Segundo Certamen
Tiempo 120 min. Al terminar, entregar sus respuestas vía AULA.

Usted deberá desarrollar programas con funcionalidades muy similares a los de la Tarea 3.

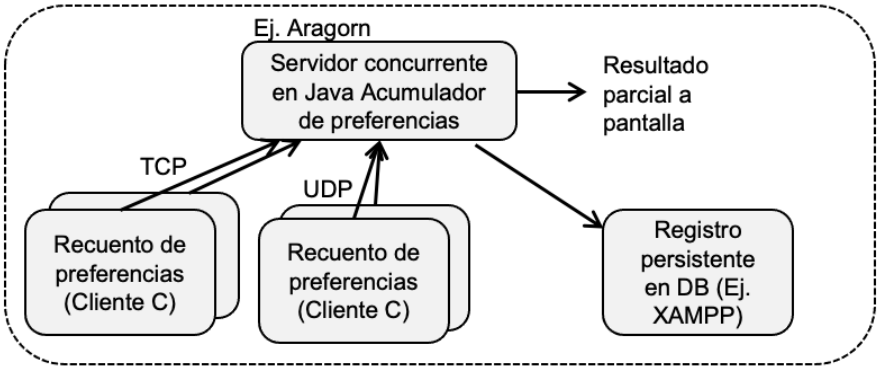

Figura 1: Procesos involucrados en la situación del certamen

De los módulos mostrados en Figura 1, **no se pide** programar el cliente TCP en C para el recuento de votos, sin embargo, el servidor concurrente en Java acumulador de preferencias sí debe contemplar en su programación la existencia de a los más 2 clientes TCP de manera simultánea.

Pregunta 1 (35 pts): Programe en lenguaje C el servidor de **recuento UDP**. Este programa fuente en C simula el escrutinio de preferencias en una elección.
Sintaxis: recuento <servidor> <puerto>
Descripción: recuento crea un **socket UDP** y envía una letra A o letra B según la preferencia ingresada por el teclado (letras a o b). Presenta un menú breve con estas dos opciones más la opción f para indicar el fin del recuento en ese local de votación.
Suba su respuesta a AULA con nombre recuento.c

El código de la pauta lo puede ver aquí.

```
#include <string.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
/* usage: recuento <server> <puerto> */

int main(int argc, char * argv[])  {
    int n, s, len;
    char preferencia;
    char input[10];
    struct hostent *hp;
    struct sockaddr_in name;

    hp = gethostbyname(argv[1]);
    s = socket(AF_INET, SOCK_DGRAM, 0);
    name.sin_family = AF_INET;
    name.sin_port = htons(atoi(argv[2]));
    memcpy(&name.sin_addr, hp->h_addr_list[0], hp->h_length);
    len = sizeof(struct sockaddr_in);

    printf(" Ingrese \n a: voto por A \n b: voto por B\n f: fin de recuento.\n");
    while ((n=read(0, input, sizeof(input))) > 0) {
        input[n]='\0';
        input[0] = input[0]-'a'+'A';  /* upercase*/
        if (input[0] == 'A' || input[0] =='B') {
            printf("Sending \"%s\"\n", input);
```

```
            sendto(s, input, 1, 0, (struct sockaddr*) &name, len);
        }
        else if (input[0]=='F') exit(0);
            else printf(" Ingrese \n a: voto por A \n b: voto por B\n f: fin de recuento.\n");
    }
}
```

Pregunta 2 (45 pts): Programe en lenguaje Java el servidor concurrente Acumulador.java.
Sintaxis: $java Acumulador <puerto de escucha>
Descripción: En el "puerto de escucha", Acumulador atiende concurrentemente varios clientes escrutadores de votos, a los más 2 son clientes TCP. A la llegada de una letra A desde un cliente TCP o uno UDP, Acumulador incrementa los votos de esa opción. Acumulador hace lo análogo a la llegada de una letra B. Ante cada actualización de votos de A o B, Acumulador muestra por pantalla el contador de preferencias para cada opción (A y B) hasta ese momento.
Suba su respuesta a AULA con nombre Acumulador.java, en él incluya otras clases posibles.

