LEIC/LETI 2017/18, Repescagem 1º Teste de Sistemas Distribuídos, 3 de julho de 2018

Responda no enunciado, usando apenas o espaço fornecido. Identifique todas as folhas.
Uma resposta errada numa escolha múltipla desconta 1/N do valor da pergunta (sendo N o número de opções).
Duração da prova: 1h30m

Grupo I [7v]

1. Considere o seguinte programa de um cliente, programado em Java, que recorre a sockets UDP para invocar uma operação remota que regista um novo utilizador num servidor, programado noutra linguagem de programação.

```
class Cliente {
   public static void main(String args[]) throws Exception
   {
      String nomeServidor = "exemplo.sd.tecnico.ulisboa.pt";
      int portoServidor = 3456;
      DatagramSocket s = new DatagramSocket();
      InetAddress enderecoIP = InetAddress.getByName(nomeServidor);

      //Argumentos do pedido
      String nome = "José Silva";
      int idade = 30;
      int codigoPostal = 1230;

      //Identificador da operação *regista*
      int op = 1;

      //Serializa o id de operacao e argumentos num array de bytes
      byte[] pedido = ...

      DatagramPacket pacotePedido =
          new DatagramPacket(pedido, pedido.length, enderecoIP, portoServidor);
      s.send(pacotePedido);
      DatagramPacket pacoteResposta =
          new DatagramPacket(pacoteResposta, pacoteResposta.length);
      s.receive(pacoteResposta);

      //Desserializa a resposta
      int resposta = ...

      System.out.println("Resposta recebida: " + resposta);
      s.close();
}}
```

   a) [1,1v] Pretende-se portar este sistema para SUN RPC. Com base na informação no código acima, componha o extrato do ficheiro .x que definirá a interface remota do novo servidor. Caso o código acima não lhe dê informação suficiente para conhecer alguns elementos do .x, escolha-os livremente.

```
typedef registoArgs {
  string nome<...>;
  int idade;
  int codigoPostal;
}
program EXEMPLOPROG {
  version EXEMPLOVERS {
    int registo(registoArgs) = 1;
    ...
  } = 1;
} = ...;
```

b) [1,1v] Apresente agora um programa cliente baseado em SUN RPC que invoca a mesma operação (com mesmos argumentos) que no exemplo de cliente Java. O seu programa pode recorrer a pseudo-código, desde que não omita os parâmetros principais das funções chamadas.

```
int main() {
  CLIENT *clnt;
  registoArgs a;
  int *r;

  clnt = clnt_create("exemplo.sd.tecnico.ulisboa.pt", EXEMPLOPROG, EXEMPLOVERS,
"udp");
  if (clnt == NULL) {
    /* tratamento de erro */
  }

  strcpy(a.nome, "José Silva");
  a.idade = 30;
  a.codigoPostal = 1230;

  r = registo_1(&a, clnt);
  if (r == NULL) {
    /* tratamento de erro */
  }

  printf("Resposta recebida: %d\n", *r);
  clnt_destroy(clnt);
  exit(0);
}
```

c) Para cada aspeto focado nas alíneas seguintes, indique se existem diferenças entre a forma como cada cliente (Java/sockets e C/SUN RPC) resolve esse aspeto. Se sim, apresente as diferenças.

i) [0,8v] Descoberta do porto do servidor por parte do cliente.

Programa apresentado (com sockets): porto é bem conhecido, especificado na constante.
Programa com SUN RPC: porto é descoberto dinamicamente via rpcbind.

ii) [0,8v] O impacto de falhas pontuais na rede aquando de uma chamada remota.

Programa apresentado (com sockets): não tolera falhas de comunicação que ocorrem com UDP.
Programa com SUN RPC: semântica pelo-menos-uma-vez tolera perdas pontuais de mensagens.

iii) [0,8v] Heterogeneidade na forma como as arquiteturas do cliente e do servidor representam inteiros localmente.

