

**Algorithm Guess Game using Socket Programming**

**Team Members :**

**RA1911003011021 – T.V.Sai Krishna**

**RA1911003011043 – Ramnath**

**RA1911003011045 – Hamesh Phani Sai**

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| ***S.No*** | ***CONTENTS*** |
| *1.* | *INTRODUCTION* |
| *2.* | *PROJECT SCOPE* |
| *3.* | *NETWORK REQUIREMENT ANALYSIS* |
| *4.* | *NETWORK DIAGRAM & COMPONENTS* |
| *5.* | *IMPLEMENTATION* |
| *6.* | *FEATURES AND SERVICES* |
| *7.* | *CONCLUSION* |

***INTRODUCTION***

In this game we will show the different algorithms to the user like 0/1 knapsack, LCS…based on the interest, the client will request the server. Then the server will send the input values to the client, hence the client will send the answer to the server. The server will tell to the client whether the client had sent a correct value or wrong value. And the server will store all the instances of the clients for future purposes.

Basically, we will solve many algorithms and the end will be by getting final result. But we don’t know whether the value we got is correct or not… so to solve this we innovate this project.

***PROJECT SCOPE***

The Algorithm guess game is a socket-based application. This game gives the users two options of algorithms to play with. When the user chooses either of the options the game proceeds accordingly. The game system has a system database that stores the scorestreaks of the players along with other details.

When the user chooses any game option the system sends some sets of values to the user and asks him/her to solve accordingly. When the user returns the answer the system crosschecks it and if the answer is correct then the user gets a point for the correct answer.

After answering each question the user is asked by the system whether he/she wants to continue or wants to quit the game. If the user chooses to quit then the game ends and the user gets a reflection of his/her scores and lists of correct and incorrect answers which he returned to the system while playing the game.

***NETWORK REQUIREMENT ANALYSIS***

***Hardware Requirements***

Processor : i3 processor or higher Clock Speed RAM : 512GB (minimum) Hard Disk : 2GB

***Software Requirements***

Operating System : Windows 7 Platform : C

Back End : File system

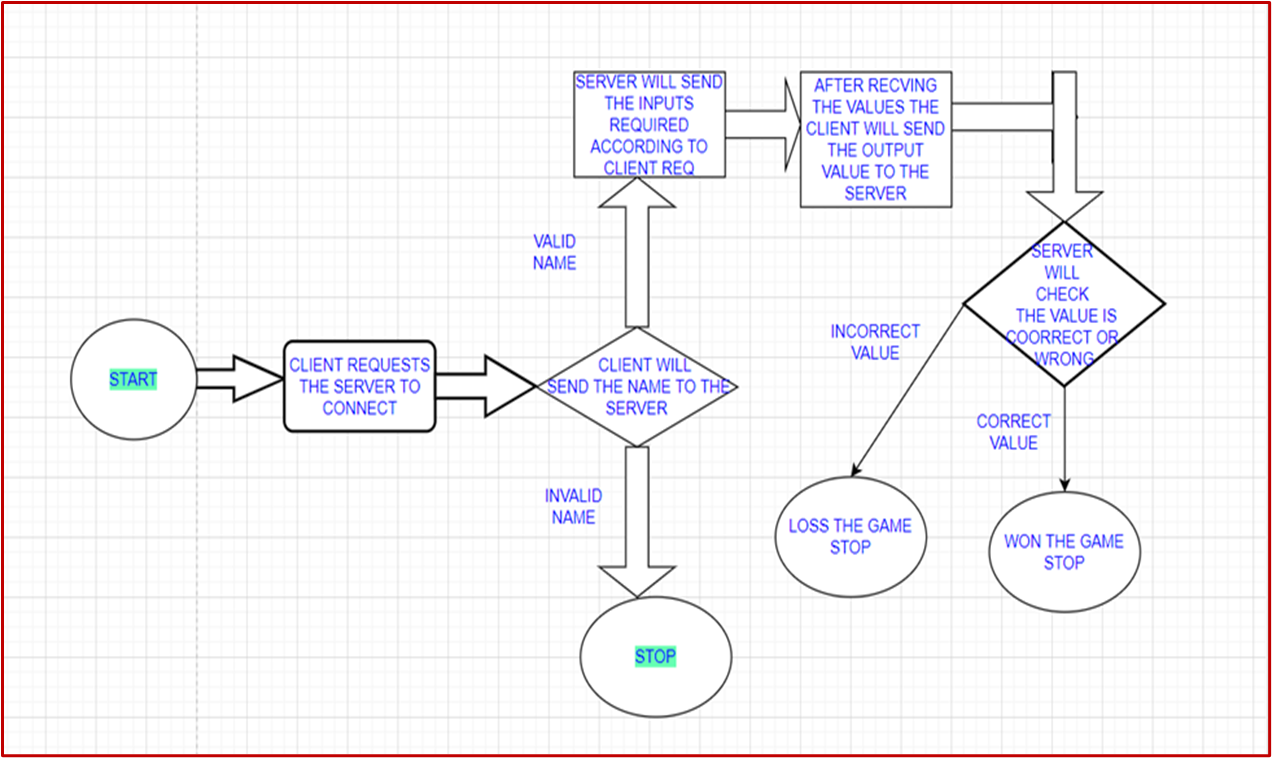
Special Tools :Socket Programming Server :TCP

