**Devang Raj Arora - 180953336 - NDP Lab 1,2,3**

**Lab 1**

**Q1**

**Client**

**#include <stdio.h>**

**#include <sys/socket.h>**

**#include <sys/types.h>**

**#include <arpa/inet.h>**

**#include <unistd.h>**

**#include <string.h>**

**#define MAX\_SIZE 1024**

**int main() {**

**int clientSocket;**

**struct sockaddr\_in serverAddr;**

**char message[MAX\_SIZE];**

**clientSocket = socket(AF\_INET, SOCK\_STREAM, 0);**

**serverAddr.sin\_family = AF\_INET;**

**serverAddr.sin\_port = htons(5000);**

**serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");**

**memset(serverAddr.sin\_zero, '\0', sizeof(serverAddr.sin\_zero));**

**int res = connect(clientSocket, (struct sockaddr \*)&serverAddr, sizeof(serverAddr));**

**if(res == -1) {**

**printf("Error in connecting at client\n");**

**return 0;**

**}**

**printf("\nChoose Option:\n1. Search in array\n2. Sort array\n3. Split array into odd and even\n4. Exit\n");**

**while(1) {**

**int opt, key = -1, i;**

**printf("\nEnter option:");**

**scanf("%d", &opt);**

**send(clientSocket, &opt, sizeof(opt), 0);**

**if(opt == 4) {**

**break;**

**}**

**printf("Enter number of elements: ");**

**int n;**

**scanf("%d", &n);**

**send(clientSocket, &n, sizeof(n), 0);**

**int numbers[n];**

**printf("Enter list of numbers: ");**

**for(i = 0 ; i < n ; i++) {**

**scanf("%d", &numbers[i]);**

**}**

**if(opt == 1) {**

**printf("Enter value to search: ");**

**scanf("%d", &key);**

**}**

**send(clientSocket, numbers, sizeof(numbers), 0);**

**send(clientSocket, &key, sizeof(key), 0);**

**recv(clientSocket, message, sizeof(message), 0);**

**printf("\n%s\n", message);**

**}**

**close(clientSocket);**

**}**

**Server**

**#include <stdio.h>**

**#include <sys/socket.h>**

**#include <sys/types.h>**

**#include <arpa/inet.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <math.h>**

**#include <stdlib.h>**

**#define MAX\_SIZE 1024**

**int cmpfunc (const void \* a, const void \* b) {**

**return ( \*(int\*)a - \*(int\*)b );**

**}**

**char \* toCharArray(int number)**

**{**

**int n = floor(log10(number)) + 1;**

**int i;**

**char \*numberArray = (char\*)calloc(n, sizeof(char));**

**for (i = n-1; i >= 0; --i, number /= 10)**

**{**

**numberArray[i] = (number % 10) + '0';**

**}**

**return numberArray;**

**}**

**int i;**

**int linearSearch(int \*a, int n, int key) {**

**for(i = 0 ; i < n ; i++) {**

**if(a[i] == key) {**

**return i;**

**}**

**}**

**return -1;**

**}**

**int main() {**

**int mainSocket, newSocket;**

**struct sockaddr\_in serverAddr, clientAddr;**

**mainSocket = socket(AF\_INET, SOCK\_STREAM, 0);**

**serverAddr.sin\_family = AF\_INET;**

**serverAddr.sin\_port = htons(5000);**

**serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");**

**memset(serverAddr.sin\_zero, '\0',sizeof(serverAddr.sin\_zero));**

**int res = bind(mainSocket, (struct sockaddr\*) &serverAddr, sizeof(serverAddr));**

**if(res == -1) {**

**printf("Error at server in binding\n");**

**return 0;**

**}**

**res = listen(mainSocket, 5);**

**if(res == 0) {**

**printf("Listening at server\n");**

**} else {**

**printf("Error at server in listening\n");**

**}**

**socklen\_t addr\_size = sizeof(clientAddr);**

**newSocket = accept(mainSocket, (struct sockaddr\*) &clientAddr, &addr\_size);**

**while(1) {**

**int key, opt, n, i;**

**recv(newSocket, &opt, sizeof(opt), 0);**

**if(opt == 4) {**

**break;**

**}**

**recv(newSocket, &n, sizeof(n), 0);**

**int numbers[n];**

**recv(newSocket, numbers, sizeof(numbers), 0);**

**recv(newSocket, &key, sizeof(key), 0);**

**for(i = 0 ; i < n ; i++) {**

**printf("%d ",numbers[i]);**

**}**

**printf("\nOption recieved from client: %d", opt);**

**printf("\nKey recieved from client: %d\n", key);**

**char message[MAX\_SIZE];**

**strcpy(message,"");**

**if(opt == 1) {**

**int idx = linearSearch(numbers, n, key);**

**if(idx == -1) {**

**strcpy(message,"Value not found");**

**} else {**

**strcpy(message,"Value found");**

**}**

**} else if(opt == 2) {**

**qsort(numbers, n, sizeof(int), cmpfunc);**

**strcpy(message,"Sorted Array: ");**

**for(i = 0 ; i < n ; i++) {**

**char \*temp =**

**strcat(message, toCharArray(numbers[i]));**

**strcat(message, " ");**

**}**

**} else {**

**char odd[MAX\_SIZE], even[MAX\_SIZE];**

**strcpy(odd,"");**

**strcpy(even,"");**

**for(i = 0 ; i < n ; i++) {**

**if(numbers[i]%2) {**

**strcat(odd, toCharArray(numbers[i]));**

**strcat(odd, " ");**

**} else {**

**strcat(even, toCharArray(numbers[i]));**

**strcat(even, " ");**

**}**

**}**

**strcat(message, "Odd: ");**

**strcat(message, odd);**

**strcat(message, "\nEven: ");**

**strcat(message, even);**

**}**

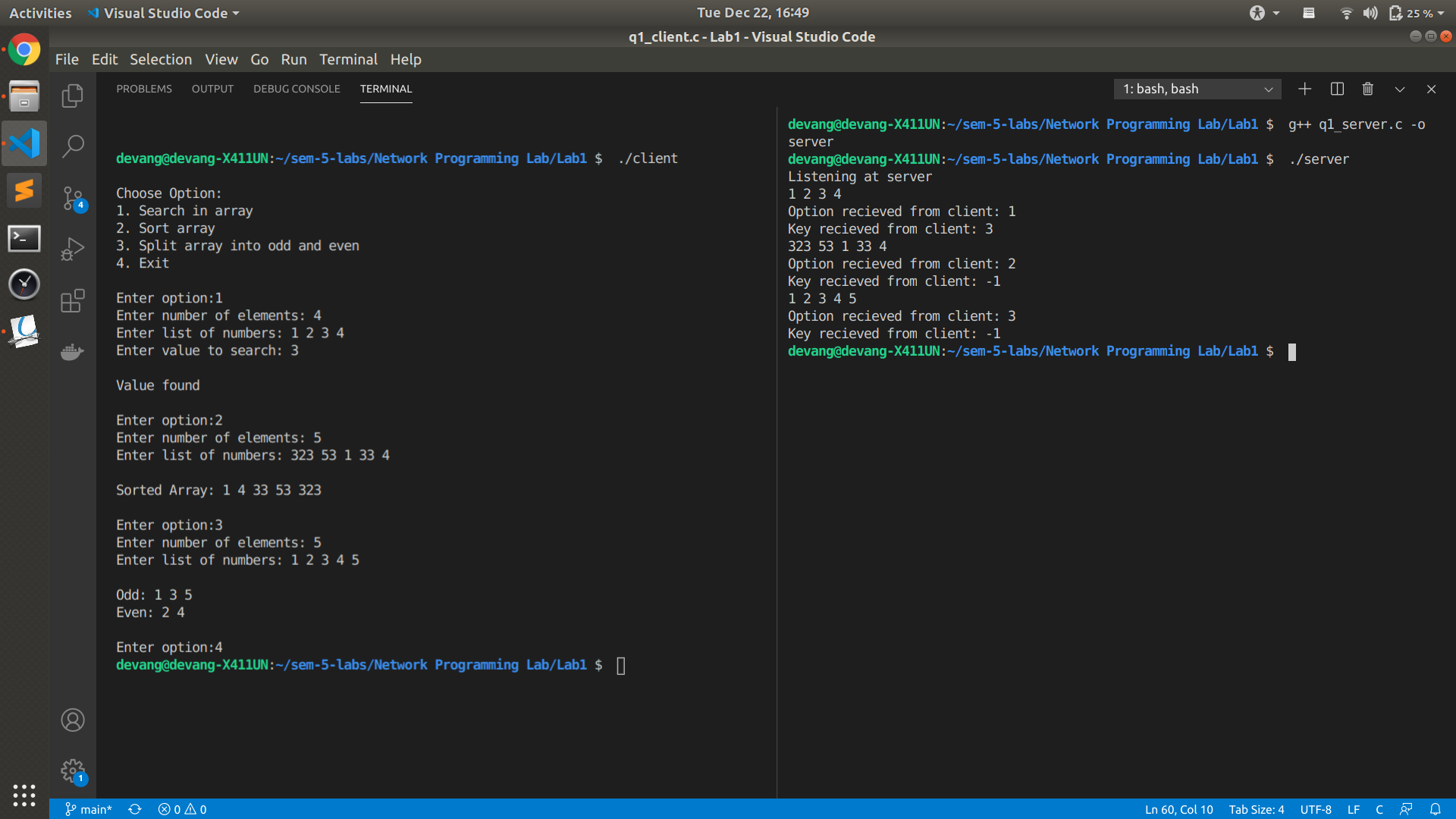
**send(newSocket, message, sizeof(message), 0);**

