# CSL 332 Networking Lab Model Questions

## 1. Create a program to implement an echo server using TCP.

**server.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock, client_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 char buffer[1024];
 int n;
 server_sock = socket(AF_INET, SOCK_STREAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+]TCP server socket created.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+]Bind to the port number: %d\n", port);
 listen(server_sock, 5);
 printf("Listening for Connections...\n");
 addr_size = sizeof(client_addr);
 client_sock = accept(server_sock, (struct sockaddr *)&client_addr, &addr_size);
 printf("[+]Client connected.\n");
 while (1)
 {
 bzero(buffer, 1024);
 recv(client_sock, buffer, sizeof(buffer), 0);
 printf("Client: %s\n", buffer);
 send(client_sock, buffer, strlen(buffer), 0);
 printf("Server: %s\n", buffer);
 }
 close(client_sock);
 printf("[+]Client disconnected.\n\n");
 return 0;
}
```

**client.c**

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

int main() {
 char ip[] = "127.0.0.1";
 int port = 5566;

 int sock;
 struct sockaddr_in addr;
 socklen_t addr_size;
 char buffer[1024];
 int n;
 sock = socket(AF_INET, SOCK_STREAM, 0);
 if (sock < 0) {
 perror("[-]Socket error");
 exit(1);
 }
 printf("[+]TCP client socket created.\n");
 memset(&addr, '\0', sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_port = htons(port);
 addr.sin_addr.s_addr = inet_addr(ip);
 connect(sock, (struct sockaddr *)&addr, sizeof(addr));
 printf("Connected to the server.\n");
 while (1) {
 bzero(buffer, 1024);
 printf("Enter message: ");
 fgets(buffer, 1024, stdin);
 printf("Client: %s\n", buffer);
 send(sock, buffer, strlen(buffer), 0);
 bzero(buffer, 1024);
 recv(sock, buffer, sizeof(buffer), 0);
 printf("Server: %s\n", buffer);
 }
 close(sock);
 printf("Disconnected from the server.\n");

 return 0;
}
```

2. **Create a program to implement a server using TCP to reverse a string.**

**server.c**

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

void reverseString(char *str)
{
 int length = strlen(str);
 int start = 0;
 int end = length - 1;
 while (start < end)
 {
 char temp = str[start];
 str[start] = str[end];
 str[end] = temp;
 start++;
 end--;
 }
}

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock, client_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 server_sock = socket(AF_INET, SOCK_STREAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+]TCP server socket created.\n");
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+]Bind to the port number: %d\n", port);
 listen(server_sock, 5);
 printf("Listening for Connections...\n");
 addr_size = sizeof(client_addr);

 client_sock = accept(server_sock, (struct sockaddr *)&client_addr, &addr_size);
 printf("[+]Client connected.\n");

 char buffer[1024];

 recv(client_sock, buffer, sizeof(buffer), 0);
 printf("Client: %s\n", buffer);

 reverseString(buffer);

 send(client_sock, buffer, strlen(buffer), 0);

 close(client_sock);
 printf("[+]Client disconnected.\n\n");
 close(server_sock);
 printf("[+]Server closed.\n");

 return 0;
}
```

**client.c**

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

```
#include <unistd.h>
#include <arpa/inet.h>

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;

 int sock;
 struct sockaddr_in addr;
 socklen_t addr_size;
 int n;
 sock = socket(AF_INET, SOCK_STREAM, 0);
 if (sock < 0)
 {
 perror("[-]Socket error");
 exit(1);
 }
 printf("[+]TCP client socket created.\n");
 memset(&addr, '\0', sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_port = htons(port);
 addr.sin_addr.s_addr = inet_addr(ip);
 connect(sock, (struct sockaddr *)&addr, sizeof(addr));
 printf("Connected to the server.\n");

 char buffer[1024];
 printf("Enter a string to reverse: ");
 fgets(buffer, sizeof(buffer), stdin);

 send(sock, buffer, strlen(buffer), 0);

 recv(sock, buffer, sizeof(buffer), 0);
 printf("Reversed string: %s\n", buffer);

 close(sock);
 printf("Disconnected from the server.\n");