Programa apresentado (com sockets): não trata da heterogeneidade.
Programa com SUN RPC: stubs traduzem automaticamente parâmetros usando representação XDR.

2. Considere um servidor RPC que oferece uma função remota `transfere(A, B, n)` que transfere um número `n` de criptomoedas a partir da conta do utilizador `A` para a conta do utilizador `B`; assim que a transferência seja confirmada, a função devolve um *timestamp* que indica o instante temporal em que tal aconteceu.
   Como a rede pela qual os clientes invocam esta operação é pouco fiável, testou-se este serviço usando as semânticas *talvez*, *pelo-menos-uma-vez* e *no-máximo-uma-vez*. Em cada experiência, observaram-se situações estranhas, que se apresentam abaixo.
   Para cada situação, indique em qual das semânticas ela se manifestou e forneça uma explicação plausível.

   a) [0,8v] Alguns utilizadores reclamaram pois observaram transferências duplicadas a partir da sua conta.

Semântica(s):
Explicação:

   b) [0,8v] Alguns utilizadores reportaram que, em algumas situações, o *timestamp* retornado dizia respeito a um momento muito anterior àquele em que a mensagem de resposta foi enviada.

Semântica(s):
Explicação:

   c) [0,8v] Alguns utilizadores observaram situações em que, embora o programa cliente tenha dado erro, a transferência afinal veio a ser executada.

Semântica(s):
Explicação:

Grupo II [6v]

Em Java, considere as seguintes interfaces:

```
public interface Map {
    void insert(String key, Element el);
    Element lookup(String key);
    void remove(String key);
}
public interface Element { ... }
```

1. O que seria necessário alterar nas interfaces acima para suportar cada cenário seguinte?
    a. [0,8v] Máquina A tem referência para instância de Map na máquina B.
    A instância de Map referencia múltiplas instâncias de Element, todas garantidamente em B.

    b. [0,8v] Máquina A tem referência para instância de Map na mesma máquina.
    A instância de Map referencia múltiplas instâncias de Element, cada uma localizada numa máquina remota distinta.

    c. [0,8v] Indique que instâncias de *proxy* existem em cada cenário acima e as respetivas classes.

2. [1,2v] Considere o cenário 1.a) acima. Complete o código abaixo do programa cliente que corre na máquina A para que este obtenha uma referência para o elemento identificado pela chave "E1". Assuma que o objeto do tipo Map está registado no RMI Registry com o nome "sd.tecnico.ulisboa.pt/mapa".
    Simplificação: omita do seu programa a definição do SecurityManager.

```
public static void main(String args[]) {









}
```

3. Considere agora o cenário 1.b). Quando o método remove("E1") é executado, levando a que o mapa descarte a única referência remota que tinha para a máquina C (onde o elemento "E1" reside), que mensagens são trocadas pelo protocolo de *garbage collection*? Explicite quem envia e quem recebe a mensagem.
    a. [0,8v] Responda assumindo um *garbage collector* baseado em contagem de referências.

    b. [0,8v] Responda assumindo um *garbage collector* baseado em *leases*.

A máquina virtual A, tendo um *lease* que já não necessita, pode simplesmente deixá-lo expirar; ao fim do tempo de validade do *lease*, o garbage collector local de C assumirá que existe menos uma referência para o elemento "E1".

4. [0,8v] O nome "sd.tecnico.ulisboa.pt/mapa" é impuro. De que forma é que essa propriedade é aproveitada para acelerar a resolução deste nome?

Este nome, com a sua estrutura hierárquica impura, permite que a sua resolução seja restringida a consultas aos diretórios correspondentes a cada domínio identificado no nome (ou a um subconjunto menor, como resultado de mecanismos de *caching*), em vez de exigir contactar o universo completo de diretórios.