**}**

**close(newSocket);**

**close(mainSocket);**

**}**



Q2

**TCP**

**Server**

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <fcntl.h>

#include <sys/stat.h>

int main()

{

int welcomeSocket, newSocket, desc, data, recvB, n;

int i, strLen, vowelCount[5], isPalin;

char charBuff[100];

socklen\_t len;

struct sockaddr\_in server, client;

welcomeSocket = socket(AF\_INET, SOCK\_STREAM, 0);

if (welcomeSocket == -1)

{

printf("\nSocket Creation Error\n");

exit(0);

}

printf("\nSocket Creation Successful\n");

server.sin\_family = AF\_INET;

server.sin\_port = htons(5000);

server.sin\_addr.s\_addr = htonl(INADDR\_ANY);

desc = bind(welcomeSocket, (struct sockaddr\*)&server, sizeof(server));

desc = listen(welcomeSocket, 1);

len = sizeof(client);

newSocket = accept(welcomeSocket, (struct sockaddr\*)&client, &len);

while (1)

{

recvB = recv(newSocket, charBuff, sizeof(charBuff), 0);

if (strcmp(charBuff, "Halt\n") == 0)

{

break;

}

strLen = 0;

for (i = 0; charBuff[i] != '\0'; i++);

strLen = i - 1;

isPalin = 1;

for (i = 0; i < strLen / 2; i++) {

if (charBuff[i] != charBuff[strLen - i - 1])

{

isPalin = 0;

break;

}

}

for (i = 0; i < 5; i++)

vowelCount[i] = 0;

for (i = 0; i < strLen; i++)

{

if (charBuff[i] == 'a' || charBuff[i] == 'A')

vowelCount[0]++;

else if (charBuff[i] == 'e' || charBuff[i] == 'E')

vowelCount[1]++;

if (charBuff[i] == 'i' || charBuff[i] == 'I')

vowelCount[2]++;

if (charBuff[i] == 'o' || charBuff[i] == 'O')

vowelCount[3]++;

if (charBuff[i] == 'u' || charBuff[i] == 'U')

vowelCount[4]++;

}

data = send(newSocket, &isPalin, sizeof(int), 0);

data = send(newSocket, &strLen, sizeof(int), 0);

data = send(newSocket, vowelCount, sizeof(vowelCount), 0);

}

close(newSocket);

close(welcomeSocket);

}

**Client**

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <fcntl.h>

#include <sys/stat.h>

int main()

{

int welcomeSocket, desc, data, recvB, n;

char charBuff[100];

char f = getchar();

struct sockaddr\_in server;

welcomeSocket = socket(AF\_INET, SOCK\_STREAM, 0);

server.sin\_family = AF\_INET;

server.sin\_port = htons(5000);

server.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

desc = connect(welcomeSocket, (struct sockaddr\*)&server, sizeof(server));

while (1)

{

fgets(charBuff, 100, stdin);

data = send(welcomeSocket, charBuff, sizeof(charBuff), 0);

if (strcmp(charBuff, "Halt\n") == 0)

{

printf("\nClient bids goodbye\n");

close(welcomeSocket);

exit(0);

}

int strLen, isPalin, vowelCount[5];

recvB = recv(welcomeSocket, &isPalin, sizeof(int), 0);

if (isPalin == 1)

printf("\nString is a Palindrome\n");

else

{

printf("\nString is not a Palindrome\n");

}

recvB = recv(welcomeSocket, &strLen, sizeof(int), 0);

printf("\nLength of string : %d\n", strLen);

recvB = recv(welcomeSocket, vowelCount, sizeof(vowelCount), 0);

if (recvB == -1)

{

printf("\nVowel Count Message recieving failed\n");

exit(0);

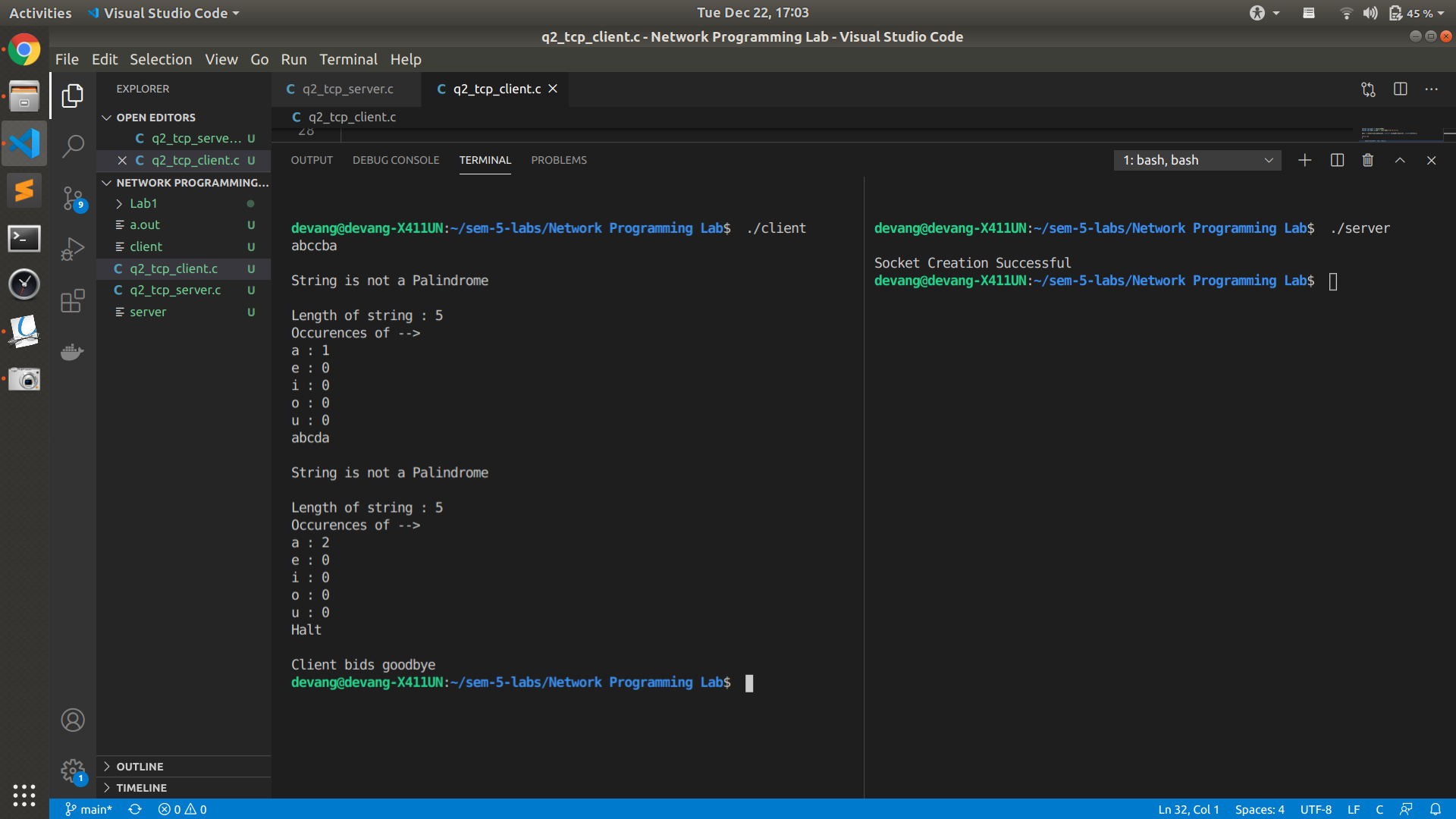
}

printf("Occurences of -->\na : %d\ne : %d\ni : %d\no : %d\nu : %d\n", vowelCount[0], vowelCount[1], vowelCount[2], vowelCount[3], vowelCount[4]);