 return 0;
}
```

## 3. Create a program to implement a server using TCP to reverse a number

**server.c**

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

int reverseNumber(int num)
{
 int reversed = 0;
 while (num != 0)
 {
 int digit = num % 10;
 reversed = reversed * 10 + digit;
 num /= 10;
 }
 return reversed;
}

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock, client_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 server_sock = socket(AF_INET, SOCK_STREAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+]TCP server socket created.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);
 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+]Bind to the port number: %d\n", port);
 listen(server_sock, 5);
 printf("Listening for Connections...\n");
 addr_size = sizeof(client_addr);

 client_sock = accept(server_sock, (struct sockaddr *)&client_addr, &addr_size);
 printf("[+]Client connected.\n");

 int number;

 recv(client_sock, &number, sizeof(number), 0);
 printf("Client: %d\n", number);

 int reversed = reverseNumber(number);

 send(client_sock, &reversed, sizeof(reversed), 0);

 close(client_sock);
 printf("[+]Client disconnected.\n\n");
 close(server_sock);
 printf("[+]Server closed.\n");

 return 0;
}
```

**client.c**

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

```
int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;

 int sock;
 struct sockaddr_in addr;
 socklen_t addr_size;
 int n;
 sock = socket(AF_INET, SOCK_STREAM, 0);
 if (sock < 0)
 {
 perror("[-]Socket error");
 exit(1);
 }
 printf("[+]TCP client socket created.\n");
 memset(&addr, '\0', sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_port = htons(port);
 addr.sin_addr.s_addr = inet_addr(ip);
 connect(sock, (struct sockaddr *)&addr, sizeof(addr));
 printf("Connected to the server.\n");

 int number;
 printf("Enter a number to reverse: ");
 scanf("%d", &number);

 send(sock, &number, sizeof(number), 0);

 int reversed;
 recv(sock, &reversed, sizeof(reversed), 0);
 printf("Reversed number: %d\n", reversed);

 close(sock);
 printf("Disconnected from the server.\n");

 return 0;
}
```

## 4. Create a program to implement a server using TCP to find the factorial of a number

**server.c**

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

int factorial(int n)
{
 if (n == 0)
 return 1;
 else
 return n * factorial(n - 1);
}

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock, client_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 server_sock = socket(AF_INET, SOCK_STREAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+]TCP server socket created.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
```

```
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+]Bind to the port number: %d\n", port);
 listen(server_sock, 5);
 printf("\nListening for Connections...");
 addr_size = sizeof(client_addr);

 client_sock = accept(server_sock, (struct sockaddr *)&client_addr, &addr_size);
 printf("\n[+]Client connected.");
 int num;

 recv(client_sock, &num, sizeof(num), 0);
 printf("Client: %d\n", num);

 int result = factorial(num);

 send(client_sock, &result, sizeof(result), 0);

 close(client_sock);
 printf("[+]Client disconnected.\n\n");
 return 0;
}
```

**client.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;

 int sock;
 struct sockaddr_in addr;
 socklen_t addr_size;
 int n;
 sock = socket(AF_INET, SOCK_STREAM, 0);
 if (sock < 0)
 {
 perror("[-]Socket error");
 exit(1);
 }
 printf("[+]TCP client socket created.\n");
 memset(&addr, '\0', sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_port = htons(port);
 addr.sin_addr.s_addr = inet_addr(ip);
 connect(sock, (struct sockaddr *)&addr, sizeof(addr));
 printf("Connected to the server.\n");
 int num;
 printf("Enter a number to calculate its factorial: ");
 scanf("%d", &num);

 send(sock, &num, sizeof(num), 0);

 int result;
 recv(sock, &result, sizeof(result), 0);
 printf("Factorial of %d: %d\n", num, result);
 close(sock);
 printf("Disconnected from the server.\n");
 return 0;
}
```

5. **Create a program to implement a chat server using UDP in which the client sends a message first and the “$” string ends the chat.**

**server.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 char buffer[1024];

 server_sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+] UDP server socket created.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+] Bind to the port number: %d\n", port);
 printf("\nChat Server is running...\n");
 addr_size = sizeof(client_addr);
 while (1)
 {

 bzero(buffer, 1024);

 n = recvfrom(server_sock, buffer, sizeof(buffer), 0, (struct sockaddr
*)&client_addr, &addr_size);
 if (n < 0)
 {
 perror("[-] Receive Error");
 exit(1);
 }
 printf("Received from client: %s", buffer);
 bzero(buffer, 1024);

 printf("Enter message for client: ");
 fgets(buffer, sizeof(buffer), stdin);
 if (strstr(buffer, "$") != NULL)
 {
 printf("Chat ended by server.\n");
 break;
 }

 n = sendto(server_sock, buffer, strlen(buffer), 0, (struct sockaddr
*)&client_addr, addr_size);
 if (n < 0)
 {
 perror("[-] Send Error");
 exit(1);
 }
 }