Grupo III [7]

1) A Amazon Product API permite a pesquisa de produtos na loja a partir de palavras-chave, entre outras funcionalidades. Considere o seguinte fragmento de XML Schema :

```
01. <xs:complexType name="ItemLookupRequest"
02. xmlns:xs="http://www.w3.org/2001/XMLSchema"
03. xmlns:tns="http://webservices.amazon.com/AWSECommerceService/2018-06-28"
04. targetNamespace="http://webservices.amazon.com/AWSECommerceService/2018-06-28" >
05.     <xs:sequence>
06.       <xs:element name="IdType" minOccurs="0">
07.         <xs:simpleType>
08.           <xs:restriction base="xs:string">
09.             <xs:enumeration value="ASIN"/>
10.             <xs:enumeration value="UPC"/>
11.             <xs:enumeration value="SKU"/>
12.             <xs:enumeration value="EAN"/>
13.             <xs:enumeration value="ISBN"/>
14.           </xs:restriction>
15.         </xs:simpleType>
16.       </xs:element>
17.       <xs:element name="ItemId" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
18.       <xs:element name="SearchDate" type="xs:date" minOccurs="1"/>
19.       <xs:element name="IncludeReviews" type="xs:boolean" minOccurs="0"/>
20.       <xs:element name="TruncateReviewsAt" type="xs:nonNegativeInteger" minOccurs="0"/>
21.     </xs:sequence>
22.  </xs:complexType>
```

a) [0,8v] Para que serve um XML Schema como este?

b) [0,8v] Construa um documento XML que seja bem-formado e válido de acordo com o Schema.

```
<itemLookupRequest>
  <IdType>ISBN</IdType>
  <ItemId>123</ItemId>
  <ItemId>456</ItemId>
  <SearchDate>2018-07-03</SearchDate>
  <IncludeReviews>false</IncludeReviews>
</itemLookupRequest>

<!—TruncateReviewsAt não aparece porque é opcional -->
```

c) [0,8v] Qual seria a ambiguidade introduzida nos documentos XML pela eliminação dos atributos *type*? Dê um exemplo concreto com um elemento de um documento.

d) [0,8v] Considera o XML um formato de representação de dados:
   i) Binário, implícito
   ii) Binário, explícito
   iii) Textual, implícito
   iv) Textual, explícito

iv

2) O XML Schema anterior faz parte de um WSDL que adiciona as seguintes definições, entre outras:

```
1311.    <message name="ItemLookupRequestMsg">
1312.      <part name="body" element="tns:ItemLookupRequest"/>
1313.    </message>

1364.    <operation name="ItemLookup">
1365.      <input message="tns:ItemLookupRequestMsg"/>
1366.      <output message="tns:ItemLookupResponseMsg"/>
1367.    </operation>

1481.    <service name="AWSECommerceService">
1482.      <port name="AWSECommerceServicePort" binding="tns:AWSECommerceServiceBinding">
1483.      <soap:address location="https://webservices.amazon.com/onca/soap?Service=AWSECommerce"/>
1484.    </port>
```

a) [0,8v] Destes três elementos, qual ou quais pertencem à interface abstrata do WSDL? Justifique.

Message e *Operation* pertencem à interface abstrata pois não descrevem nada de concreto

quanto aos protocolos, endereços e codificações a serem utilizadas. Estas definições são feitas no

binding (não apresentado acima) e no *service*.

b) [0,7v] A linha 1483 faz referência a SOAP. Qual é o papel do SOAP nos Web Services?

c) [0,8v] Considere a componente "webservices.amazon.com" retirada do URL da linha 1483. É, por si só, um nome global ou local? Justifique.

d) [0,6v] O UDDI seria um servidor adequado para armazenar um mapa com a localização da sede da empresa que presta o serviço? Justifique.

e) [0,8v] Nas ferramentas de programação Java para Web Services, um *Handler* pode:
   i) Adicionar ou remover elementos nos cabeçalhos da mensagem.
   ii) Adicionar ou remover elementos no corpo da mensagem.
   iii) i e ii
   iv) iii e modificar o endereço para onde vai ser enviada a mensagem.

iii