}

close(welcomeSocket);

}



**UDP**

**Server**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#define MAXSIZE 100

bool checkBuff(char \*buff) {

return buff[0] == 'H' && buff[1] == 'a' && buff[2] == 'l' && buff[3] == 't';

}

int isPalindrome(char str[])

{

int l = 0;

int h = strlen(str) - 1;

while (h > l)

{

if (str[l++] != str[h--])

return 0;

}

return 1;

}

int main()

{

int sockfd, newsockfd, retval, i;

socklen\_t actuallen;

int recedbytes, sentbytes;

struct sockaddr\_in serveraddr, clientaddr;

char buff[MAXSIZE];

char output[MAXSIZE];

int a = 0;

sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

if (sockfd == -1)

{

printf("\nSocket creation error");

}

clientaddr.sin\_family = AF\_INET;

clientaddr.sin\_port = htons(3389);

clientaddr.sin\_addr.s\_addr = htons(INADDR\_ANY);

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_port = htons(3388);

serveraddr.sin\_addr.s\_addr = htons(INADDR\_ANY);

retval = bind(sockfd, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));

if (retval == 1)

{

printf("Binding error");

close(sockfd);

}

while (1)

{

actuallen = sizeof(clientaddr);

recedbytes = recvfrom(sockfd, buff, sizeof(buff), 0, (struct sockaddr \*)&clientaddr, &actuallen);

if (recedbytes == -1)

{

printf("Reciving error\n");

close(sockfd);

}

puts(buff);

printf("\n");

if ((buff[0] == 'H' || buff[0] == 'h') && buff[1] == 'a' && buff[2] == 'l' && buff[3] == 't')

{

break;

}

int check = isPalindrome(buff);

if (!check)

{

int i = 0;

char res[] = "Not a Palindrome";

strcpy(output, res);

}

else

{

int i = 0, c = 0, count = 0;

char res[] = "Is a Palindrome";

strcpy(output, res);

char len[MAXSIZE], countstr[MAXSIZE], space[] = " ";

sprintf(len, "%ld", strlen(buff));

strcat(output, " \nLength: ");

strcat(output, len);

puts(output);

while (buff[c] != '\0')

{

if (buff[c] == 'a' || buff[c] == 'A' || buff[c] == 'e' || buff[c] == 'E' || buff[c] == 'i' || buff[c] == 'I' || buff[c] == 'o' || buff[c] == 'O' || buff[c] == 'u' || buff[c] == 'U')

count++;

c++;

}

sprintf(countstr, "%d", count);

strcat(output, " \nVowels: ");

strcat(output, countstr);

}

sentbytes = sendto(sockfd, output, sizeof(buff), 0, (struct sockaddr \*)&clientaddr, sizeof(clientaddr));

if (checkBuff(buff)) {

break;

}

}

close(sockfd);

close(newsockfd);

}

**Client**

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <arpa/inet.h>

#define MAXSIZE 50

bool check(char \*buff) {

return buff[0] == 'H' && buff[1] == 'a' && buff[2] == 'l' && buff[3] == 't';

}

int main()

{

int sockfd, retval;

char buff[MAXSIZE];

int recedbytes, sentbytes;

struct sockaddr\_in serveraddr, clientaddr;

sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

if (sockfd == -1)

{

printf("\nSocket Creation Error");

return 0;

}

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_port = htons(3388);

serveraddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

clientaddr.sin\_family = AF\_INET;

clientaddr.sin\_port = htons(3389);

clientaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

retval = bind(sockfd, (struct sockaddr \*)&clientaddr, sizeof(clientaddr));

if (retval == 1)

{

printf("Binding error");

close(sockfd);

}

while (1)

{

printf("Enter a string: ");

scanf("%s", buff);

sentbytes = sendto(sockfd, buff, sizeof(buff), 0,

(struct sockaddr \*)&serveraddr, sizeof(serveraddr));

if (sentbytes == -1)

{

printf("sending error");

close(sockfd);

}

if (check(buff))

{

break;

}

socklen\_t size = sizeof(serveraddr);

recedbytes = recvfrom(sockfd, buff, sizeof(buff), 0, (struct sockaddr \*)&serveraddr, &size);

puts(buff);

printf("\n");

if (check(buff)) {

break;

}

}

close(sockfd);

}

**Lab 2**

File.txt

I am Devang Arora

I am 20 years old

I love to code

**TCP**

**Server**

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <stdio.h>

#define MAX\_SIZE 100

void replaceAll(char \*str, const char \*oldWord, const char \*newWord)

{

char \*pos, temp[1000];

int index = 0;

int owlen;

owlen = strlen(oldWord);

while ((pos = strstr(str, oldWord)) != NULL) {

strcpy(temp, str);

index = pos - str;

str[index] = '\0';

strcat(str, newWord);

strcat(str, temp + index + owlen);

}

}

int main()

{

int s, r, recb, sntb, ns, a = 0;

socklen\_t len;

struct sockaddr\_in server, client;

char buff[50];

s = socket(AF\_INET, SOCK\_STREAM, 0);

if (s == -1)

{

printf("\nSocket creation error.");

exit(0);

}

server.sin\_family = AF\_INET;

server.sin\_port = htons(5000);

server.sin\_addr.s\_addr = htonl(INADDR\_ANY);

r = bind(s, (struct sockaddr \*)&server, sizeof(server));

if (r == -1)

{

printf("\nBinding error.");

exit(0);

}

r = listen(s, 1);

if (r == -1)

{

close(s);

exit(0);

}

len = sizeof(client);

ns = accept(s, (struct sockaddr \*)&client, &len);

if (ns == -1)

{

close(s);

exit(0);

}

recb = recv(ns, buff, sizeof(buff), 0);

printf("\nFile recieved");

char fil[50];

if (access(buff, F\_OK) != -1) {

strcpy(fil, buff);

strcpy(buff, "File exists");

} else {

strcpy(buff, "File not present");

}

sntb = send(ns, buff, sizeof(buff), 0);

if (sntb == -1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

if (strcmp(buff, "File not present") == 0)

{

close(s);

close(ns);

exit(0);

}

int ch = 0;

while (ch != 4)

{

recb = recv(ns, buff, sizeof(buff), 0);

if (recb == -1)

{

printf("\nMessage Recieving Failed");

close(s);

close(ns);

exit(0);

}

ch = buff[0];

int i, n, n1, n2, j;

char str[MAX\_SIZE], str1[MAX\_SIZE], str2[MAX\_SIZE];

char strTempData[MAX\_SIZE];

char \*\*strData = NULL; // String List

int noOfLines = 0;

switch (ch)

{

case 1:

{ printf("\nSearch operation called\n");

n = buff[1];

for (i = 0; i < n; i++)

str[i] = buff[i + 2];

str[n] = '\0';

FILE \*fp;

int line\_num = 1;

int find\_result = 0;

char temp[512];

if ((fp = fopen(fil, "r")) == NULL)

{

printf("\nFile not found");

close(s);

close(ns);

exit(0);

}

while (fgets(temp, 512, fp) != NULL)

{

if ((strstr(temp, str)) != NULL)