 close(server_sock);
 printf("[+] Server closed.\n");

 return 0;
}
```

**client.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int client_sock;
 struct sockaddr_in server_addr;
 socklen_t addr_size;
 int n;
 char buffer[1024];

 client_sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (client_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+] UDP client socket created.\n");

 memset(&server_addr, '\0', sizeof(server_addr));
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 while (1)
 {

 bzero(buffer, 1024);

 printf("Enter message for server: ");
 fgets(buffer, sizeof(buffer), stdin);
 if (strstr(buffer, "$") != NULL)
 {
```

```
 printf("Chat ended by client.\n");
 break;
 }

 n = sendto(client_sock, buffer, strlen(buffer), 0, (struct sockaddr
*)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Send Error");
 exit(1);
 }

 addr_size = sizeof(server_addr);
 bzero(buffer, 1024);

 n = recvfrom(client_sock, buffer, sizeof(buffer), 0, (struct sockaddr
*)&server_addr, &addr_size);
 if (n < 0)
 {
 perror("[-] Receive Error");
 exit(1);
 }
 printf("Received from server: %s", buffer);
 }

 close(client_sock);
 printf("[+] Client closed.\n");

 return 0;
}
```

## 6. Create a program to implement an echo server using UDP

**server.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 char buffer[1024];

 server_sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+] UDP server socket created.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+] Bind to the port number: %d\n", port);

 printf("\nListening for Messages...\n");
 addr_size = sizeof(client_addr);
 while (1)
 {

 bzero(buffer, 1024);
 n = recvfrom(server_sock, buffer, sizeof(buffer), 0, (struct sockaddr
*)&client_addr, &addr_size);
 if (n < 0)
 {
 perror("[-] Receive Error");
 exit(1);
 }
 printf("Received from client: %s", buffer);

 n = sendto(server_sock, buffer, strlen(buffer), 0, (struct sockaddr
*)&client_addr, addr_size);
 if (n < 0)
 {
 perror("[-] Send Error");
 exit(1);
 }
 printf("Sent to client: %s", buffer);
 }

 close(server_sock);
 printf("[+] Server closed.\n");

 return 0;
}
```

**client.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int client_sock;
 struct sockaddr_in server_addr;
 socklen_t addr_size;
 int n;
 char buffer[1024];

 client_sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (client_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+] UDP client socket created.\n");

 memset(&server_addr, '\0', sizeof(server_addr));
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 while (1)
 {

 bzero(buffer, 1024);

 printf("Enter message: ");
 fgets(buffer, sizeof(buffer), stdin);

 n = sendto(client_sock, buffer, strlen(buffer), 0, (struct sockaddr
*)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Send Error");
 exit(1);
 }

 addr_size = sizeof(server_addr);
 bzero(buffer, 1024);
 n = recvfrom(client_sock, buffer, sizeof(buffer), 0, (struct sockaddr
*)&server_addr, &addr_size);
```

```
 if (n < 0)
 {
 perror("[-] Receive Error");
 exit(1);
 }
 printf("Received from server: %s", buffer);
 }

 close(client_sock);
 printf("[+] Client closed.\n");

 return 0;
}
```

7. **Create a program to implement using UDP to reverse a number.**

**server.c**

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

int reverseNumber(int num)
{
 int reversed = 0;
 while (num != 0)
 {
 int digit = num % 10;
 reversed = reversed * 10 + digit;
 num /= 10;
 }
 return reversed;
}

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 server_sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+]UDP server socket created.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);
 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+]Bind to the port number: %d\n", port);

 addr_size = sizeof(client_addr);
 int number;

 recvfrom(server_sock, &number, sizeof(number), 0, (struct sockaddr *)&client_addr,
&addr_size);
 printf("Client: %d\n", number);

 int reversed = reverseNumber(number);

 sendto(server_sock, &reversed, sizeof(reversed), 0, (struct sockaddr
*)&client_addr, addr_size);

 close(server_sock);
 printf("[+]Server closed.\n");

 return 0;
}
```

**client.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int sock;
 struct sockaddr_in server_addr;
 socklen_t addr_size;
 int n;
 sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (sock < 0)
 {
 perror("[-]Socket error");
 exit(1);
 }
 printf("[+]UDP client socket created.\n");
 memset(&server_addr, '\0', sizeof(server_addr));
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 int number;
 printf("Enter a number to reverse: ");
 scanf("%d", &number);

 sendto(sock, &number, sizeof(number), 0, (struct sockaddr *)&server_addr,
sizeof(server_addr));
 printf("Number sent to server.\n");

 int reversed;
 addr_size = sizeof(server_addr);

 recvfrom(sock, &reversed, sizeof(reversed), 0, (struct sockaddr *)&server_addr,
&addr_size);
 printf("Reversed number: %d\n", reversed);

 close(sock);
 printf("Disconnected from the server.\n");

 return 0;
}
```

