Certamen Final 100 minutos. Puede usar Apuntes.

Todas las preguntas tienen igual puntaje.

1.- Para el programa listado al final, qué valor se obtiene por pantalla cuando es ejecutado, en los siguientes casos:

a) código de deposit de la case Account es:

```
void deposit (int amount, String name) {
  int balance;
  synchronized(this) {
     out. println(name + " trying to deposit " + amount);
     out. println(name + " getting balance...");
     balance = getBalance();
     out. println(name + " balance got is " + balance);
     balance+= amount;
     out. println(name + " setting balance...");
     setBalance(balance);
   }
   out. println(name + " new balance set to " + Deposit2.balance);
 }
```

b) Cuando es:

```
void deposit (int amount, String name) {
  int balance;
  synchronized(name) {
     out. println(name + " trying to deposit " + amount);
     out. println(name + " getting balance...");
     balance = getBalance();
     out. println(name + " balance got is " + balance);
     balance+= amount;
     out. println(name + " setting balance...");
     setBalance(balance);
   }
   out. println(name + " new balance set to " + Deposit2.balance);
 }
```

c) Cuando es:

```
void deposit (int amount, String name) {
  int balance;
  out. println(name + " trying to deposit " + amount);
  out. println(name + " getting balance...");
  balance = getBalance();
  synchronized(this) {
     out. println(name + " balance got is " + balance);
     balance+= amount;
     out. println(name + " setting balance...");
     setBalance(balance);
   }
   out. println(name + " new balance set to " + Deposit2.balance);
 }
```

a) 9 pts. Corresponde al programa original el cual genera:

```
#1 trying to deposit 1000
#1 getting balance...
#1 balance got is 1000
#1 setting balance...
    #2 trying to deposit 2000
    #2 getting balance...
#1 new balance set to 2000
    #2 balance got is 2000
```

```
    #2 setting balance...
    #2 new balance set to 4000
***** Final Balance is 4000
```

b) 8 pts En este caso el objeto usado como control de la zona de exclusión mutua es uno de los argumentos. Entonces para distintos argumentos las zonas críticas serán distintas. Como en cada llamado este objeto es distinto cada thread o hilo manejará una zona distinta y el objetivo no final de la aplicación no es logrado.

```
#1 trying to deposit 1000
#1 getting balance...
    #2 trying to deposit 2000
    #2 getting balance...
#1 balance got is 1000
#1 setting balance...
    #2 balance got is 1000
    #2 setting balance...
#1 new balance set to 2000
    #2 new balance set to 3000
***** Final Balance is 3000
```

c) 8 pts. En este caso se usa un único objeto para controlar el acceso exclusivo, sin embargo por ponerse luego de obtener el balance actual el resultado tampoco es el deseado.

```
#1 trying to deposit 1000
#1 getting balance...
    #2 trying to deposit 2000
    #2 getting balance...
#1 balance got is 1000
#1 setting balance...
    #2 balance got is 1000
    #2 setting balance...
#1 new balance set to 2000
    #2 new balance set to 3000
***** Final Balance is 3000
```

2.- La función rand() de C no es segura para uso con hilos. Esto es, si dos hilos la invocan en forma concurrente, su resultado no es el esperado. Proponga la función randSafe() la cual retorna el mismo valor de rand pero sin conflicto de uso con hilos (Nota: usted dentro de randSafe puede llamar a rand)

```
#include <pthread.h>
#include <stdlib.h>

int randsafe(void) {
   static pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
   double ran;

   pthread_mutex_lock(&lock);
   ran = rand();
   pthread_mutex_unlock(&lock);
   return ran;
}
```