{

find\_result++;

}

line\_num++;

}

if (fp)

{

fclose(fp);

}

buff[0] = find\_result;

sntb = send(ns, buff, sizeof(buff), 0);

if (sntb == -1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

break;

}

case 2:

{

n1 = buff[1];

i = 2;

for (j = 0; j < n1; j++)

{

str1[j] = buff[i];

i++;

}

str1[j] = '\0';

recb = recv(ns, buff, sizeof(buff), 0);

if (recb == -1)

{

printf("\nMessage Recieving Failed");

close(s);

close(ns);

exit(0);

}

n = buff[1];

i = 2;

for (j = 0; j < n; j++)

{

str2[j] = buff[i];

i++;

}

str2[j] = '\0';

printf("\nReplace called: %s -> %s\n", str1, str2);

FILE \*fPtr;

FILE \*fTemp;

char buffer[1000];

fPtr = fopen(fil, "r");

fTemp = fopen("replace.tmp", "w");

if (fPtr == NULL || fTemp == NULL)

{

printf("\nUnable to open file.\n");

printf("Please check whether file exists and you have read/write privilege.\n");

exit(0);

}

while ((fgets(buffer, 1000, fPtr)) != NULL)

{

replaceAll(buffer, str1, str2);

fputs(buffer, fTemp);

}

fclose(fPtr);

fclose(fTemp);

remove(fil);

rename("replace.tmp", fil);

strcpy(buff, "Replace done");

sntb = send(ns, buff, sizeof(buff), 0);

if (sntb == -1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

break;

}

case 3:

{

printf("\nOrdering operation called");

FILE \*ptrFileLog = NULL;

FILE \*ptrSummary = NULL;

if ((ptrFileLog = fopen(fil, "r")) == NULL)

{

fprintf(stderr, "Error: Could not open %s\n", fil);

return 1;

}

if ((ptrSummary = fopen("temp.txt", "a")) == NULL)

{

fprintf(stderr, "Error: Could not open temp.txt\n");

return 1;

}

while (fgets(strTempData, MAX\_SIZE, ptrFileLog) != NULL)

{

if (strchr(strTempData, '\n'))

strTempData[strlen(strTempData) - 1] = '\0';

strData = (char \*\*)realloc(strData, sizeof(char \*\*) \* (noOfLines + 1));

strData[noOfLines] = (char \*)calloc(MAX\_SIZE, sizeof(char));

strcpy(strData[noOfLines], strTempData);

noOfLines++;

}

for (i = 0; i < (noOfLines - 1); ++i)

{

for (j = 0; j < (noOfLines - i - 1); ++j)

{

if (strcmp(strData[j], strData[j + 1]) > 0)

{

strcpy(strTempData, strData[j]);

strcpy(strData[j], strData[j + 1]);

strcpy(strData[j + 1], strTempData);

}

}

}

for (i = 0; i < noOfLines; i++)

fprintf(ptrSummary, "%s\n", strData[i]);

for (i = 0; i < noOfLines; i++)

free(strData[i]);

free(strData);

remove(fil);

rename("temp.txt", fil);

fclose(ptrFileLog);

fclose(ptrSummary);

strcpy(buff, "Ordering done");

sntb = send(ns, buff, sizeof(buff), 0);

if (sntb == -1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

break;

}

case 4:

{

ch = 4;

break;

}

}

}

close(ns);

close(s);

}

**Client**

#include <string.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <fcntl.h>

#include <sys/stat.h>

int main()

{

int s, r, recb, data;

struct sockaddr\_in server;

char buff[50];

s = socket(AF\_INET, SOCK\_STREAM, 0);

if (s == -1)

{

printf("\nSocket creation error.");

exit(0);

}

server.sin\_family = AF\_INET;

server.sin\_port = htons(5000);

server.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

r = connect(s, (struct sockaddr \*)&server, sizeof(server));

if (r == -1)

{

printf("\nConnection error.");

exit(0);

}

printf("\n\n");

printf("Input File Name: ");

scanf("%s", buff);

data = send(s, buff, sizeof(buff), 0);

if (data == -1) {

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

recb = recv(s, buff, sizeof(buff), 0);

if (recb == -1) {

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("\n");

printf("%s", buff);

printf("\n\n");

if (strcmp(buff, "File not present!") == 0)

{

close(s);

exit(0);

}

int ch = 0;

printf("\n1.Search\n2.Replace\n3.Reorder\n4.Exit\nEnter your choice: ");

while (ch != 4)

{

scanf("%d", &ch);

buff[0] = ch;

char str1[50], str2[50];

int n, i, j;

switch (ch)

{

case 1:

printf("\nEnter string to be searched: ");

scanf("%s", str1);

n = strlen(str1);

buff[1] = n;

for (i = 0; i < n; i++)

buff[i + 2] = str1[i];

buff[i + 2] = '\0';

data = send(s, buff, sizeof(buff), 0);

if (data == -1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

recb = recv(s, buff, sizeof(buff), 0);

if (recb == -1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

n = buff[0];

printf("\n%s found %d number of times\n", str1, n);

break;

case 2:

printf("\nEnter string to be searched and replaced: ");

scanf("%s", str1);

n = strlen(str1);

buff[1] = n;

for (i = 0; i < n; i++)

buff[i + 2] = str1[i];

buff[i + 2] = '\0';

data = send(s, buff, sizeof(buff), 0);

if (data == -1)

{

close(s);

printf("Message Sending Failed");

exit(0);

}

printf("Enter new string: ");

scanf("%s", str2);

n = strlen(str2);

buff[1] = n;

i = 2;

for (j = 0; j < n; j++)

{

buff[i] = str2[j];

i++;

}

buff[i] = '\0';

data = send(s, buff, sizeof(buff), 0);

if (data == -1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

recb = recv(s, buff, sizeof(buff), 0);

if (recb == -1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("%s\n", buff);

break;

case 3:

data = send(s, buff, sizeof(buff), 0);

if (data == -1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

recb = recv(s, buff, sizeof(buff), 0);

if (recb == -1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("%s\n", buff);

break;

case 4:

data = send(s, buff, sizeof(buff), 0);

if (data == -1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

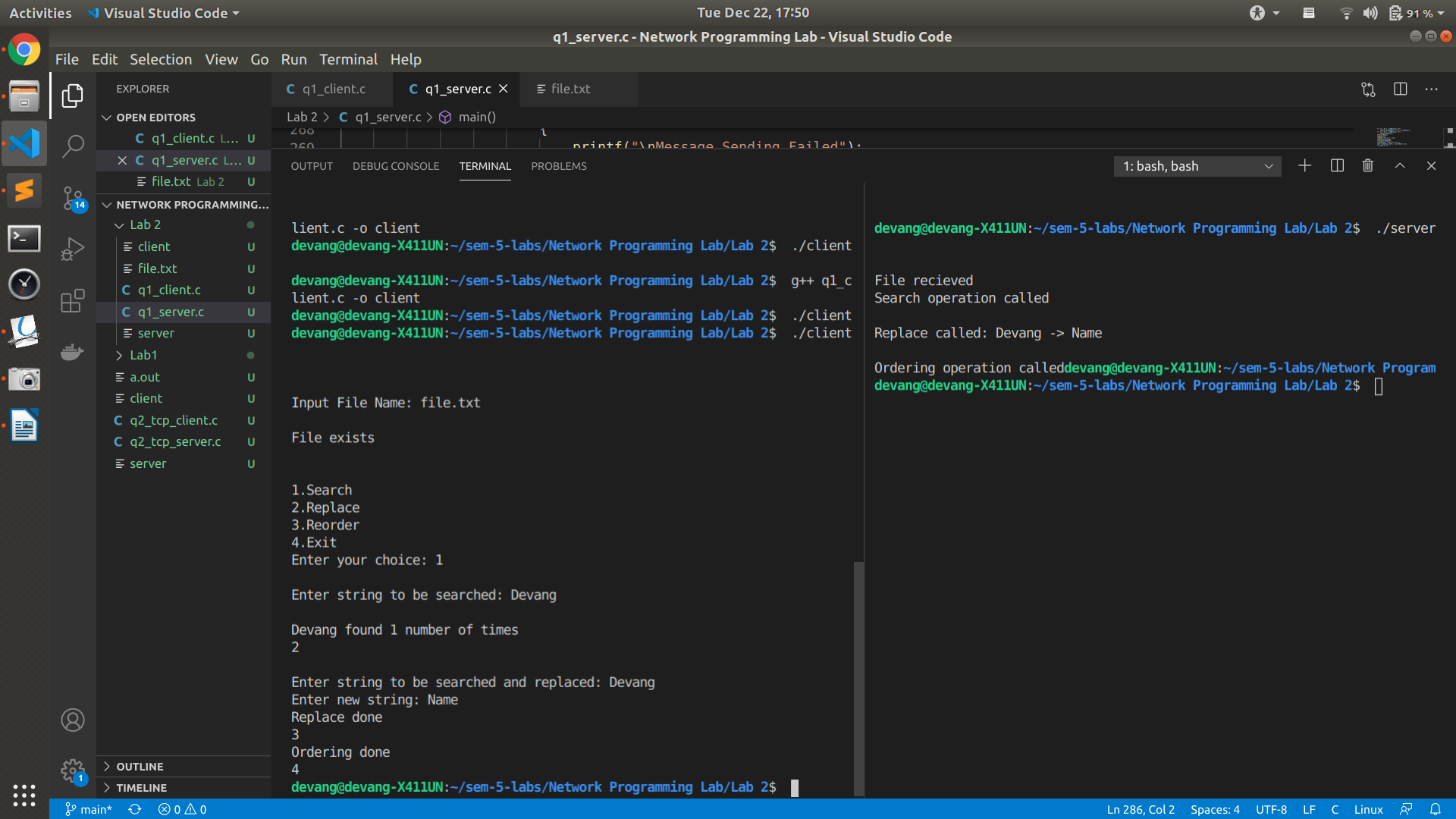
break;

