2. Write a program using TCP socket to implement the following: (20 marks)

i. Server maintains records of fruits in the format: fruit-name quantity Last-sold (server timestamp),

ii. Multiple client purchase the fruits one at a time,

iii. The fruit quantity is updated each time any fruit is sold,

iv. Show warning messages to the client if the quantity requested is not available.

v. Display the customer ids <IP, port> who has done transactions already. This list should be updated in the server everytime a transaction occurs.

vi. The total number of unique customers who did some transaction will be displayed to the customer everytime.

Solution: CLIENT SIDE:

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <arpa/inet.h>

int main(void){

int sockfd = 0,n = 0,num=0;

char recvBuff[1024];

char sendBuff[1025];

struct sockaddr\_in serv\_addr;

memset(recvBuff, '0' ,sizeof(recvBuff));

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0))< 0){

printf("\n Error : Could not create socket \n");

return 1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(5000);

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

if(connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr))<0){

printf("\n Error : Connect Failed \n");

return 1;

}

printf("Welcome to our shop\n");

num = recv(sockfd, recvBuff, sizeof(recvBuff),0);

if ( num <= 0 ){

printf("Either Connection Closed or Error\n");

}

recvBuff[num] = '\0';

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

recvBuff[0] = '\0';

printf("Please enter the first letter of the item you want to buy followed by the quantity\n");

fgets(sendBuff,1025,stdin);

if ((send(sockfd,sendBuff,strlen(sendBuff),0))== -1){fprintf(stderr, "Failure Sending Message\n");

close(sockfd);

}

num = recv(sockfd, recvBuff,sizeof(recvBuff),0);

if ( num <= 0 ){

printf("Either Connection Closed or Error\n");

}

recvBuff[num] = '\0';

printf("%s",recvBuff);

close(sockfd);

return 0;

}

SERVER SIDE:

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <time.h>

struct customer {

char product\_bought[20];

int qty;

char cli\_addr[16];

int cli\_port;

char time[80];

struct customer \*next;

};

struct customer \*Create\_Customer(){

struct customer \*x =(struct customer \*)malloc(sizeof(struct customer));

x->next=NULL;

return x;

}

int main(){

time\_t now;

struct tm \*ts;

char buf[80];

int listenfd = 0,connfd = 0,n=0,num=0;

struct sockaddr\_in serv\_addr;

struct sockaddr\_in client\_addr;

char sendBuff[1025];

char recvBuff[1024];

int numrv;

int mango=30, orange=30, guava=30, petrol=30 ,sav;

char new1[50];

int newn,i;

int bought=1;

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

printf("socket retrieve success\n");

memset(&serv\_addr, '0', sizeof(serv\_addr));

memset(sendBuff, '0', sizeof(sendBuff));

serv\_addr.sin\_family = AF\_INET;

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

serv\_addr.sin\_port = htons(5000);

bind(listenfd, (struct sockaddr\*)&serv\_addr,sizeof(serv\_addr));

if(listen(listenfd, 10) == -1){

printf("Failed to listen\n");

return -1;

}

printf("\nAvailable items");

printf("\nProduct\tQuantity");

printf("\nMango\t%d", mango);

printf("\nOrange\t%d",orange);

printf("\nGuava\t%d",guava);

printf("\nPetrol\t%d",petrol);

struct customer \*\*Customers,\*head,\*temp;

Customers = &head;

int customer\_count = 0;

while(1){

socklen\_t addr\_size = sizeof(client\_addr);

connfd = accept(listenfd, (struct sockaddr\*)&client\_addr, &addr\_size);

struct sockaddr\_in\* pV4Addr = (struct sockaddr\_in\*)&client\_addr;

struct in\_addr ipAddr = pV4Addr->sin\_addr;

char str[INET\_ADDRSTRLEN];

inet\_ntop( AF\_INET, &ipAddr, str, INET\_ADDRSTRLEN );

sprintf(sendBuff,"Number of Unique Customers so far: %d",customer\_count);

if ((send(connfd,sendBuff,strlen(sendBuff),0))== -1){

fprintf(stderr, "Failure Sending Message\n");

close(connfd);

break;

}

num = recv(connfd, recvBuff, sizeof(recvBuff),0);now = time(NULL);//time-stamping

ts = localtime(&now);

strftime(buf, sizeof(buf), "%a %Y-%m-%d %H:%M:%S %Z", ts);

if ( num <= 0 ){

printf("Either Connection Closed or Error\n");

}

recvBuff[num] = '\0';

for(i=1;i<strlen(recvBuff);i++){

new1[i-1]=recvBuff[i];

}

new1[i-1]='\0';

newn=atoi(new1);

if(recvBuff[0]=='m'){

sav=mango;

mango=mango-newn;

if(mango<0){

mango=sav;

printf("Requested quantity of Mango is not available");

bought=-1;

sprintf(sendBuff,"Requested quantity of Mango is not available");

}

if(bought>0){

\*Customers = Create\_Customer();

strcpy((\*Customers)->product\_bought,"Mango");

(\*Customers)->qty = newn;

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"Congratulations, transaction successfull");

}

}

if(recvBuff[0]=='o'){

sav=orange;

orange=orange-newn;

if(orange<0){

orange=sav;

printf("Requested quantity of Orange is not available");

bought = -1;

sprintf(sendBuff,"Requested quantity of Orange is not available");

}

if(bought>0){

\*Customers = Create\_Customer();

strcpy((\*Customers)->product\_bought,"Orange");

(\*Customers)->qty = newn;

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"Congratulations, transaction successfull");

}

}

if(recvBuff[0]=='g'){

sav=guava;

guava=guava-newn;

if(guava<0){

guava=sav;

printf("Requested quantity of Guava is not available");

bought = -1;

sprintf(sendBuff,"Requested quantity of Guava is not available");

}

if(bought>0){

\*Customers = Create\_Customer();

strcpy((\*Customers)->product\_bought,"Guava");

(\*Customers)->qty = newn;

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"Congratulations, transaction successfull");

}

}

if(recvBuff[0]=='p'){

sav=petrol;

petrol=petrol-newn;

if(petrol<0){

petrol=sav;

printf("Requested quantity of Petrol is not available");

bought = -1;

sprintf(sendBuff,"Requested quantity of Petrol is not available");

}

if(bought>0){

\*Customers = Create\_Customer();

strcpy((\*Customers)->product\_bought,"Petrol");

(\*Customers)->qty = newn;

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"Congratulations, transaction successfull");

}

}

printf("\nMango\t%d\nOrange\t%d\nGuava\t%d\nPetrol\t%d\n",mango,orange,guava,petrol);

struct customer \*t;

t=head;

int x=0;

while(t){

x++;

printf("\n===============%d===========",x);

printf("\nCustomer IP: %s",t->cli\_addr);

printf("\nCustomer port: %d",t->cli\_port);

printf("\nProduct Bought: %s",t->product\_bought);

printf("\nQuantity Bought: %d",t->qty);

printf("\nTime-Stamp: %s",t->time);

printf("\n\n");

t=t->next;

}

if ((send(connfd,sendBuff,strlen(sendBuff),0))== -1){

fprintf(stderr, "Failure Sending Message\n");

close(connfd);

break;

}

close(connfd);

sleep(1);

}

return 0;

}

OUTPUT:

