//Assignment tcp in C

/\*\*\*\*\* A7. A. Arithmatic client \*\*\*\*\*/

// Client

import java.io.\*;

import java.net.\*;

public class arithtcpclient

{

public static void main(String[] args) throws IOException

{

System.out.println();

System.out.println("ARITHMETIC CLIENT");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

try

{

Socket clientsoc = new Socket("localhost", 6666);

System.out.println("Enter the inputs");

PrintWriter out = new PrintWriter(clientsoc.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(clientsoc.getInputStream()));

BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));

String userinput;

while (true)

{

do

{

userinput = stdin.readLine();

out.println(userinput);

}while(!userinput.equals("."));

System.out.println("Sever Says : " + in.readLine());

}

}

catch(Exception e)

{

System.exit(0);

}

}

}

/\*\*\*\* A7. A. Arithmatic Server \*\*\*\*/

//arithtcpserver

import java.io.\*;

import java.net.\*;

public class arithtcpserver

{

public static void main(String arg[]) throws Exception

{

System.out.println();

System.out.println("ARITHMETIC SERVER");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Server is ready to accept inputs…");

ServerSocket serversoc=new ServerSocket(6666);

Socket clientsoc = serversoc.accept();

PrintWriter out = new PrintWriter(clientsoc.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(clientsoc.getInputStream()));

String inputline;

BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));

try

{

while (true)

{

String s,op="",st;

int i=0,c=0;

int[] a=new int[2];

while(true)

{

s=in.readLine();

if(s.equals("+") || s.equals("-") || s.equals("\*") || s.equals("/"))

op=s;

else if(s.equals("."))

break;

else

{

a[i]=Integer.parseInt(s);

i++;

}

}

if(op.equals("+"))

c=a[0]+a[1];

else if(op.equals("-"))

c=a[0]-a[1];

else if(op.equals("\*"))

c=a[0]\*a[1];

else if(op.equals("/"))

c=a[0]/a[1];

s=Integer.toString(c);

out.println(s);

}

}

catch(Exception e)

{

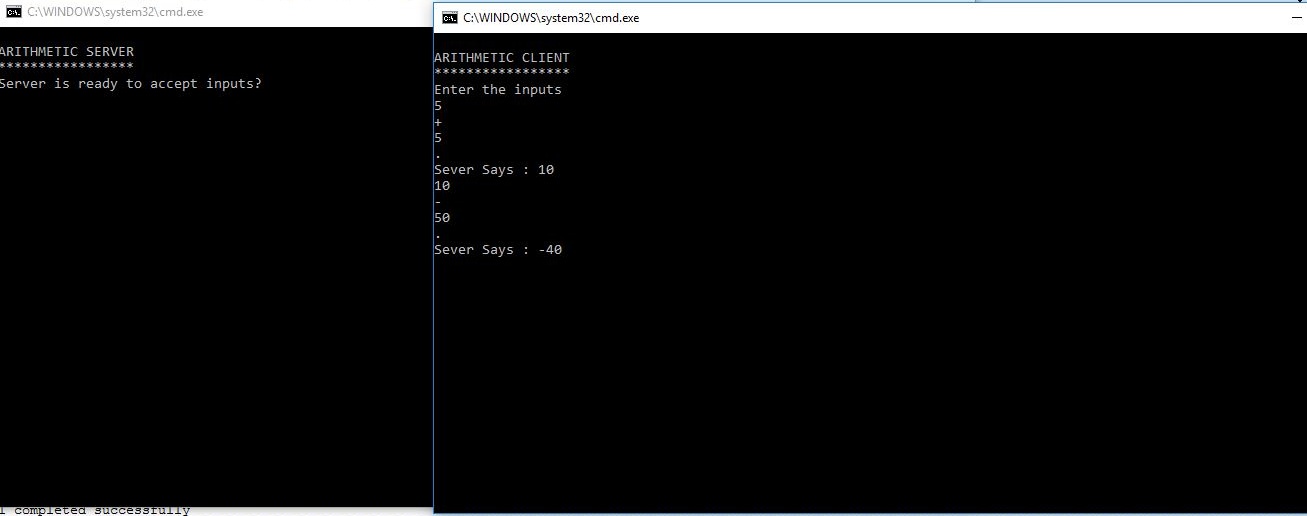
System.exit(0);

}

}

}

\*\*\*\*\*OUTPUT\*\*\*\*\*



/\*\*\* A7.B. File transfer client \*\*\*/

#include<sys/types.h>

#include<sys/socket.h>

#include<stdio.h>

#include<netinet/in.h>

#include <unistd.h>

#include<string.h>

#include<strings.h>

#include <arpa/inet.h>

#include <stdlib.h>

#define MAXBUFLEN 1000000

//#define buffsize 150

void main()

{

int b,sockfd,sin\_size,con,n,len;

//char buff[256];

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

printf("Socket Created Sucessfully \n");

struct sockaddr\_in servaddr;

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=6007;

sin\_size = sizeof(struct sockaddr\_in);

if((connect(sockfd,(struct sockaddr \*) &servaddr, sin\_size))==0) //initiate a connection on a socket

printf("Connect Sucessful \n");

char buffer[10000];

char c[10000];

FILE \*fp;

//bzero(buffer,10000);

//bzero(c,10000);

read(sockfd, buffer, 10000);

fp= fopen("/home/Student/1/CNL/Group A/Assg2\_TCPSocket/Assg2\_b/receive.txt", "w+");

/\* Read and display data \*/

fwrite(buffer, 1,strlen(buffer) + 1, fp);

//fseek(fp, 0, SEEK\_SET);

//fread(c, strlen(buffer)+1, 1, fp);

printf("Received File Contents :%s \n", buffer);

fclose(fp);

close(sockfd);

}

/\*\*\* A7.B. File transfer server \*\*\*/

#include<sys/types.h>

#include<sys/socket.h>

#include<stdio.h>

#include<netinet/in.h>

#include <unistd.h>

#include<string.h>

#include <arpa/inet.h>

#include <stdlib.h>

#define MAXBUFLEN 1000000

void main()

{

int b,sockfd,connfd,sin\_size,l,n,len;

if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))>=0) //socket creation

printf("Socket Created Sucessfully \n"); //on success 0 otherwise -1

struct sockaddr\_in servaddr;

struct sockaddr\_in clientaddr;

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=6007;

if((bind(sockfd, (struct sockaddr \*)&servaddr,sizeof(servaddr)))==0) //bind() assigns the

// address specified by addr to the socket referred to by the file

// descriptor sockfd. addrlen specifies the size, in bytes, of the

// address structure pointed to by addr. Traditionally, this operation is

// called â€œassigning a name to a socketâ€.

printf("Bind Sucessful \n");

if((listen(sockfd,5))==0) //listen for connections on a socket

printf("Listen Sucessful \n");

sin\_size = sizeof(struct sockaddr\_in);

if((connfd=accept(sockfd,(struct sockaddr \*)&clientaddr,&sin\_size))>0)

printf("Accept Sucessful \n");

char buffer[100];

char c[10000] = "this is file transfer program";

//char source[MAXBUFLEN + 1];

//bzero(buffer,10000);

FILE \*fp;

fp= fopen("/home/Student/1/CNL/Group A/Assg2\_TCPSocket/Assg2\_b/send.txt", "w+");

//fp = fopen("file.txt", "w+");

/\* Write data to the file \*/

fwrite(c, 1, strlen(c) + 1, fp);

/\* Seek to the beginning of the file \*/

fseek(fp, 0, SEEK\_SET);

/\* Read and display data \*/

fread(buffer, 1,strlen(c)+1, fp);

// fclose(fp);

write(connfd, buffer, strlen(buffer));

printf("Sent File Contents: %s\n", buffer);

fclose(fp);

close(sockfd);

}

\*\*\*OUTPUT\*\*\*