}

}

close(s);

}



File.txt after ordering

I am 20 years old

I am Name Arora

I love to code

**UDP**

**File.txt**

I am Devang Arora

I am 20 years old

I love to code

**Server**

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <stdio.h>

#define MAX\_LEN 100

void replaceAll(char \*str, const char \*oldWord, const char \*newWord)

{

char \*pos, temp[1000];

int index = 0;

int owlen;

owlen = strlen(oldWord);

while ((pos = strstr(str, oldWord)) != NULL) {

strcpy(temp, str);

index = pos - str;

str[index] = '\0';

strcat(str, newWord);

strcat(str, temp + index + owlen);

}

}

int main()

{

int s, r, recData, sentData;

socklen\_t ca;

socklen\_t len;

struct sockaddr\_in server, client;

char buff[50];

s = socket(AF\_INET, SOCK\_DGRAM, 0);

server.sin\_family = AF\_INET;

server.sin\_port = htons(5000);

server.sin\_addr.s\_addr = htonl(INADDR\_ANY);

len = sizeof(client);

ca = sizeof(client);

r = bind(s, (struct sockaddr \*)&server, sizeof(server));

recData = recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr \*)&client, &ca);

printf("\nFile Name Recieved!\n");

char fil[50];

if (access(buff, F\_OK) != -1)

{

strcpy(fil, buff);

strcpy(buff, "File exists");

}

else

{

strcpy(buff, "File does not exist!");

}

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&client, len);

if (sentData == -1)

{

printf("\nMessage Sending Failed");

close(s);

exit(0);

}

if (strcmp(buff, "File does not exist!") == 0)

{

close(s);

exit(0);

}

int ch = 0;

while (ch != 4)

{

recData = recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr \*)&client, &ca);

ch = buff[0];

int i, n, n1, n2, j;

char str[50], str1[50], str2[50];

char strTempData[MAX\_LEN];

char \*\*strData = NULL;

int noOfLines = 0;

switch (ch)

{

case 1:

{

printf("\nSearching operation");

n = buff[1];

for (i = 0; i < n; i++)

str[i] = buff[i + 2];

str[n] = '\0';

FILE \*fp;

int line\_num = 1;

int find\_result = 0;

char temp[512];

if ((fp = fopen(fil, "r")) == NULL)

{

printf("\nFile not found");

close(s);

exit(0);

}

while (fgets(temp, 512, fp) != NULL)

{

if ((strstr(temp, str)) != NULL)

{

find\_result++;

}

line\_num++;

}

if (fp)

{

fclose(fp);

}

buff[0] = find\_result;

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&client, len);

break;

}

case 2:

{

n1 = buff[1];

i = 2;

for (j = 0; j < n1; j++) {

str1[j] = buff[i];

i++;

}

str1[j] = '\0';

recData = recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr \*)&client, &ca);

n = buff[1];

i = 2;

for (j = 0; j < n; j++)

{

str2[j] = buff[i];

i++;

}

str2[j] = '\0';

printf("\nReplace operation: %s -> %s..\n", str1, str2);

FILE \*fPtr;

FILE \*fTemp;

char buffer[1000];

fPtr = fopen(fil, "r");

fTemp = fopen("replace.tmp", "w");

while ((fgets(buffer, 1000, fPtr)) != NULL)

{

replaceAll(buffer, str1, str2);

fputs(buffer, fTemp);

}

fclose(fPtr);

fclose(fTemp);

remove(fil);

rename("replace.tmp", fil);

strcpy(buff, "Replace done");

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&client, len);

break;

}

case 3:

{

printf("\nOrdering operation\n");

FILE \*ptrFileLog = NULL;

FILE \*ptrSummary = NULL;

if ((ptrFileLog = fopen(fil, "r")) == NULL)

{

fprintf(stderr, "Error: Could not open %s\n", fil);

return 1;

}

if ((ptrSummary = fopen("temp.txt", "a")) == NULL)

{

fprintf(stderr, "Error: Could not open temp.txt\n");

return 1;

}

while (fgets(strTempData, MAX\_LEN, ptrFileLog) != NULL)

{

if (strchr(strTempData, '\n'))

strTempData[strlen(strTempData) - 1] = '\0';

strData = (char \*\*)realloc(strData, sizeof(char \*\*) \* (noOfLines + 1));

strData[noOfLines] = (char \*)calloc(MAX\_LEN, sizeof(char));

strcpy(strData[noOfLines], strTempData);

noOfLines++;

}

for (i = 0; i < (noOfLines - 1); ++i)

{

for (j = 0; j < (noOfLines - i - 1); ++j)

{

if (strcmp(strData[j], strData[j + 1]) > 0)

{

strcpy(strTempData, strData[j]);

strcpy(strData[j], strData[j + 1]);

strcpy(strData[j + 1], strTempData);

}

}

}

for (i = 0; i < noOfLines; i++)

fprintf(ptrSummary, "%s\n", strData[i]);

for (i = 0; i < noOfLines; i++)

free(strData[i]);

free(strData);

remove(fil);

rename("temp.txt", fil);

fclose(ptrFileLog);

fclose(ptrSummary);

strcpy(buff, "Ordering done");

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&client, len);

if (sentData == -1)

{

printf("\nMessage Sending Failed");

close(s);

exit(0);

}

break;

}

case 4:

ch = 4;

break;

}

}

close(s);

}

**Client**

#include <string.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <fcntl.h>

#include <sys/stat.h>

int main()

{

int s, r, recData, sentData, x;

socklen\_t sa;

socklen\_t len;

struct sockaddr\_in server, client;

char buff[50];

s = socket(AF\_INET, SOCK\_DGRAM, 0);

if (s == -1)

{

printf("\nSocket creation error.");

exit(0);

}

server.sin\_family = AF\_INET;

server.sin\_port = htons(5000);

server.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

sa = sizeof(server);

len = sizeof(server);

printf("\n\n");

printf("Type File Name: ");

scanf("%s", buff);

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, len);

if (sentData == -1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

recData = recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, &sa);

if (recData == -1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("\n");

printf("%s", buff);

printf("\n\n");

if (strcmp(buff, "File does not exist!") == 0)

{

close(s);

exit(0);

}

int ch = 0;

printf("\n1.Search\n2.Replace\n3.Reorder\n4.Exit\n");

while (ch != 4)

