departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Concorrência e Paralelismo — 2º teste — 2013-12-13

Número: _____ Nome: _____

1 (a)	1 (b)	2	3 (a)	3 (b)	3 (c)	4	5 (a)	5 (b)	5 (c)	5 (d)
1 val	*2 val*	*2 val*	*1 val*	*2 val*	*2 val*	*2 val*	*2 val*	*2 val*	*2 val*	*2 val*

1. Em baixo apresenta-se um trecho do código de uma classe StringBuffer.

```
1.  class MyStringBuffer {
2.      private int size;
3.      private char[] value;
4.      ...
5.      synchronized void myAppend(MyStringBuffer msb) {
6.          int len = msb.length();
7.          if(this.size + len > this.value.length)
 8.             this.expand(...);  // expand 'this.value' to be at least 'this.size + len'
9.          msb.myAppendTo(this.value,this.size);
10.     }
11.     synchronized void myAppendTo(char[] buf, int pos) {
12.         int len = this.length();
13.         for (int i=0; i<len; i++)
14.             buf[pos+i]= this.value[i];
15.     }
16. }
```

a) [1 val] Para cada um dos tipos de erro identificados de seguida, identifique quais os que podem ocorrer no caso de haver invocação concorrente de métodos definidos na classe MyStringBuffer. Coloque a letra 'S' para indicar os erros que podem ocorrer e depois coloque a letra 'N' nos restantes.

[ ] Low-level data race
[ ] High-level data race
[ ] Stale-value error
[ ] Livelock
[ ] Deadlock

b) [2 val] Considere que tem N *threads* (t_1, …, t_n) que estão a executar em concorrentemente. Descreva como pode ocorrer um dos erros (à sua escolha) que marcou com 'S'. Indique quais os *threads* e objetos envolvidos, quais os métodos invocados e com que argumentos. Explique também o entrelaçamento em causa utilizando os números de linha para referenciar o código.

Threads: _____ Objetos: _____ Métodos + argumentos: _____

Entrelaçamento:

2) [2 val] Uma forma de melhorar o desempenho de um sistema de armazenamento em disco é de utilizar uma técnica de *disk striping*, em que os dados são repartidos rotativamente por *n* discos. Para aceder a um conjunto contínuo de *n* blocos de dados, utilizando *disk striping* com *n* discos, permite reduzir os custos de transferência por um factor de *n*. Por exemplo, o custo de transferência para um sistema de *disk striping* com 5 discos é 1/5 do custo de transferência para um único disco. Assumindo que o custo de transferência de dados representa 40% do tempo total de acesso aos dados, qual é o *speedup* obtido quando se utiliza um *disk array* com 8 discos?

3. Considere o algoritmo TL2 utilizado para implementar sistemas de suporte (*run-time*) para memória transacional. Sempre que realiza uma leitura ou uma escrita transacional na memória, o algoritmo TL2 adiciona informação ao *read-set* ou ao *write-set* da transação.

a) [1 val] Que informação é guardada no *read-set* e no *write-set*? Dê um exemplo para cada.

read-set:

write-set:

b) [2 val] Em que contexto(s) (quando e porquê) é que o algoritmo faz uso do *read-set*?

c) [2 val] Em que contexto(s) (quando e porquê) é que o algoritmo faz uso do *write-set*?

4. [2 val] Considere o seguinte programa Java.

```
1.  public class WaNot{
2.  int i=0;
3.  public static void main(String argv[]){
4.      WaNot w = new WaNot();
5.      w.amethod();
6.  }
7.  public void amethod(){
8.      while(true){
9.          try{
10.             wait();
11.         } catch (InterruptedException e) {}
12.         i++;
13.     }//End of while
14. }//End of amethod
15. }//End of class
```

Qual dos seguintes comportamentos se aplica ao programa acima:

[ ] Dá erro na compilação.	Justificação:
[ ] Compila e corre, mas fica num ciclo infinito no ciclo "while".	
[ ] Compila, corre e termina normalmente.	
[ ] Compila, corre e termina com a exceção "IllegalMonitorStateException".	

5. Considere que um repositório de artigos, onde cada artigo é contém os seguintes campos: título, lista de autores, lista de keywords, abstract, url para o ficheiro PDF ou DOC. À semelhança do que aconteceu no 2º projeto, assuma que lhe é disponibilizada uma classe "Article" que representa um artigo novo de cada vez que é instanciada. Pretende-se criar uma topologia Storm para processar o *stream* de artigos para periodicamente, listar as 5 *keywords* mais populares até ao momento e, para cada uma dessas *keywords*, listar os 3 autores que mais as usam nos seus artigos.

a) [2 val] Apresente uma proposta de topologia Storm para realizar o objetivo pretendido. Identifique os *spout* com S_1, S_2, ..., e os *bolt* com B_1, B_2, ... Nas arestas que ligam os componentes, indique que informação flui entre cada dois componentes da topologia.

b) [2 val] Apresente uma breve descrição (2 linhas max) da funcionalidade de cada componente da topologia proposta na alínea anterior.

c) [2 val] Apresente o código dos métodos nextTuple() e declareOutputFields () do *spout*.

```
public class ArticleSampleSpout extends BaseRichSpout {
    SpoutOutputCollector _collector;
    [...]
    @Override
    public void nextTuple () {









    }
    [...]
    @Override
    public void declareOutputFields (OutputFieldsDeclarer declarer) {


    }
}
```

d) [2 val] Apresente o código dos métodos execute() e declareOutputFields () de um *bolt* que suceda ao *spout* da alínea anterior.

```
public static class WordCount extends BaseBasicBolt {

    [...]
    @Override
    public void execute(Tuple tuple, BasicOutputCollector collector) {









    }

    @Override
    public void declareOutputFields(OutputFieldsDeclarer declarer) {


    }
}
```