## 8. Create a program to implement using UDP to find the factorial of a number

**server.c**

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

int factorial(int n)
{
 if (n == 0)
 return 1;
 else
 return n * factorial(n - 1);
}

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 server_sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+]UDP server socket created.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
```

```
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+]Bind to the port number: %d\n", port);

 addr_size = sizeof(client_addr);
 int number;

 recvfrom(server_sock, &number, sizeof(number), 0, (struct sockaddr *)&client_addr,
&addr_size);
 printf("Client: %d\n", number);

 int result = factorial(number);

 sendto(server_sock, &result, sizeof(result), 0, (struct sockaddr *)&client_addr,
addr_size);

 close(server_sock);
 printf("[+]Server closed.\n");

 return 0;
}
```

**client.c**

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

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;

 int sock;
 struct sockaddr_in server_addr;
 socklen_t addr_size;
 int n;
 sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (sock < 0)
 {
 perror("[-]Socket error");
 exit(1);
 }
 printf("[+]UDP client socket created.\n");
 memset(&server_addr, '\0', sizeof(server_addr));
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 int number;
 printf("Enter a number to calculate its factorial: ");
 scanf("%d", &number);

 sendto(sock, &number, sizeof(number), 0, (struct sockaddr *)&server_addr,
sizeof(server_addr));
 printf("Number sent to server.\n");

 int result;
 addr_size = sizeof(server_addr);

 recvfrom(sock, &result, sizeof(result), 0, (struct sockaddr *)&server_addr,
&addr_size);
 printf("Factorial of %d: %d\n", number, result);

 close(sock);
 printf("Disconnected from the server.\n");

 return 0;
}
```

## 9. Create a program to implement using UDP to reverse a string

**server.c**

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

void reverseString(char *str)
{
 int length = strlen(str);
 int start = 0;
 int end = length - 1;
 while (start < end)
 {
 char temp = str[start];
 str[start] = str[end];
 str[end] = temp;
 start++;
 end--;
 }
}

int main()
{
 char ip[] = "127.0.0.1";
 int port = 5566;
 int server_sock;
 struct sockaddr_in server_addr, client_addr;
 socklen_t addr_size;
 int n;
 server_sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (server_sock < 0)
 {
 perror("[-] Socket Error");
 exit(1);
 }
 printf("[+]UDP server socket created.\n");
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 n = bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
 if (n < 0)
 {
 perror("[-] Bind Error");
 exit(1);
 }
 printf("[+]Bind to the port number: %d\n", port);

 addr_size = sizeof(client_addr);
 char buffer[1024];

 recvfrom(server_sock, buffer, sizeof(buffer), 0, (struct sockaddr *)&client_addr,
&addr_size);
 printf("Client: %s\n", buffer);

 reverseString(buffer);

 sendto(server_sock, buffer, strlen(buffer), 0, (struct sockaddr *)&client_addr,
addr_size);

 close(server_sock);
 printf("[+]Server closed.\n");

 return 0;
}
```

**client.c**

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

int main() {
 char ip[] = "127.0.0.1";
 int port = 5566;
 int sock;
 struct sockaddr_in server_addr;
 socklen_t addr_size;
 int n;
 sock = socket(AF_INET, SOCK_DGRAM, 0);
 if (sock < 0) {
 perror("[-]Socket error");
 exit(1);
 }
 printf("[+]UDP client socket created.\n");
 memset(&server_addr, '\0', sizeof(server_addr));
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(port);
 server_addr.sin_addr.s_addr = inet_addr(ip);

 char buffer[1024];
 printf("Enter a string to reverse: ");
 fgets(buffer, sizeof(buffer), stdin);

 sendto(sock, buffer, strlen(buffer), 0, (struct sockaddr *)&server_addr,
sizeof(server_addr));
 printf("String sent to server.\n");

 addr_size = sizeof(server_addr);

 recvfrom(sock, buffer, sizeof(buffer), 0, (struct sockaddr *)&server_addr,
&addr_size);
 printf("Reversed string: %s\n", buffer);

 close(sock);
 printf("Disconnected from the server.\n");

 return 0;
}
```