{

printf("\nEnter choice: ");

scanf("%d", &ch);

buff[0] = ch;

char str1[50], str2[50];

int n, i, j;

switch (ch)

{

case 1:

{

printf("\nEnter string to be searched: ");

scanf("%s", str1);

n = strlen(str1);

buff[1] = n;

for (i = 0; i < n; i++)

buff[i + 2] = str1[i];

buff[i + 2] = '\0';

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, len);

if (sentData == -1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

recData = recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, &sa);

if (recData == -1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

n = buff[0];

printf("\nWord found %d number of times!\n", n);

break;

}

case 2:

{

printf("\nEnter string to be searched and replaced: ");

scanf("%s", str1);

n = strlen(str1);

buff[1] = n;

for (i = 0; i < n; i++)

buff[i + 2] = str1[i];

buff[i + 2] = '\0';

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, len);

if (sentData == -1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

printf("\nEnter new string: ");

scanf("%s", str2);

n = strlen(str2);

buff[1] = n;

i = 2;

for (j = 0; j < n; j++)

{

buff[i] = str2[j];

i++;

}

buff[i] = '\0';

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, len);

if (sentData == -1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

recData = recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, &sa);

if (recData == -1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("%s\n", buff);

break;

}

case 3:

{

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, len);

if (sentData == -1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

recData = recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, &sa);

if (recData == -1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("%s\n", buff);

break;

case 4:

sentData = sendto(s, buff, sizeof(buff), 0, (struct sockaddr \*)&server, len);

if (sentData == -1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

break;

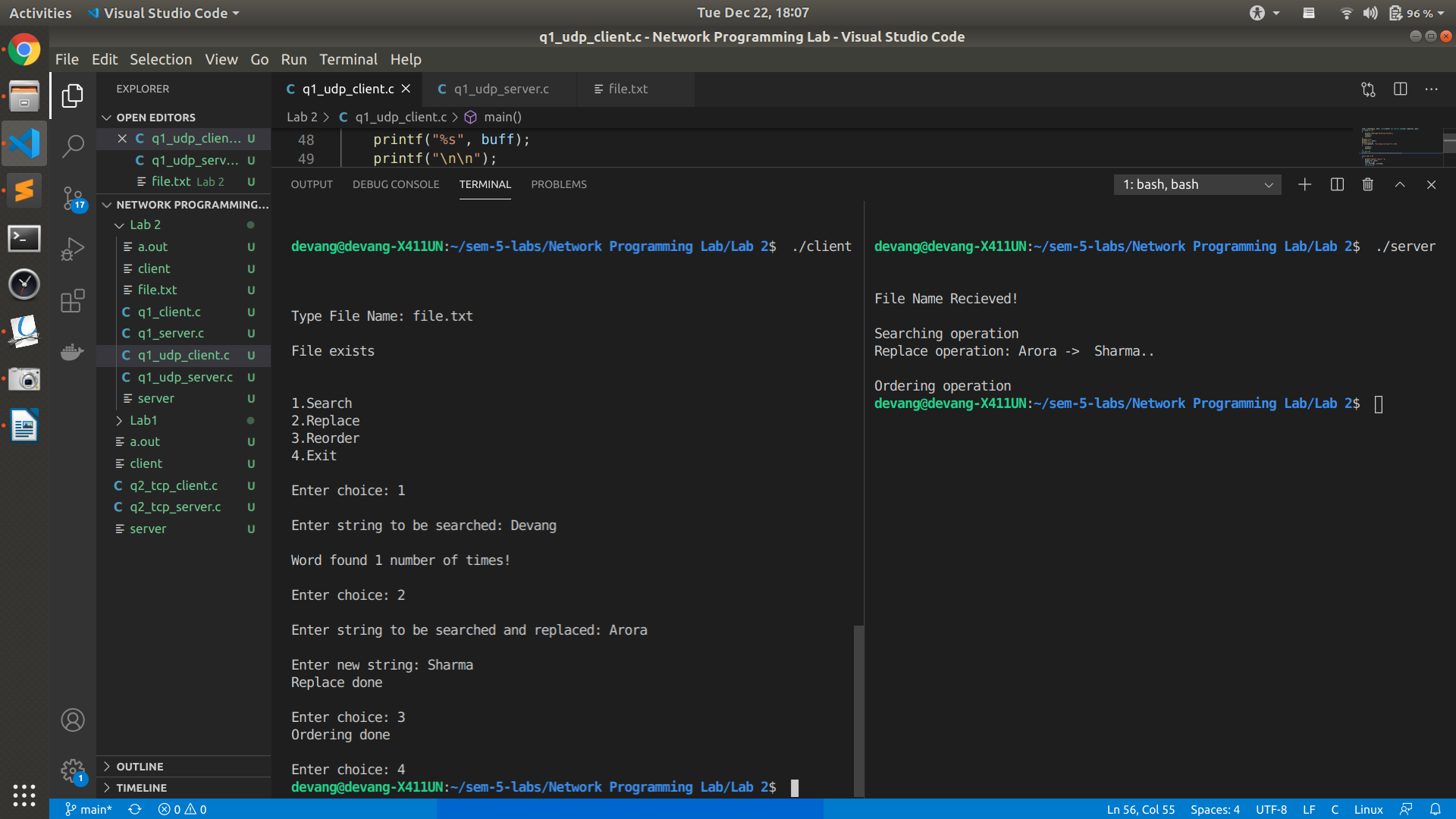
}

}

}

close(s);

}

****

**File.txt after operations**

I am 20 years old

I am Devang Sharma

I love to code

**Lab 3**

**Q1**

**Server**

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <stdio.h>

int main()

{

int s,r,recb,sntb,x,ns,a=0;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50],buff2[50];

s=socket(AF\_INET,SOCK\_STREAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(5001);

server.sin\_addr.s\_addr=htonl(INADDR\_ANY);

r=bind(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1)

{

printf("\nBinding error.");

exit(0);

}

printf("\nSocket binded.");

r=listen(s,1);

if(r==-1)

{

close(s);

exit(0);

}

printf("\nSocket listening.");

len=sizeof(client);

ns=accept(s,(struct sockaddr\*)&client, &len);

if(ns==-1)

{

close(s);

exit(0);

}

printf("\nSocket accepting.\n");

int pid=fork();

while(1){

if(pid>0){

printf("\n\nParent - Type message: ");

scanf("%s", buff2);

printf("\nParent's PID: %d", pid);

sntb=send(ns,buff2,sizeof(buff2),0);

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

if(strcmp(buff2,"BYE")==0)

break;

}

else{

printf("\nChild's PID: %d", pid);

recb=recv(ns,buff,sizeof(buff),0);

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

close(ns);

exit(0);

}

if(strcmp(buff,"BYE")==0)

break;

printf("\nChild - Message Recieved: ");

printf("%s\n", buff);

}

}

close(ns);

close(s);

}

**Client**

#include <string.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <fcntl.h>

#include <sys/stat.h>

int main()

{

int s,r,recb,sntb,x;

struct sockaddr\_in server;

char buff[50],buff2[50];

s=socket(AF\_INET,SOCK\_STREAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(5001);

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

r=connect(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1)

{

printf("\nConnection error.");

exit(0);

}

printf("\nSocket connected.");

printf("\n\n");

int pid;

pid=fork();

while(1){

if(pid>0)

{

recb=recv(s,buff,sizeof(buff),0);

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

if(strcmp(buff,"BYE")==0)

break;

printf("\nParent - Message Recieved: ");

printf("%s\n", buff);

printf("\nParent's PID: %d", pid);

}

else

{

printf("\nChild - Type Message: ");

scanf("%s", buff2);

sntb=send(s,buff2,sizeof(buff2),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

if(strcmp(buff2,"BYE")==0)

break;

