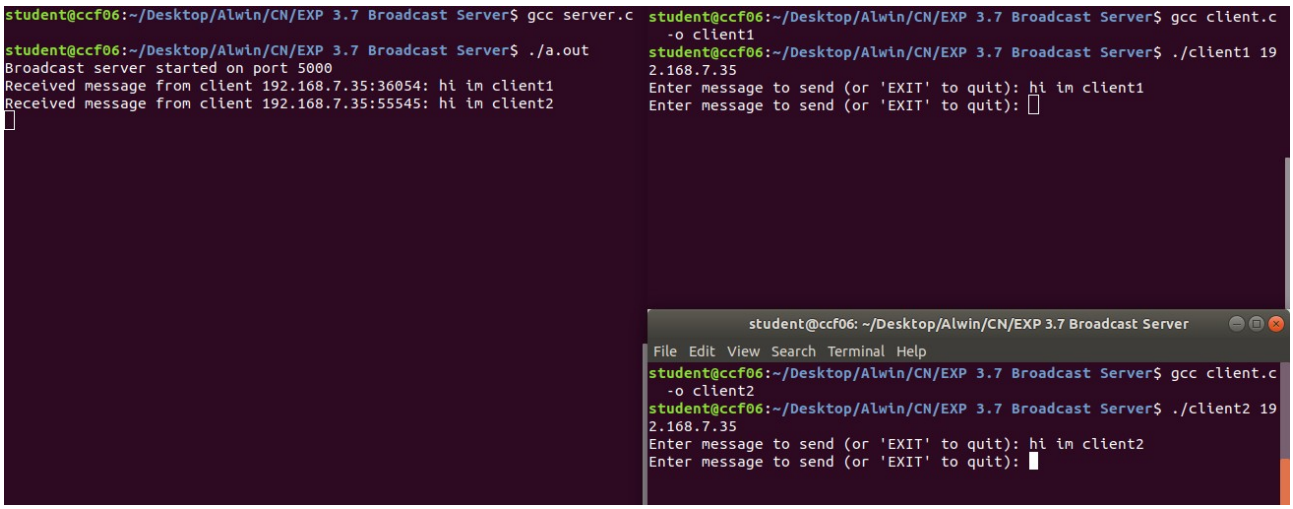
CLIENT

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

#define PORT 5000
#define BUFFER_SIZE 1024

// Define the maximum number of clients that can connect (adjust as needed)
#define MAX_CLIENTS 10

// Structure to store client information
typedef struct {
  int socket;
  struct sockaddr_in address;
} Client;

int main(int argc, char *argv[]) {
  if (argc < 1) {
    fprintf(stderr, "Usage: %s\n", argv[0]);
    exit(1);
  }

  int server_socket;
  struct sockaddr_in server_address;
  Client clients[MAX_CLIENTS]; // Array to store connected clients
  int num_clients = 0;

  // Create a UDP socket
  if ((server_socket = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
    perror("socket() failed");
    exit(1);
  }

  // Set up server address structure
  server_address.sin_family = AF_INET;
  server_address.sin_addr.s_addr = INADDR_ANY; // Listen on all interfaces
  server_address.sin_port = htons(PORT);

  // Bind the socket to the address
  if (bind(server_socket, (struct sockaddr *) &server_address, sizeof(server_address)) == -1) {
    perror("bind() failed");
    exit(1);
  }

  printf("Broadcast server started on port %d\n", PORT);
```

```
  char message[BUFFER_SIZE];

  while (1) {
    struct sockaddr_in client_address;
    socklen_t client_address_size = sizeof(client_address);

    int bytes_received = recvfrom(server_socket, message, BUFFER_SIZE - 1, 0,
                                  (struct sockaddr *) &client_address, &client_address_size);
    if (bytes_received == -1) {
      perror("recvfrom() failed");
      exit(1);
    }
    message[bytes_received] = '\0'; // Null-terminate the received message
    int client_found = 0;
    for (int i = 0; i < num_clients; i++) {
      if (clients[i].socket == client_address.sin_port) {
        client_found = 1;
        break;
      }
    }

    if (!client_found) {
      if (num_clients >= MAX_CLIENTS) {
        printf("Maximum number of clients reached\n");
        continue;
      }
      clients[num_clients].socket = client_address.sin_port;
      clients[num_clients].address = client_address;
      num_clients++;
    }

    printf("Received message from client %s:%d: %s\n",
           inet_ntoa(client_address.sin_addr), ntohs(client_address.sin_port), message);
    for (int i = 0; i < num_clients; i++) {
      if (clients[i].socket != client_address.sin_port) {
        if (sendto(server_socket, message, strlen(message), 0,
                   (struct sockaddr *) &clients[i].address, sizeof(clients[i].address)) == -1) {
          perror("sendto() failed");
          exit(1);
        }
      }
    }
    if (strcmp(message, "EXIT") == 0) {
      break;
    }
  }

  // Close the socket and free resources
  close(server_socket);

  return 0;
}
```

SERVER

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

#define PORT 5000
#define BUFFER_SIZE 1024

int main(int argc, char *argv[]) {
  if (argc < 2) {
    fprintf(stderr, "Usage: %s <server_IP>\n", argv[0]);
    exit(1);
  }

  int client_socket;
  struct sockaddr_in server_address;

  if ((client_socket = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
    perror("socket() failed");
    exit(1);
  }
  server_address.sin_family = AF_INET;
  server_address.sin_port = htons(PORT);
  inet_pton(AF_INET, argv[1], &server_address.sin_addr); // Convert server IP from string

  char message[BUFFER_SIZE];

  while (1) {
    printf("Enter message to send (or 'EXIT' to quit): ");
    fgets(message, BUFFER_SIZE, stdin);
    message[strcspn(message, "\n")] = '\0';
    if (sendto(client_socket, message, strlen(message), 0,
               (struct sockaddr *) &server_address, sizeof(server_address)) == -1) {
      perror("sendto() failed");
      exit(1);
    }

    if (strcmp(message, "EXIT") == 0) {
      break;
    }
  close(client_socket);

  return 0;
}
```


student@ccf06:~/Desktop/Alwin/CN/EXP 3.7 Broadcast Server$ gcc server.c
student@ccf06:~/Desktop/Alwin/CN/EXP 3.7 Broadcast Server$ ./a.out
Broadcast server started on port 5000
Received message from client 192.168.7.35:36054: hi im client1
Received message from client 192.168.7.35:55545: hi im client2
student@ccf06:~/Desktop/Alwin/CN/EXP 3.7 Broadcast Server$ gcc client.c -o client1
student@ccf06:~/Desktop/Alwin/CN/EXP 3.7 Broadcast Server$ ./client1 192.168.7.35
Enter message to send (or 'EXIT' to quit): hi im client1
Enter message to send (or 'EXIT' to quit):
student@ccf06: ~/Desktop/Alwin/CN/EXP 3.7 Broadcast Server
File Edit View Search Terminal Help
student@ccf06:~/Desktop/Alwin/CN/EXP 3.7 Broadcast Server$ gcc client.c -o client2
student@ccf06:~/Desktop/Alwin/CN/EXP 3.7 Broadcast Server$ ./client2 192.168.7.35
Enter message to send (or 'EXIT' to quit): hi im client2
Enter message to send (or 'EXIT' to quit):