3.-Desarrolle una aplicación Java servidora en puerto 1234 que atienda usando hilos respuestas de clientes TCP/IP los cuales se conectan a ella y sólo envían una letra a o b. El programa servidor debe acumular el número de cada letra recibida y ante la recepción de t (total) desde un cliente TCP, debe mostrar en pantalla del servidor el número total de cada letra y terminar.

```
import java.io.*;
import java.net.*;

public class CuentaRespuestas
{
   public static void main(String[] args ){
     try {
       ServerSocket s = new ServerSocket(1234);
       CuentaRespuestas cr = new CuentaRespuestas();
       while (true) {
         Socket incoming = s.accept( );
         ServerThread st = new ServerThread(incoming, cr);
         st.start();
       }
     } catch (Exception e){
       e.printStackTrace();
     }
   }
   public synchronized void incrementA(){
      a++;
   }
   public synchronized void incrementB(){
      a++;
   }
   public void showResult(){
      System.out.println("A answers: "+ a);
      System.out.println("B answers: "+ b);
      System.exit(0);
   }
   private int a=0;
   private int b=0;
}
```

```
class ServerThread extends Thread
{
  public ServerThread (Socket incoming, CuentaRespuestas cr) {
    inSocket = incoming;
    contadores = cr;
  }

  public void run() {
    try {
      BufferedReader in = new BufferedReader
         (new InputStreamReader(inSocket.getInputStream()));
      String line = in.readLine();
      if (line.trim().equals("a")) contadores.incrementA();
      if (line.trim().equals("b")) contadores.incrementB();
      if (line.trim().equals("t")) contadores.showResult();
      inSocket.close();
    } catch (Exception e){
      e.printStackTrace();
    }
  }
  private CuentaRespuestas contadores;
  private Socket inSocket;
}
```

4.- Suponga que contamos con el programa ***serial.c*** y su ejecutable ***serial***, el cual toma la entrada estándar y la transmite por la puerta serial y envía a la salida estándar todo lo que lega por la puerta serial (RS232, USB, u otra).
Haga un programa en Java, EcoCopia.java, que envíe a pantalla y devuelva todo lo recibido por la puerta serial.

```
import java.io.*;
class EcoCopia
{
  public static void main(String[ ] args) {
    Runtime runTime= Runtime.getRuntime(); /* 6 pts */
    Process process=null;
    try {
      process = runTime.exec("serial"); /* 6 pts */
      BufferedReader in = new BufferedReader(
        new InputStreamReader(process.getInputStream())); /*3 pts*/
      PrintWriter out = new PrintWriter                   /* 3 pts*/
            (process.getOutputStream(), true /* autoFlush */);
    } catch (IOException e) {
      System.out.println("No pudimos correr serial");
      System.exit(-1);
    }
    while (trae) {  /* 7 pts */
      String line = in.readline();
      out.println(line);
      System.out.println(line);
```

```
     }
   }
}


import java.io.*;

public class Deposit2 {
    static int balance =1000;  // simulate balance kept remotely

    public static void main (String argc[]) {
        PrintWriter out = new PrintWriter(System.out, true);
        Account account = new Account (out);
        DepositThread first, second;
        first = new DepositThread (account, 1000, "#1");
        second = new DepositThread(account, 2000, "\t\t\t\t#2");
        // start the transactions
        first.start();
        second.start();
        // wait for both transactions to finish
        try {
            first.join();
            second.join();
        } catch (InterruptedException e) {}
        // print the final balance
        out. println("***** Final Balance is " + balance);
    }
}

class Account {
    PrintWriter out;
    Account (PrintWriter out) {
        this.out = out;
    }
    void deposit (int amount, String name) {
        int balance;
        synchronized(this) {
            out. println(name + " trying to deposit " + amount);
            out. println(name + " getting balance...");
            balance = getBalance();
            out. println(name + " balance got is " + balance);
            balance+= amount;
            out. println(name + " setting balance...");
            setBalance(balance);
        }
        out. println(name + " new balance set to " + Deposit2.balance);
    }

    int getBalance() {
        try {  // simulated delay in getting balance remotely
            Thread.sleep(5000);
        } catch (InterruptedException e) {}
        return Deposit2.balance;
    }

    void setBalance (int balance) {
        try {  // simulate the delay in setting new balance remotely
            Thread.sleep(5000);
            } catch(InterruptedException e) {}
        Deposit2.balance = balance;
```

```
    }
}


class DepositThread extends Thread
{
    Account account;
    int depositAmount;
    String message;
    DepositThread (Account account, int amount, String message) {
        this.message = message;
        this.account = account;
        this.depositAmount = amount;
    }
    public void run () {
        account.deposit (depositAmount, message);
    }
}
```