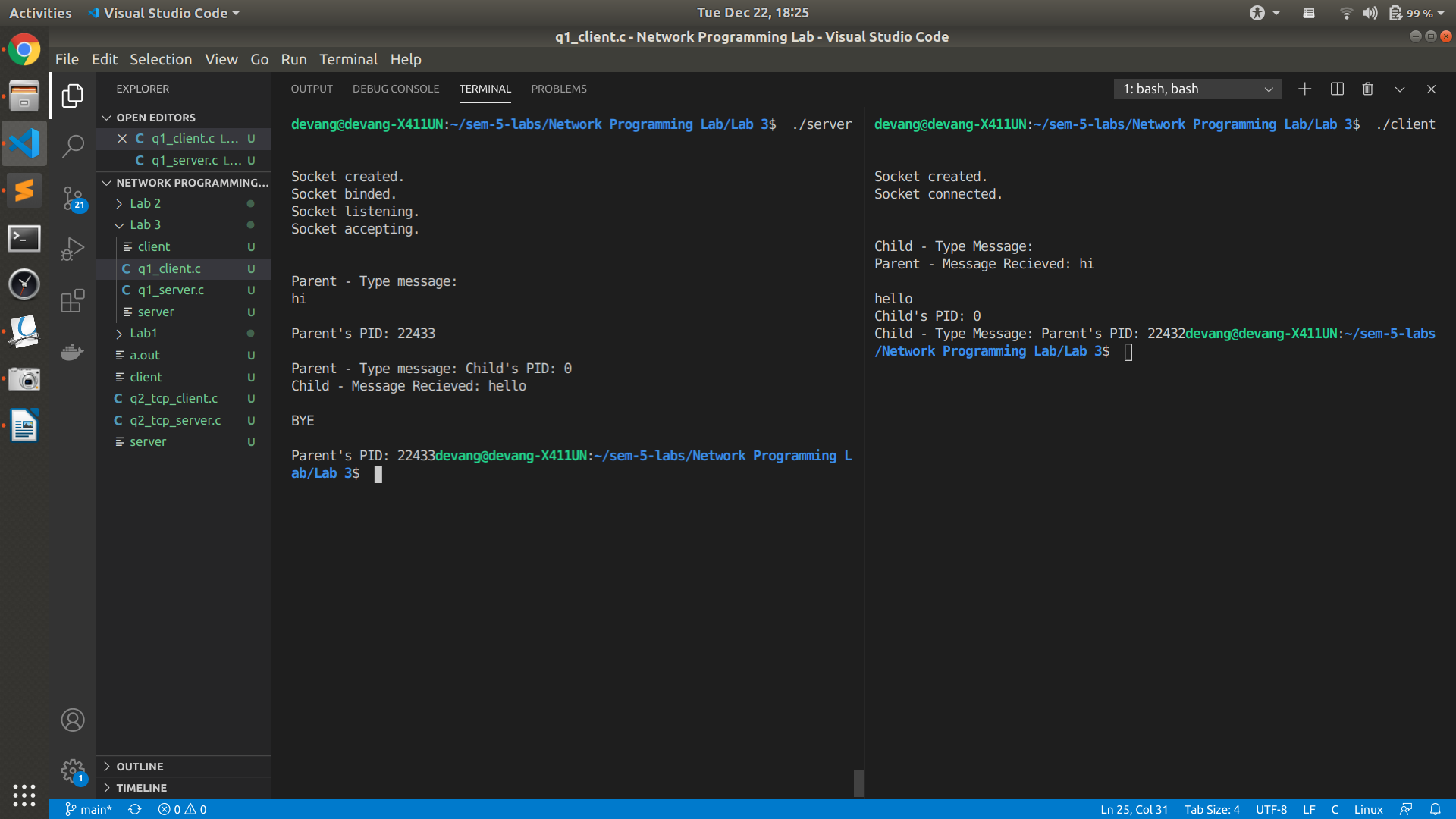
printf("Child's PID: %d", pid);

}

}

close(s);

}

****

**Q2**

**Server**

#include<string.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdlib.h>

#include<stdio.h>

void swap(char \*x, char \*y)

{

char temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

void permute(char \*a, int l, int r)

{

int i;

if (l == r)

printf("%s\n", a);

else

{

for (i = l; i <= r; i++)

{

swap((a+l), (a+i));

permute(a, l+1, r);

swap((a+l), (a+i)); //backtrack

}

}

}

int main()

{

int s,r,recb,sntb,x;

socklen\_t ca;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

s=socket(AF\_INET,SOCK\_DGRAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.\n");

server.sin\_family=AF\_INET;

server.sin\_port=htons(5000);

server.sin\_addr.s\_addr=htonl(INADDR\_ANY);

len=sizeof(client);

ca=sizeof(client);

r=bind(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1)

{

printf("\nBinding error.");

exit(0);

}

printf("\nSocket binded.");

while(1){

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,&ca);

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

if(!strcmp(buff,"stop"))

break;

printf("\nPermutations of the %s are: \n", buff);

int n=strlen(buff);

permute(buff, 0, n-1);

}

close(s);

}

**Client**

#include<string.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<fcntl.h>

#include<sys/stat.h>

int main()

{

int s,r,recb,sntb,x;

int sa;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

s=socket(AF\_INET,SOCK\_DGRAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(5000);

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

sa=sizeof(server);

len=sizeof(server);

while(1){

printf("\n\n");

printf("Type Message: ");

scanf("%s", buff);

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

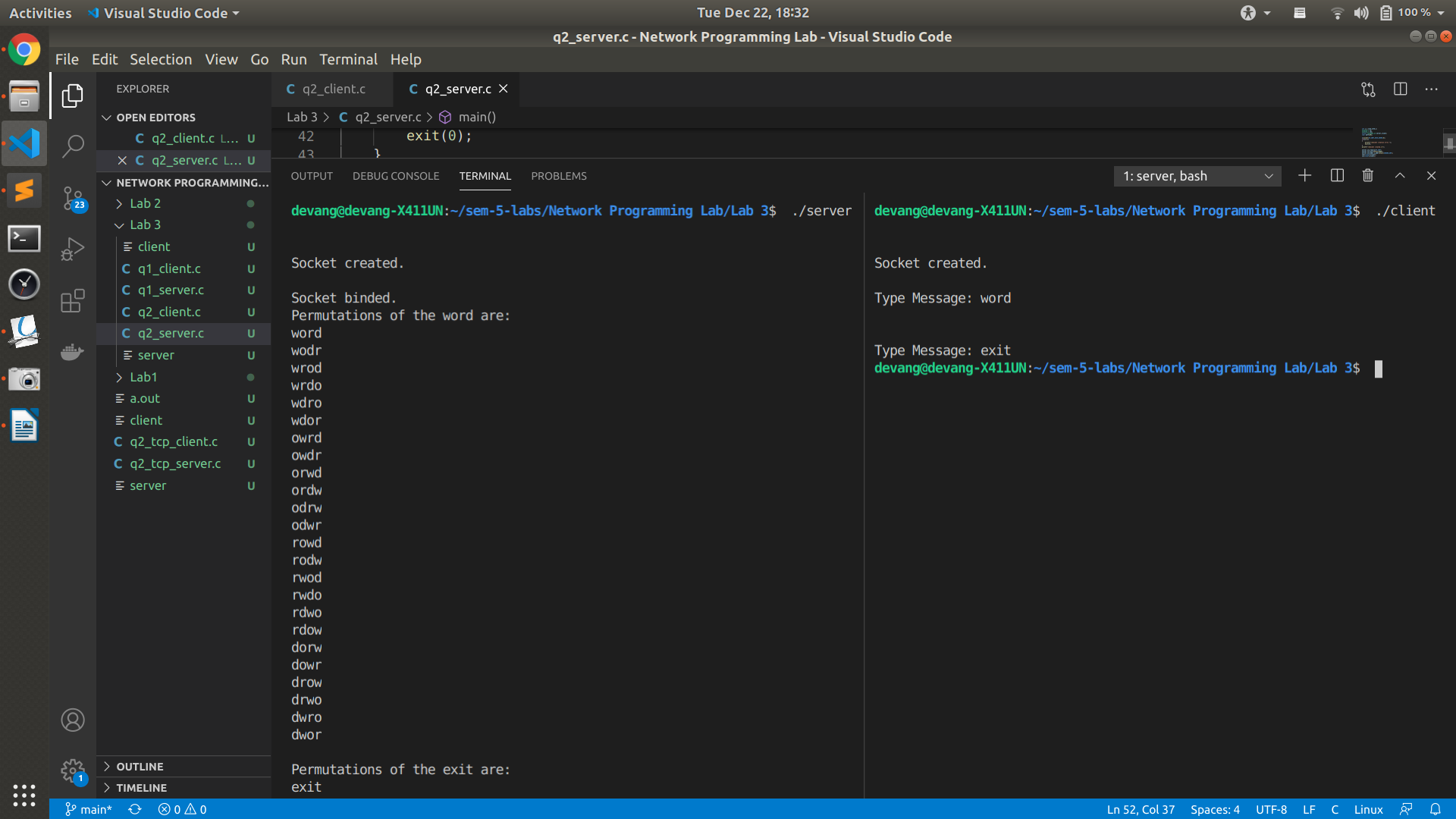
if(!strcmp(buff,"exit"))

break;