***NETWORK DIAGRAM & COMPONENTS***

Basically, we will solve so many algorithms and we will end by getting final result but we don’t know whether the value we got is correct or not…to solve this we innovate this project. But implementing it using client-server program along with a Graphical User Interface would be tough.

So to keep things simple, we tried to display the inputs of the algorithm and final output on the screen without GUI assistance by adding some code snippet to source code.

Also we are storing all the instances of the clients like what algorithm he played, what are the inputs generated by the server, server calculates output value and client input.

**

# IMPLEMENTATION

This is a multiplayer strategy game on the linux or any compatible platform.There is so much scope for this project in the future with making few changes and modifications.

Some of them are: By adding some more details of the players, Making themes by integrating GUI to the project and background music can be added as well.

The server is connected to the client whenever the client requests. Then the client will send the name of the algorithm which he wants to play based on the given names of the algorithm.

After getting request from the client, the server will generate values according to the user request and send it back to the client. Server will automatically calculate the output and will store it. Whenever the client sends the output value, the server will check whether the output is correct or wrong and the result will be sent back to the client.

*Server will be saving all the responses of each and every client in a different* file for further uses

***FEATURES AND SERVICES***

**Cloud9 Services of AWS**

**SERVER CODE :**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

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

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

**#include <netinet/in.h>**

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

**#include<unistd.h>**

**float knapsack(int weight[], int profit[]) {**

**float x[20], tp = 0;**

**int i, j, u;**

**u = 10;**

**int n=4;**

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

**x[i] = 0.0;**

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

**if (weight[i] > u)**

**break;**

**else {**

**x[i] = 1.0;**

**tp = tp + profit[i];**

**u = u - weight[i];**

**}**

**}**

**if (i < n)**

**x[i] = u / weight[i];**

**tp = tp + (x[i] \* profit[i]);**

**return tp;**

**}**

**int m2=10,n2=6;**

**int max(int a, int b)**

**{**

**return (a > b)? a : b;**

**}**

**int lcs( char \*X, char \*Y, int m, int n )**

**{**

**if (m == 0 || n == 0)**

**return 0;**

**if (X[m-1] == Y[n-1])**

**return 1 + lcs(X, Y, m-1, n-1);**

**else**

**return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));**

**}**

**int main(int agrc,char \*argv[]){**

**int sockfd;int var,var1;int s1,s2;**

**char