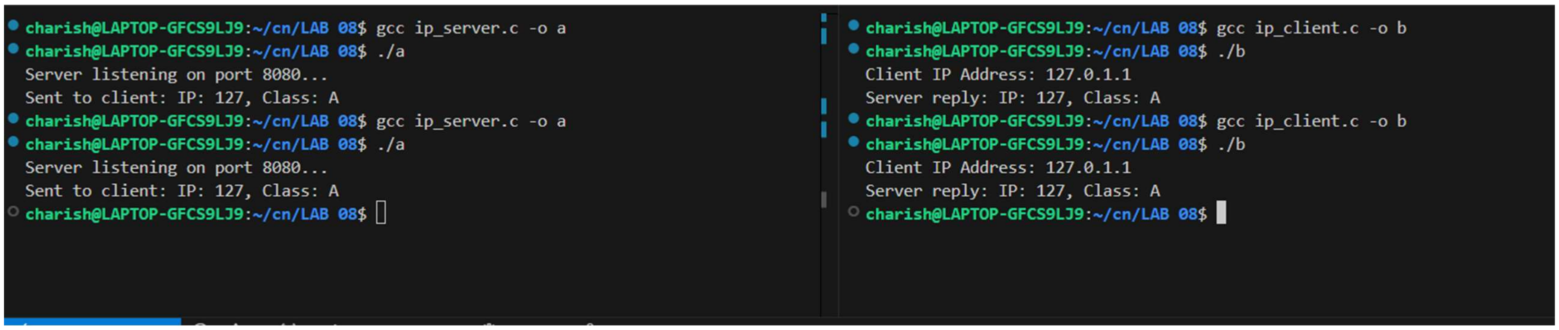
```
    snprintf(response, sizeof(response), "IP: %s, Class: %c", client_ip, ip_class);
    send(new_socket, response, strlen(response), 0);

    printf("Sent to client: %s\n", response);

    close(new_socket);
    close(server_socket);

    return 0;
}
```

Client Code:

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

void get_ip_address(char *ip) {
    char hostname[50];
    struct hostent *host_entry;

    if (gethostname(hostname, sizeof(hostname)) == -1) {
        perror("gethostname");
        exit(EXIT_FAILURE);
    }

    host_entry = gethostbyname(hostname);
    if (host_entry == NULL) {
        perror("gethostbyname");
        exit(EXIT_FAILURE);
    }
```

```
   // Copy the IP address to the provided buffer
   strcpy(ip, inet_ntoa(*((struct in_addr*)host_entry->h_addr_list[0])));
 }
// void get_ip_address(char *ip) {
//     struct ifaddrs *ifaddr, *ifa;
//     void *tmp_addr_ptr;

//     if (getifaddrs(&ifaddr) == -1) {
//         perror("getifaddrs");
//         exit(EXIT_FAILURE);
//     }

//     for (ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
//         if (ifa->ifa_addr == NULL) continue;

//         if (ifa->ifa_addr->sa_family == AF_INET) {  // Only IPv4
//             tmp_addr_ptr = &((struct sockaddr_in *)ifa->ifa_addr)->sin_addr;
//             inet_ntop(AF_INET, tmp_addr_ptr, ip, INET_ADDRSTRLEN);
//             break;  // Take the first non-loopback address
//         }
//     }
//     freeifaddrs(ifaddr);
// }

int main() {
   int client_socket;
   struct sockaddr_in server_addr;
   char server_reply[32];
   char client_ip[INET_ADDRSTRLEN];

   get_ip_address(client_ip);
   printf("Client IP Address: %s\n", client_ip);

   client_socket = socket(AF_INET, SOCK_STREAM, 0);
   if (client_socket == -1) {
      perror("Could not create socket");
      exit(EXIT_FAILURE);
   }
```

```
    server_addr.sin_family = AF_INET;

    server_addr.sin_port = htons(8080);

    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");


    if (connect(client_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {

        perror("Connection failed");

        close(client_socket);

        exit(EXIT_FAILURE);

    }


    send(client_socket, client_ip, strlen(client_ip), 0);


    recv(client_socket, server_reply, sizeof(server_reply), 0);

    printf("Server reply: %s\n", server_reply);


    // Close socket

    close(client_socket);


    return 0;

}
```



```
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ gcc ip_server.c -o a
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./a
Server listening on port 8080...
Sent to client: IP: 127, Class: A
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ gcc ip_server.c -o a
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./a
Server listening on port 8080...
Sent to client: IP: 127, Class: A
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$
```

```
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ gcc ip_client.c -o b
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./b
Client IP Address: 127.0.1.1
Server reply: IP: 127, Class: A
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ gcc ip_client.c -o b
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$ ./b
Client IP Address: 127.0.1.1
Server reply: IP: 127, Class: A
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 08$
```