}

close(s);

}

****

**Q3**

**Server**

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <stdio.h>

#define PORT 8000

int main()

{

int s, r, recb, sntb, x, ns, a = 0;

socklen\_t len;

struct sockaddr\_in server, client;

char buff[50], buff2[50];

s = socket(AF\_INET, SOCK\_STREAM, 0);

if (s == -1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

//port number for communication

//socket creation

server.sin\_family = AF\_INET;

server.sin\_port = htons(PORT);

server.sin\_addr.s\_addr = htonl(INADDR\_ANY);

r = bind(s, (struct sockaddr \*)&server, sizeof(server));

if (r == -1)

{

//socket binding

printf("\nBinding error.");

exit(0);

}

printf("\nSocket binded.");

r = listen(s, 1);

if (r == -1)

{

close(s);

exit(0);

}

printf("\nSocket listening.");

len = sizeof(client);

ns = accept(s, (struct sockaddr \*)&client, &len);

//socket accepting

if (ns == -1)

{

close(s);

exit(0);

}

printf("\nSocket accepting.\n");

recb = recv(ns, buff, sizeof(buff), 0);

if (recb == -1)

{

printf("\nMessage receiving failed.");

close(s);

close(ns);

exit(0);

}

printf("\nMessage Received: ");

printf("%s\n", buff);

int pid = fork();

if (pid > 0)

{

//parent process

char num[50] = {0}, alpha[50] = {0};

int i, j, k;

i = 0;

j = 0;

k = 0;

int n = strlen(buff);

for (i = 0; i < n; i++)

{

if ((buff[i] >= 65 && buff[i] <= 91) || (buff[i] >= 97 && buff[i] <= 123))

{

alpha[j] = buff[i];

j++;

}

else

{

num[k] = buff[i];

k++;

}

}

alpha[j] = '\0';

num[k] = '\0';

for (i = 0; i < j - 1; i++)

{

for (int x = 0; x < j - i - 1; x++)

{

if (alpha[x] < alpha[x + 1])

{

char temp = alpha[x];

alpha[x] = alpha[x + 1];

alpha[x + 1] = temp;

}

}

}

printf("\nSorted in descending order: %s \nby parent process with PID: %d:", alpha, pid);

printf("\n\n");

}

else

{

char num[50] = {0}, alpha[50] = {0};

int i, j, k;

i = 0;

j = 0;

k = 0;

int n = strlen(buff);

for (i = 0; i < n; i++)

{

if ((buff[i] >= 65 && buff[i] <= 91) || (buff[i] >= 97 && buff[i] <= 123))

{

alpha[j] = buff[i];

j++;

}

else

{

num[k] = buff[i];

k++;

}

}

alpha[j] = '\0';

num[k] = '\0';

for (i = 0; i < k - 1; i++)

{

for (int x = 0; x < k - i - 1; x++)

{

if (num[x] > num[x + 1])

{

char temp = num[x];

num[x] = num[x + 1];

num[x + 1] = temp;

}

}

}

printf("\nSorted in ascending order: %s \nby child process with PID %d:", num, pid);

}

close(ns);

close(s);

}

**Client**

#include <string.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#define PORT 8000

int main()

{

int s,r,recb,sntb,x;

struct sockaddr\_in server;

char buff[50];

s=socket(AF\_INET,SOCK\_STREAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(PORT);

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

r=connect(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1)

{

printf("\nConnection error.");

exit(0);

}

printf("\nSocket connected.\n");

printf("\n\n");

strcpy(buff,"");

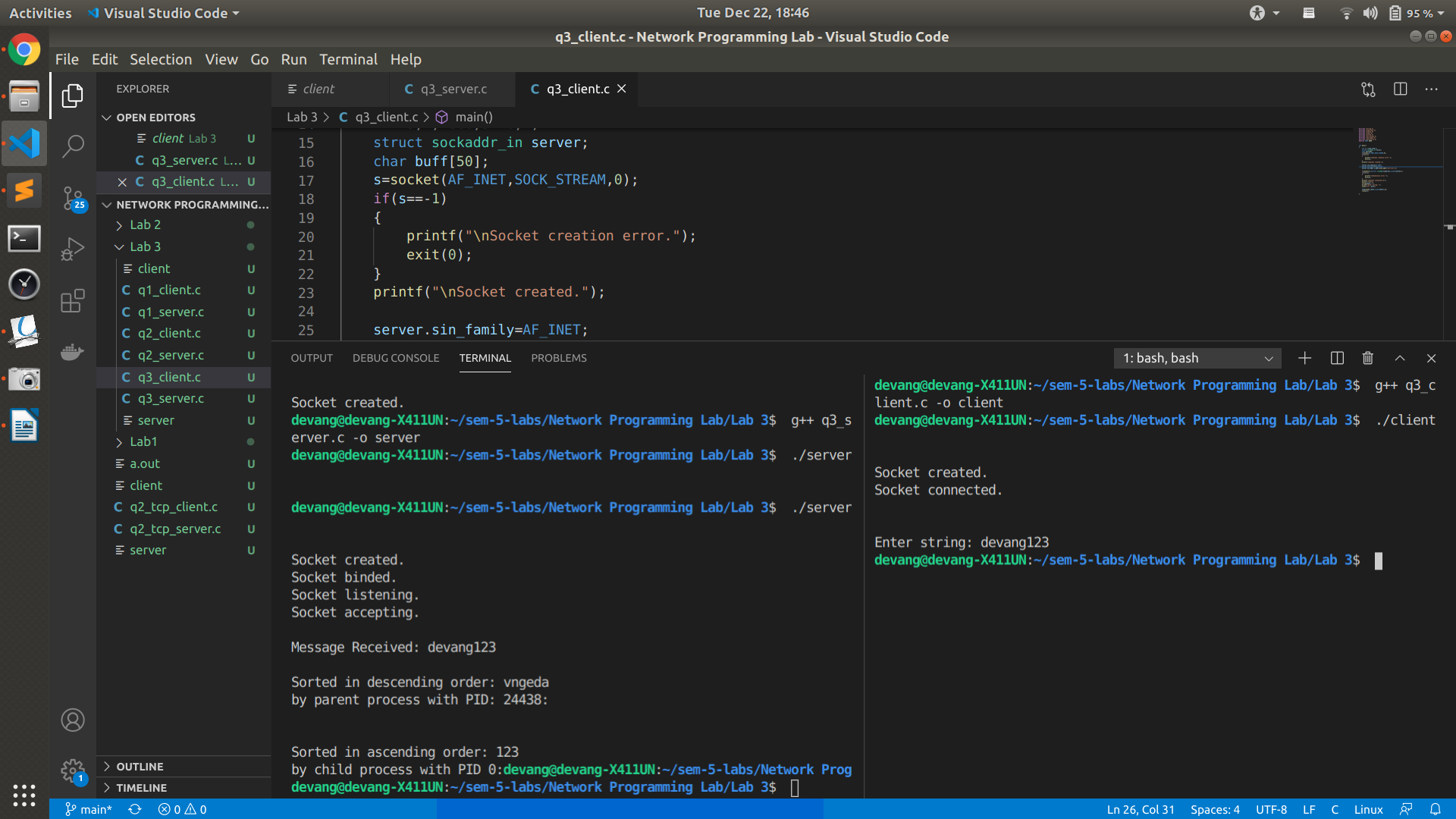
printf("Enter string: ");

scanf("%s", buff);

sntb=send(s,buff,sizeof(buff),0);

close(s);

}

****