a[]="abcdefghijklmnopqrstuvwxyz";**

**FILE \*c1,\*c2;**

**c1=fopen("/home/ec2-user/environment/Project/client1","w+");**

**c2=fopen("/home/ec2-user/environment/Project/client2","w+");**

**int z=sizeof(a)-1;**

**struct sockaddr\_in serverAddr;int var3;int score1=0,score2=0;**

**int g1[4],g2[4];char g3[10],n[6],g4[10],n8[6];int g5[4];int g6[4];**

**int l,y,var2,l2;**

**int newSocket,newSocket2,x;**

**struct sockaddr\_in newAddr;**

**int ans1,ans2;**

**float profit=knapsack(g1,g2);**

**socklen\_t addr\_size;**

**char buffer[1024];**

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

**memset(&serverAddr, '\0', sizeof(serverAddr));**

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

**serverAddr.sin\_port = htons(atoi(argv[1]));**

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

**bind(sockfd, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));**

**listen(sockfd, 5);**

**int g=fork();**

**if(g==0)**

**{**

**newSocket = accept(sockfd, NULL, NULL);**

**printf("Connected with client socket number 1\n");**

**while(1)**

**{**

**recv(newSocket,&s1,sizeof(s1),0);**

**if(s1==0)**

**{**

**send(newSocket,&score1,sizeof(score1),0);**

**break;**

**}**

**else if(s1==1)**

**{**

**printf("client1 requested fractional knapsack\n");**

**for(int i=0;i<4;i++)**

**{**

**g1[i]=rand()%10;**

**if(g1[i]==0)**

**g1[i]=g1[i]+1;**

**}**

**for(int i=0;i<4;i++)**

**{**

**g2[i]=rand()%10;**

**if(g2[i]==0)**

**g2[i]=g2[i]+2;**

**}**

**send(newSocket, &g1, sizeof(g1), 0);**

**send(newSocket, &g2, sizeof(g2), 0);**

**float profit=knapsack(g1,g2);**

**int a1=(int)profit;**

**recv(newSocket,&ans1,sizeof(ans1),0);**

**send(newSocket,&a1,sizeof(a1),0);**

**fprintf(c1,"client1 requested knapsack\n");**

**fprintf(c1,"Weights :");**

**for(int i=0;i<4;i++)**

**{**

**fprintf(c1,"%d ",g1[i]);**

**}**

**fprintf(c1,"\nProfits :");**

**for(int i=0;i<4;i++)**

**{**

**fprintf(c1,"%d ",g2[i]);**

**}**

**fprintf(c1,"\nMaximum profit calcualted by server:%d\n",a1);**

**fprintf(c1,"Maximum profit calcualted by client:%d\n",ans1);**

**if(ans1==a1)**

**{**

**score1=score1+1;**

**fprintf(c1,"status:Client won the game\n");**

**}**

**else**

**fprintf(c1,"status:Client Loss the game\n");**

**}**

**else**

**{**

**printf("client1 requested LCS\n");**

**for(int i=0;i<10;i++)**

**{**

**y=rand()%z;**

**g3[i]=a[y];**

**}**

**for(int i=0;i<7;i++)**

**{**

**y=rand()%z;**

**n[i]=a[y];**

**}**

**send(newSocket,&g3,sizeof(g3),0);**

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

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