El código de la pauta lo puede ver aquí.

```
// usage $ java Acumulador <port>
import java.io.*;
import java.net.*;
public class Acumulador {
   public static void main(String[] args ) throws Exception {
         Conteo conteo = new Conteo();
         Thread tcpService = new TCPClientServer(Integer.parseInt(args[0]), conteo);
         Thread udpService = new UDPClientService(Integer.parseInt(args[0]), conteo);
         tcpService.start();
         udpService.start();
   }
}

class Conteo {
   synchronized public void incrA(){ A++; }
   synchronized public void incrB(){ B++; }
   synchronized public void printCount() {
      System.out.println("Preferencias: A:"+A+"  B:"+B);
   }
   private int A=0, B=0;
}

class TCPClientServer extends Thread  {
   public TCPClientServer(int port, Conteo cont) throws Exception {
      s = new ServerSocket(port);
      conteo = cont;
   }
   public void run() {
      while (true) {
         try {
            Socket client = s.accept( );
            Thread t = new TCPClientService(client, conteo);
            System.out.println("New TCP client");
            t.start();
         } catch (Exception e){ e.printStackTrace(); }
      }
   }
   private ServerSocket s;
   private Conteo conteo;
}

class TCPClientService extends Thread  {
   public TCPClientService (Socket cs, Conteo cont) {
      incoming = cs;
      conteo = cont;
   }
   public void run() {
      try {
         BufferedReader in = new BufferedReader
```

```
            (new InputStreamReader(incoming.getInputStream()));
         boolean done = false;
         while (!done) {
            String str = in.readLine();
            if (str == null) done = true;
            else {
               if (str.charAt(0)=='A') conteo.incrA();
               else conteo.incrB();
               conteo.printCount();
            }
         }
         incoming.close();
      } catch (Exception e) { e.printStackTrace();}
   }
   private Socket incoming;
   private Conteo conteo;
}

class UDPClientService extends Thread  {
   public UDPClientService (int port, Conteo cont) throws Exception {
      incoming = new DatagramSocket(port);
      conteo=cont;
   }

   public void run() {
      try {
         DatagramPacket dp = new DatagramPacket(new byte[10],10);
         while(true) {
            incoming.receive(dp);
            if (dp.getData()[0]=='A') conteo.incrA();
            else conteo.incrB();
            conteo.printCount();
         }
      } catch (Exception e) { e.printStackTrace(); }
   }
   private DatagramSocket incoming ;
   private Conteo conteo;
}
```

Pregunta 3 (20 pts): Considere que un servidor XAMPP está corriendo en la misma máquina donde se ejecuta el servidor "Acumulador". Suponga que ya existe la base de datos "Votacion" en MariaDB de XAMPP y que en el constructor de Acumulador ya se cuenta con la conexión a la base de datos y la inicialización del atributo de la clase sentencia, instancia de Statement.

a) En clase Acumulador agregue método createTable(), el cual crea la tabla ConteoGeneral con dos atributos de la tabla: "Correlativo" de tipo INT(11) y "Preferencia" de tipo CHAR.
b) En clase Acumulador agregue método storeVote(int corr, char vote), para ingresar en la base de datos el número correlativo y el voto recibido.

Suba la implementación de estos dos métodos en archivo metodos.java o bien agréguelos a Acumulador_DB.java. (versión extendida de su respuesta a pregunta 2).

El código de la pauta lo puede ver aquí.

```
import java.io.*;
import java.net.*;
import java.sql.*;
public class Acumulador_DB {
   private Statement sentencia;
//....parte omitida
   public void createTable() {  // Metodo pedido
      try {
         sentencia.execute( "DROP TABLE ConteoGeneral" );
      } catch( SQLException e ) {System.out.println( "Tabla no eliminada" );}
      try {
         sentencia.executeUpdate( "CREATE TABLE ConteoGeneral (Correlativo INT(11), Preferencia
CHAR)");
      } catch( SQLException e ) {System.out.println( "Tabla no eliminada" );}
```

```
   }
   public void storeVote(int corr, char vote) { // Metodo pedido
      try {
         sentencia.executeUpdate("INSERT INTO ConteoGeneral VALUES (' "+corr+"','"+vote+"' )");
      } catch (Exception e) {System.out.print(e);}
   }

   public static void main(String[] args ) throws Exception {
         Acumulador_DB db= new Acumulador_DB();
         db.createTable();
         Conteo conteo = new Conteo(db);
         Thread tcpService = new TCPClientServer(Integer.parseInt(args[0]), conteo);
         Thread udpService = new UDPClientService(Integer.parseInt(args[0]), conteo);
         tcpService.start();
         udpService.start();
   }
}
// resto del programa
```