DOMAIN NAME SERVER USING UDP

server code:

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

#define PORT 8080
#define MAXLINE 1024

typedef struct {
    char domain_name[50];
    char ip_address[20];
} DNSRecord;

DNSRecord dns_table[] = {
    {"www.yahoo.com", "10.2.45.67"},
    {"www.annauniv.edu", "197.34.53.122"},
    {"www.google.com", "142.89.78.66"},
};

int table_size = sizeof(dns_table) / sizeof(dns_table[0]);

void display_table() {
    printf("\nServer Name\t\tIP Address\n");
    for (int i = 0; i < table_size; i++) {
        printf("%s\t%s\n", dns_table[i].domain_name, dns_table[i].ip_address);
    }
}

void modify_table() {
    char domain_name[50];
    char ip_address[20];

    printf("\nDo you want to modify (yes or no): ");
    char choice[4];
    scanf("%s", choice);

    if (strcmp(choice, "yes") == 0) {
        printf("\nDomain name: ");
        scanf("%s", domain_name);
        int found = 0;
        for (int i = 0; i < table_size; i++) {
            if (strcmp(dns_table[i].domain_name, domain_name) == 0) {
                found = 1;

                while (1) {
                    printf("\nIP address: ");
                    scanf("%s", ip_address);

                    struct sockaddr_in sa;
                    if (inet_pton(AF_INET, ip_address, &(sa.sin_addr)) == 0) {
                        printf("Invalid IP address\n");
                    } else if (strcmp(dns_table[i].ip_address, ip_address) == 0) {
                        printf("IP address already exists\n");
                    } else {
                        strcpy(dns_table[i].ip_address, ip_address);
                        break;
                    }
                }
                break;
            }
        }

        if (!found) {
            table_size++;
            dns_table[table_size - 1].domain_name[0] = '\0';
            strcpy(dns_table[table_size - 1].domain_name, domain_name);
            strcpy(dns_table[table_size - 1].ip_address, ip_address);
        }

        printf("\nUpdated table is:\n");
        display_table();
    }
}

void handle_client_request(char *domain_name, char *response) {
    for (int i = 0; i < table_size; i++) {
        if (strcmp(dns_table[i].domain_name, domain_name) == 0) {
            strcpy(response, dns_table[i].ip_address);
            return;
        }
    }
    strcpy(response, "Domain name not found");
}
```

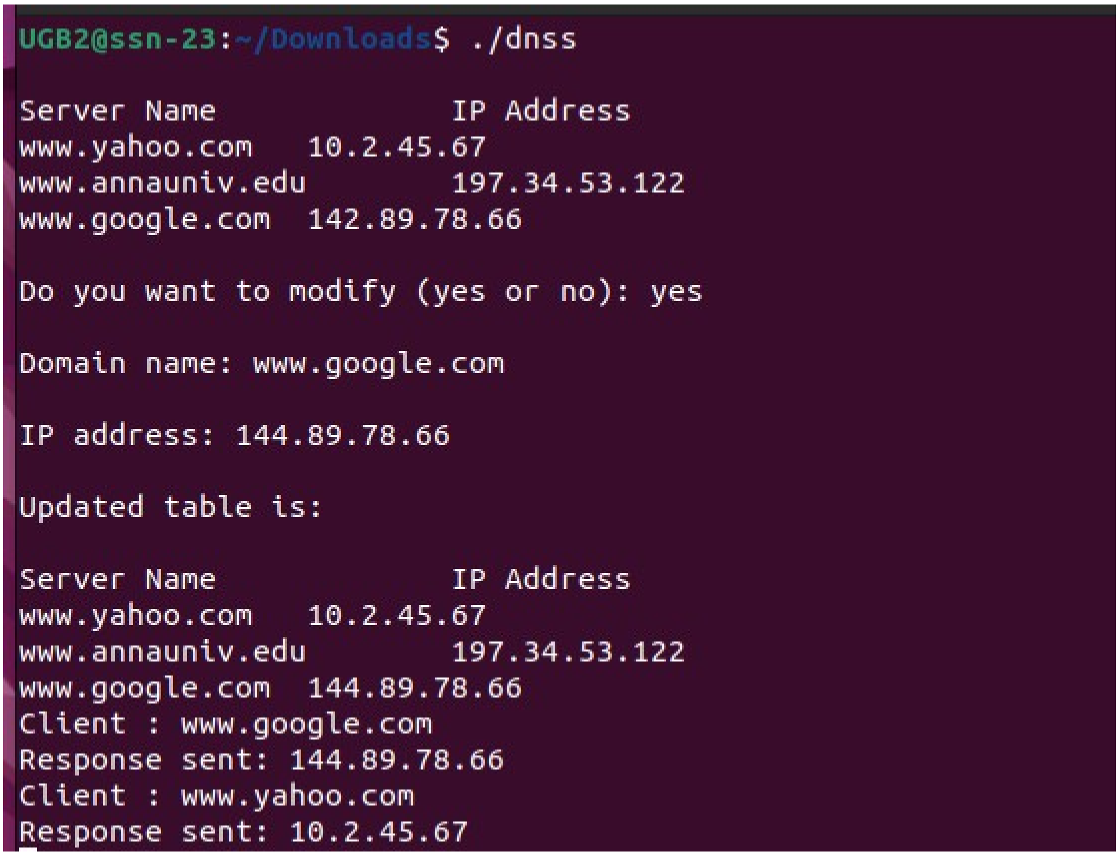
```
int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr, cliaddr;