**int ms1=lcs(g3,n,m2,n2);**

**send(newSocket,&ms1,sizeof(ms1),0);**

**fprintf(c1,"client1 requested LCS\n");**

**fprintf(c1,"String1 :");**

**for(int i=0;i<10;i++)**

**{**

**fprintf(c1,"%c",g3[i]);**

**}**

**fprintf(c1,"\nString2 :");**

**for(int i=0;i<6;i++)**

**{**

**fprintf(c1,"%c",n[i]);**

**}**

**fprintf(c1,"\nServer calculated LCS value is %d\n",ms1);**

**fprintf(c1,"Client calcualted LCS value is %d\n",l);**

**if(ms1==l)**

**{**

**score1=score1+1;**

**fprintf(c1,"Status:Client won the game\n");**

**}**

**else**

**{**

**fprintf(c1,"Status:Client loss the game\n");**

**}**

**}**

**}**

**fprintf(c1,"Total Score is %d",score1);**

**printf("Score sent to Client1 Successfully and client1 exited\n");**

**}**

**else**

**{**

**newSocket2 = accept(sockfd, NULL, NULL);**

**printf("Connected with client socket number 2\n");**

**while(1)**

**{**

**recv(newSocket2,&s2,sizeof(s2),0);**

**if(s2==0)**

**{**

**send(newSocket2,&score2,sizeof(score2),0);**

**break;**

**}**

**else if(s2==1)**

**{**

**printf("client2 requested fractional knapsack\n");**

**for(int i=0;i<4;i++)**

**{**

**g5[i]=rand()%10;**

**if(g5[i]==0)**

**g5[i]=g5[i]+2;**

**}**

**for(int i=0;i<4;i++)**

**{**

**g6[i]=rand()%10;**

**if(g6[i]==0)**

**g6[i]=g6[i]+4;**

**}**

**send(newSocket2, &g5, sizeof(g5), 0);**

**send(newSocket2, &g6, sizeof(g6), 0);**

**recv(newSocket2,&ans2,sizeof(ans2),0);**

**float profit2=knapsack(g5,g6);**

**int a2=(int)profit2;**

**send(newSocket2,&a2,sizeof(a2),0);**

**fprintf(c2,"Client2 selected Knapsack\n");**

**fprintf(c2,"Weights :");**

**for(int i=0;i<4;i++)**

**{**

**fprintf(c2,"%d ",g5[i]);**

**}**

**fprintf(c2,"\nProfits :");**

**for(int i=0;i<4;i++)**

**{**

**fprintf(c2,"%d ",g6[i]);**

**}**

**fprintf(c2,"\nMaximum profit calcualted by the Server is %d\n",a2);**

**fprintf(c2,"Maximum profit calcuated by the client is %d\n",ans2);**

**if(a2==ans2)**

**{**

**score2=score2+1;**

**fprintf(c2,"Status:Client won the game\n");**

**}**

**else**

**{**

**fprintf(c2,"Status:Client Loss the game\n");**

**}**

**}**

**else**

**{**

**printf("client2 requested LCS\n");**

**for(int i=0;i<10;i++)**

**{**

**y=rand()%z;**

**g4[i]=a[y];**

**}**

**for(int i=0;i<7;i++)**

**{**

**y=rand()%z;**

**n8[i]=a[y];**

**}**

**send(newSocket2,&g4,sizeof(g4),0);**

**send(newSocket2,&n8,sizeof(n8),0);**

**recv(newSocket2,&l2,sizeof(l2),0);**

**int ms2=lcs(g4,n8,m2,n2);**

**send(newSocket2,&ms2,sizeof(ms2),0);**

**fprintf(c2,"Client requested LCS\n");**

**fprintf(c2,"String1 :");**

**for(int i=0;i<10;i++)**

**fprintf(c2,"%c",g4[i]);**

**for(int i=0;i<6;i++)**

**fprintf(c2,"%c",n8[i]);**

**fprintf(c2,"Server calculated LCS value is %d\n",ms2);**

**fprintf(c2,"Client calcualted LCS value is %d\n",l2);**

**if(ms2==l2)**

**{**

**score2=score2+1;**

**fprintf(c2,"Status:Client won the game\n");**

**}**

**else**

**fprintf(c2,"Status:Client Loss the game\n");**

**}**

**}**

**fprintf(c2,"Score :%d",score2);**

**printf("score sent to client2 successfully and client2 exited\n");**

**}**

**}**

**CLIENT CODE :**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

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

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

