SERVER:

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

#define MAX_ADDRESS 5
#define MAX_DOMAIN 10

struct record{
      char *domain;
      char *address[MAX_ADDRESS];
};

typedef struct record Record;

void init(Record *r){
      r->domain = (char*)calloc(100, sizeof(char));
      for(int i =0;i<MAX_ADDRESS;i++)
            r->address[i] = (char*)calloc(100, sizeof(char));
}

Record *return_address(Record *table, char *domain){

    Record result;
    init(&result);
    strcpy(result.domain, domain);
```

```
    for (int i = 0; i < MAX_DOMAIN; i++){
        if (strcmp(table[i].domain, domain) == 0){
            for (int j = 0; j < MAX_ADDRESS; j++)            {
                strcpy(result.address[j], table[i].address[j]);
            }
            break;
        }
    }
    return (&result);
}

int check_address(Record *table, char *address){

    char* addr_copy = (char*)calloc(100, sizeof(char));
    strcpy(addr_copy, address);
    char *split;
    int val;
    split = strtok(addr_copy, ".");

    while (split){
        val = atoi(split);
        if (val < 0 || val > 255){
            printf("\nInvalid Address!!\n");
            return 0;
        }
        split = strtok(NULL, ".");
    }


    for (int i = 0; i < MAX_DOMAIN; i++){
        if (!table[i].domain[0])
```

```
            continue;

        for (int j = 0; j < MAX_ADDRESS && table[i].address[j][0];
j++)
            if (strcmp(address, table[i].address[j]) == 0){
                printf("\nIP address exists!!\n");
                return 0;
            }
    }

    return 1;
}

int create_record(Record table[MAX_DOMAIN], char *domain, char
*address){

    int ix = -1;
    int flag = 0;

    int addr_valid = check_address(table, address);
    if (!addr_valid)
        return flag;

    for (int i = 0; i < MAX_DOMAIN; i++){
        if (strcmp(table[i].domain, domain) == 0){
            for (int j = 0; j < MAX_DOMAIN; j++)
                if (!table[i].address[j][0]){
                    strcpy(table[i].address[j], address);
                    flag = 1;
                    break;
                }
            break;
        }
```

```
        if (!table[i].domain[0] && ix == -1)
            ix = i;
    }

    if (!flag){
        strcpy(table[ix].domain, domain);
        strcpy(table[ix].address[0], address);
        flag = 1;
    }

    return flag;
}

void Table_lookup(Record table[MAX_DOMAIN]){
    printf(" ========================================= \n");
    printf("|    Domain Name    |      Address        |\n");

    for (int i = 0; i < MAX_DOMAIN; i++){
        if (table[i].domain[0]){
            printf("| %-15s | %-20s |\n", table[i].domain,
table[i].address[0]);

            for (int j = 1; j < MAX_ADDRESS &&
table[i].address[j][0]; j++)
                printf("| %-15s | %-20s |\n", "",
table[i].address[j]);

printf("|==================|======================|\n");
        }
    }
    printf("\n");
}

int main(int argc, char **argv){
```

```
	Record table[MAX_DOMAIN], *result;
	for(int i =0; i<MAX_DOMAIN;i++){
		init(&table[i]);
	}

	if (argc > 1){
		perror("\nNo arguments required!!\n");
		exit(0);
	}


	struct sockaddr_in server_address, client_address;

	char buffer[1024];

	char *domain = (char*)calloc(100, sizeof(char));
	char *address = (char*)calloc(100, sizeof(char));

	int   sockfd = socket(AF_INET, SOCK_DGRAM, 0);
	if (sockfd < 0)
		perror("\nSocket creation failed!!\n");

	bzero(&server_address, sizeof(server_address));

	server_address.sin_family = AF_INET;
	server_address.sin_addr.s_addr = htonl(INADDR_ANY);
	server_address.sin_port = htons(7894);


	if ((bind(sockfd, (struct sockaddr *)&server_address,
sizeof(server_address)))< 0)
		perror("\nBinding Error !!\n");
```

```
	int len = sizeof(client_address);



	create_record(table, "google.com", "192.168.1.1");
	create_record(table, "yahoo.com", "194.12.34.12");
	create_record(table, "google.com", "17.10.23.123");



	int opt=0;
	while(1){
		Table_lookup(table);


		printf("\nDo you want to update table? 1/0:
");scanf("%d", &opt);
		if(opt != 1 )
			break;


		printf("\nEnter domain: ");scanf(" %[^\n]", domain);
		printf("\nEnter address: ");scanf(" %[^\n]", address);


		int rval = create_record(table, domain, address);
		if(rval)
			printf("\nSuccessfully added entry\n");
	}


	printf("\nDNS Server set up\n");


	while(1){
		bzero(&buffer, sizeof(buffer));
		recvfrom(sockfd, buffer, sizeof(buffer), MSG_WAITALL,
(struct sockaddr *)&client_address, &len);
		printf("\n%s\n", buffer);
```

```
            result = return_address(table, &buffer);
            sendto(sockfd, result, sizeof(Record), MSG_CONFIRM,
(struct sockaddr *)&client_address, len);
      }
      close(sockfd);
}
```

CLIENT:

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

#define MAX_ADDRESS 5
#define MAX_DOMAIN 10


struct record{
      char *domain;
      char *address[MAX_ADDRESS];
};

typedef struct record Record;

void init(Record *r){
      r->domain = (char*)calloc(100, sizeof(char));
      for(int i =0;i<MAX_ADDRESS;i++)
            r->address[i] = (char*)calloc(100, sizeof(char));
}



int main(int argc, char **argv){
      printf("\nHI1\n");
```

```
if (argc < 2){
      perror("\nPlease pass IP Address as argument!\n");
      exit(0);
}
printf("\nHI2\n");

struct sockaddr_in server_address;

char buffer[1024];

Record query;
printf("\nHI3\n");

int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockfd < 0){
      perror("\nSocket creation failed!\n");
}

bzero(&server_address, sizeof(server_address));
printf("\nHI4\n");

server_address.sin_family = AF_INET;
printf("\nHI4.1\n");
server_address.sin_addr.s_addr = INADDR_ANY;
printf("\nHI4.2\n");
server_address.sin_port = htons(7894);
printf("\nHI5\n");
int len = sizeof(Record);
while(1){
      printf("\nHI6\n");
      init(&query);
```

```
            printf("\nHI7\n");
            printf("\nEnter the domain name: ");scanf(" %[^\n]",
query.domain);

            if (strcmp(query.domain, "END") == 0)
                  break;

            bzero(&buffer, sizeof(buffer));
            strcpy(buffer, query.domain);
            sendto(sockfd, buffer, sizeof(buffer), MSG_CONFIRM,
(struct sockaddr *)&server_address, sizeof(server_address));
            recvfrom(sockfd, &query, sizeof(Record), MSG_WAITALL,
(struct sockaddr *)&server_address, &len);

            if (!query.address[0][0])
                  printf("\nNo entry in DNS!\n");
            else{
                  printf("\nThe IP Address is: ");
                  for (int i = 0; i < MAX_ADDRESS; i++){
                        if (query.address[i][0])
                              printf("%s\n", query.address[i]);
                  }
                  printf("\n");
            }
      }

      close(sockfd);
}
```