    // Creating socket file descriptor
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    // Bind the socket with the server address
    if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    display_table();  // Display the initial table
    modify_table();   // Allow modification of the table

    int len, n;

    len = sizeof(cliaddr);  // len is value/result

    while (1) {
        n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct
sockaddr *)&cliaddr, &len);
        buffer[n] = '\0';
        printf("Client : %s\n", buffer);

        char response[MAXLINE];
        handle_client_request(buffer, response);

        sendto(sockfd, (const char *)response, strlen(response), MSG_CONFIRM,
(const struct sockaddr *)&cliaddr, len);
        printf("Response sent: %s\n", response);
    }

    return 0;
}
```

Clinet code

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

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr;

    // Creating socket file descriptor
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;

    while (1) {
        printf("Enter the domain name: ");
        fgets(buffer, MAXLINE, stdin);
        buffer[strcspn(buffer, "\n")] = '\0'; // remove trailing newline

        sendto(sockfd, (const char *)buffer, strlen(buffer), MSG_CONFIRM, (const struct sockaddr *)&servaddr, sizeof(servaddr));
        n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *)&servaddr, &len);
        buffer[n] = '\0';
        printf("Server : %s\n", buffer);
    }

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

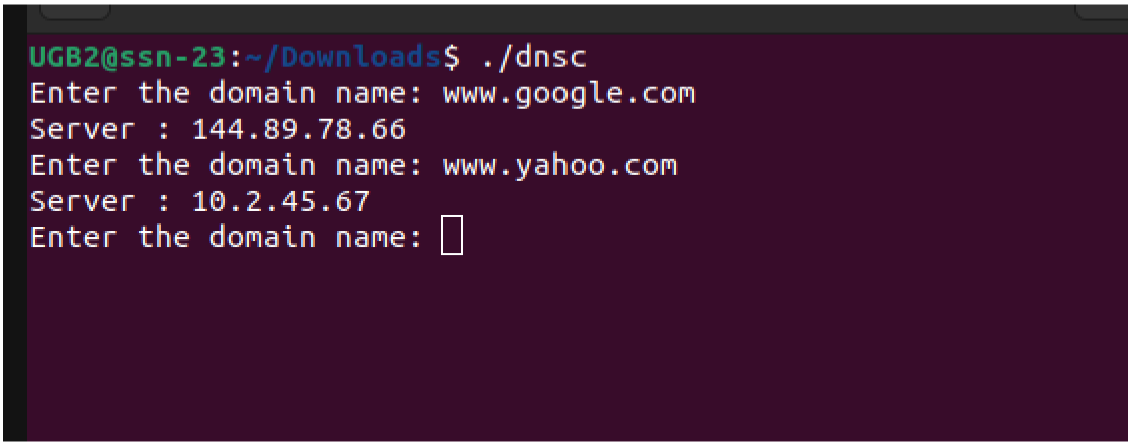
Output:

server:



```
UGB2@ssn-23:~/Downloads$ ./dnss

Server Name             IP Address
www.yahoo.com   10.2.45.67
www.annauniv.edu        197.34.53.122
www.google.com  142.89.78.66

Do you want to modify (yes or no): yes

Domain name: www.google.com

IP address: 144.89.78.66

Updated table is:

Server Name             IP Address
www.yahoo.com   10.2.45.67
www.annauniv.edu        197.34.53.122
www.google.com  144.89.78.66
Client : www.google.com
Response sent: 144.89.78.66
Client : www.yahoo.com
Response sent: 10.2.45.67
```

client:



```
UGB2@ssn-23:~/Downloads$ ./dnsc
Enter the domain name: www.google.com
Server : 144.89.78.66
Enter the domain name: www.yahoo.com
Server : 10.2.45.67
Enter the domain name: 
```