**#include <netinet/in.h>**

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

**void main(int argc,char\* argv[]){**

**int clientSocket;int profit;int var;int ci;int score;**

**struct sockaddr\_in serverAddr;**

**int buffer1[4],buffer2[4];**

**char buffer3[10],buffer4[6];**

**int andl,ands;**

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

**memset(&serverAddr, '\0', sizeof(serverAddr));**

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

**serverAddr.sin\_port = htons(atoi(argv[2]));**

**serverAddr.sin\_addr.s\_addr = inet\_addr(argv[1]);**

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

**printf("Welcome to algortim Guess Game\n");**

**printf("Connected to Game Server\n");**

**printf("here we are going to solve 2 famous algorithms\n");**

**printf("1)Fractional knapsack\n");**

**printf("2)LCS\n");**

**printf("Fractional Knapsack:The main objective is fiiling the knapsack"**

**"That maximizes the total profit earned,here the the total capacity of knapsack is 10\n\n");**

**printf("LCS:A subsequence is a sequence that appears in the same relative order"**

**",but not necessarily contiguous");**

**while(1)**

**{**

**printf("press 1 for knapsack and 2 for LCS and 0 for exiting the game");**

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

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

**if(ci==0)**

**{**

**recv(clientSocket,&score,sizeof(score),0);**

**printf("Your total score is %d\n",score);**

**printf("Thanks for playing the game");**

**break;**

**}**

**if(ci==1)**

**{**

**recv(clientSocket, &buffer1, sizeof(buffer1), 0);**

**recv(clientSocket, &buffer2, sizeof(buffer2), 0);**

**printf("\nweights: ");**

**for(int i=0;i<4;i++)**

**{**

**printf("%d ",buffer1[i]);**

**}**

**printf("\n");**

**printf("profits: ");**

**for(int i=0;i<4;i++)**

**{**

**printf("%d ",buffer2[i]);**

**}**

**printf("enter the total profit");**

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

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

**recv(clientSocket,&var,sizeof(var),0);**

**if(var!=profit)**

**printf("sorry better luck next time value is %d\n",var);**

**else**

**printf("congrtulations you won the game\n");**

**}**

**else**

**{**

**recv(clientSocket,&buffer3,sizeof(buffer3),0);**

**recv(clientSocket,&buffer4,sizeof(buffer4),0);**

**printf("\nStringA: ");**

**for(int i=0;i<10;i++)**

**{**

**printf("%c",buffer3[i]);**

**}**

**printf("\nStringB: ");**

**for(int i=0;i<6;i++)**

**{**

**printf("%c",buffer4[i]);**

**}**

**printf("\nenter the LCS value ");**

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

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

**recv(clientSocket,&ands,sizeof(ands),0);**

**if(ands==andl)**

**printf("congrats you won the game\n");**

**else**

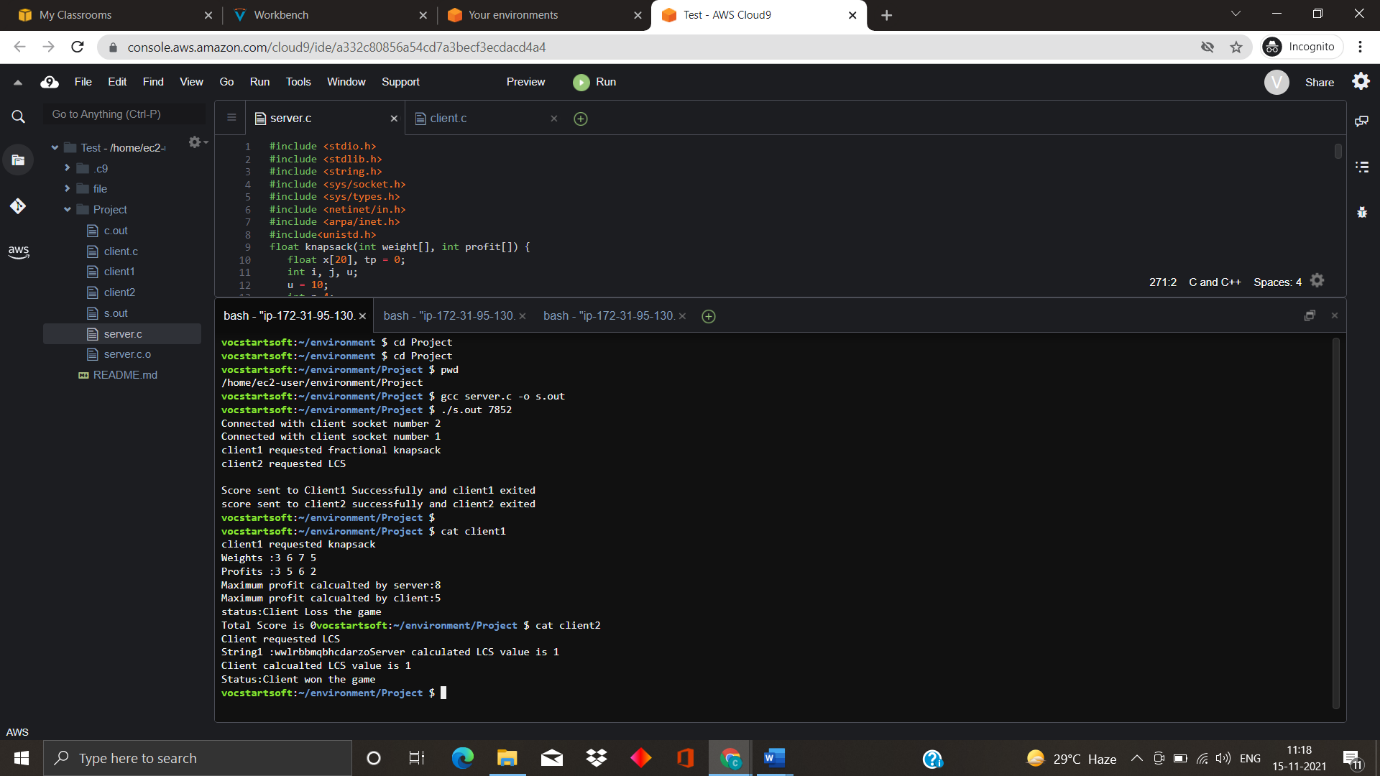
**printf("better luck nexttime value is %d\n",ands);**

