Parte A (12 valores)

1. A disponibilização generalizada de serviços e a apetência demonstrada pelos sistemas computacionais paralelos para a execução cooperativa de aplicações levantam a questão de se saber quão seguros são os fluxos de informação trocados e quão segura é a informação armazenada nos recursos associados. Indique os diferentes níveis em que se pode colocar a segurança de informação e descreva medidas que podem ser usadas para reduzir os riscos.

2. Uma forma de implementação do modelo cliente-servidor é efectuar a replicação do servidor sempre que há uma solicitação de um novo cliente. Mostre que vantagens e inconvenientes é que esta arquitectura apresenta.

3. A *serialização* de objectos em Java constitui uma abordagem possível ao problema da comunicação de dados em ambientes distribuídos. Explique neste contexto em que consiste o *marshalling* e o *unmarshalling* de informação.

4. O modelo de invocação remota de métodos (RMI) constitui um modelo de comunicação com uma abstracção mais elevada do que o modelo puro de passagem de mensagens implementado com sockets. Mostre porquê.

Parte B (8 valores)

As classes de Java apresentadas abaixo descrevem um modelo produtor-consumidor muito simples.

```
public class Distributor extends Thread
{
    private FiFo fifo;
    private Location loco, loce;

    public Distributor (FiFo f, Location lo, Location le)
    {
       fifo = f;    loco = lo;    loce = le;
    }

    public synchronized void run()
    {
      int val;
      for (int i = 0; i < 10; i++)
      { val = fifo.out ();
        if (val % 2 == 0)
          { loce.in (val);
            System.out.println ("Valor par distribuido!");
          }
          else { loco.in (val);
                 System.out.println ("Valor impar distribuido!");
               }
        try
        {
          sleep((int)(Math.random() * 100));
        }
        catch (InterruptedException e) { }
      }
        try
        { wait ();
        }
        catch (InterruptedException e) { }
    }

    public synchronized void resu()
    {
      notifyAll ();
    }
}
```

```
public class Consumer extends Thread
{
  private Location loc;

  public Consumer (Location l)
  {
     loc = l;
  }

  public synchronized void run ()
  {
    int value = 0;
    for (int i = 0; i < 5; i++)
      value = value + loc.out ();
    try
    { wait ();
    }
    catch (InterruptedException e) { }
    System.out.println ("A soma e " + value);
  }

  public synchronized void resu()
  {
    notifyAll ();
  }
}
```

```
public class FiFo
{
  private int N;
  private int [] contents;
  private boolean empty;
  private boolean full;
  private int pin, pout;

  public FiFo (int N)
  {
     this.N = N;
     contents = new int [N];
     empty = true;
     full = false;
     pin = 0;
     pout = 0;
  }

  public synchronized int out ()
  {
    int tmp;
    while (empty)
    { try
      { wait ();
      }
      catch (InterruptedException e) { }
    }
    tmp = contents[pout];
    pout = (pout +1) % N;
    if (pin == pout) empty = true;
    if (!empty && (pout == (pin+1) % N)) full = false;
    notifyAll ();
    return tmp;
  }
```

```
    public synchronized void in (int value)
    {
      while (full)
      { try
        { wait, ();
        }
        catch (InterruptedException e) { }
      }
      contents[pin] = value;
      pin = (pin +1) % N;
      if (pin == pout) full = true;
      if (!full && (pin == (pout+1) % N)) empty = false;
      notifyAll ();
    }
}
```

```
public class Location
{
    private int contents;
    private boolean empty;

    public Location ()
    {
       empty = true;
    }

    public synchronized int out ()
    {
      while (empty)
      {
        try
        { wait ();
        }
        catch (InterruptedException e) { }
      }
      empty = true;
      notifyAll ();
      return contents;
    }

    public synchronized void in (int value)
    {
      while (! empty)
      { try
        { wait ();
        }
        catch (InterruptedException e) { }
      }
      contents = value;
      empty = false;
      notifyAll ();
    }
}
```

```
public class Cooperation
{
    public static void main (String[] args)
    {
       FiFo f = new FiFo (5);
       Location lo = new Location (),
                le = new Location ();
       Distributor p = new Distributor (f, lo, le);
       Consumer co = new Consumer (lo);
       Consumer ce = new Consumer (le);
       int [] val = {1, 3, 5, 0, 2, 4, 7, 6, 8, 9};
```

```
            p.start();
            co.start();
            ce.start();
            for (int i = 0; i < 10; i++)
              f.in (val[i]);
            p.resu ();
            co.resu ();
            ce.resu ();
        }
    }
```

a) Que valores são impressos no écran do monitor vídeo quando este programa é executado?
b) Que alterações teria que introduzir ao programa anterior se quisesse obter somas parcelares de valores módulo 5? Justifique a sua resposta.
c) Altere o programa anterior para permitir processar os valores por ordem inversa da sua introdução.