**}**

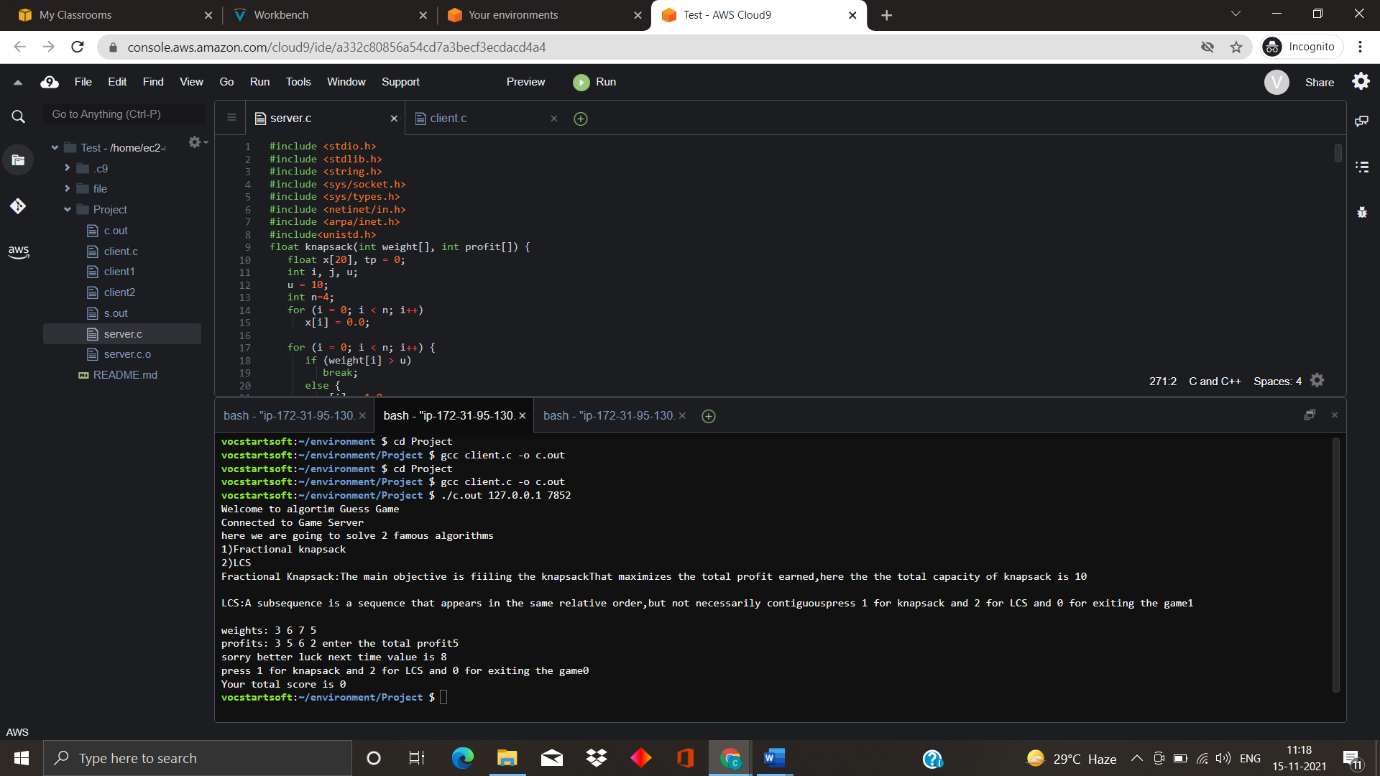
**}**

**}**

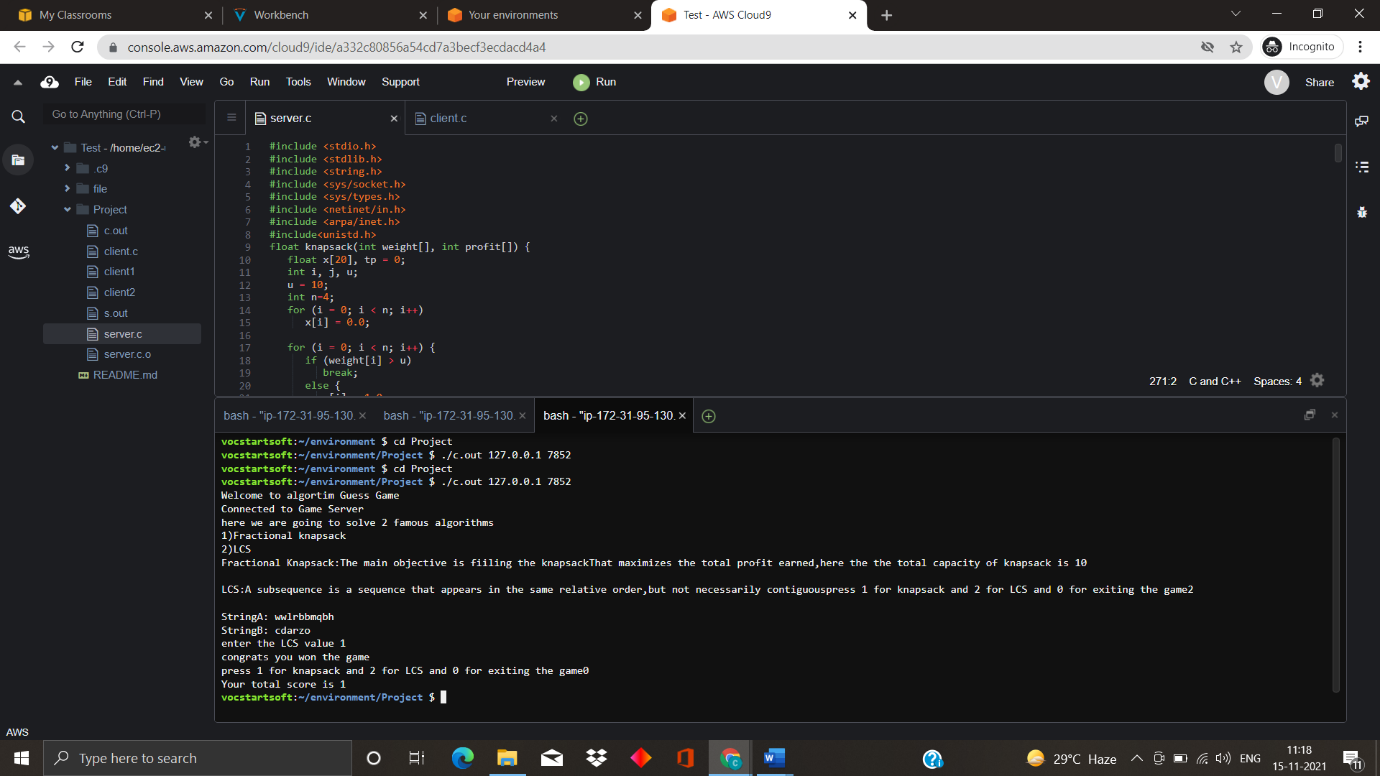
**SERVER :**



**CLIENT 1 :**



**CLIENT 2 :**



***CONCLUSION***

The game is a very good braim exercise. It involves solving Mathematical problems and solving different kinds of algorithms. In these tough times we innovate this project that cant boost people. Something that is fun and people can actually enjoy. Overall, the client and server implementation using socket programming works fine and is able to use. Hereby we come to an end of or project.