



Grupo 1. [Lógica de Hoare; 2,5 valores]

- a) A lógica de Hoare utiliza triplos para raciocinar sobre a correcção de programas. Explique o significado informal de um triplo da forma $\{\varphi\} P \{\psi\}$.
b) Explique por palavras suas porque é que o triplo

$$\{(n \geq 0) \wedge (n^2 > 28)\} m := n + 1; m := m * m \{\neg(m = 36)\}$$

está correcto.

- c) Utilizando a lógica de Hoare mostre que o triplo está correcto.
d) Considere a seguinte variante da regra para a instrução condicional.

$$\frac{\{\varphi \wedge p\} S_1 \{\psi_1\} \quad \{\varphi \wedge \neg p\} S_2 \{\psi_2\}}{\{\varphi\} \text{if } p \text{ then } S_1 \text{ else } S_2 \{\psi_1 \wedge \psi_2\}}$$

A regra permite inferir apenas asserções correctas? Se sim explique porquê; senão dê um contra-exemplo. Lembre-se que $\psi_1 \wedge \psi_2$ implica ψ_1 , mas que o inverso não é verdade.

Grupo 2. [Java Modeling Language; 2,5 valores]

- a) Considere o seguinte excerto de uma classe Java com anotações JML.

```
public class ConjuntoOrdenado {  
    private int dimensão;  
    //@ requires dimensão > 0;  
    public int mínimo () {...}  
    ...  
}
```

Estará este código correcto para o JML? (Por outras palavras, compila em JML?) Se sim explique porquê; senão mostre como resolver o problema.

- b) Escreva uma especificação JML para um método Java que implemente a seguinte a função h .

$$h(n) = \begin{cases} (3n + 1)/2 & \text{se } n > 0 \text{ e } n \text{ é par} \\ n/2 & \text{se } n > 0 \text{ e } n \text{ é ímpar} \end{cases}$$

c) Os automóveis modernos estão todos equipados com fecho centralizado. Neste sistema a porta do condutor comanda todas as outras portas, no sentido em que abrindo a porta do condutor abrem-se todas as outras e fechando a porta do condutor fecham-se todas as outras portas. Cada uma das outras portas abre-se e fecha-se a si própria apenas.

Escreva uma interface Java que descreva um automóvel com fecho centralizado. Assuma que cada porta tem associado um número inteiro 0, 1, 2, ..., e que o número 0 representa a porta do condutor. Considere métodos para trancar e para destrancar uma dada porta e para saber se uma dada porta está trancada ou não. Escreva uma especificação JML para esta interface utilizando variáveis de modelo.

Grupo 3. [ESC/Java; 2,5 valores]

a) ESC/Java é uma ferramenta para encontrar erros num programa. Explique sucintamente como funciona a ferramenta.

b) “ESC/Java não completo”. Explique o significado desta afirmação. Aponte uma fonte de incompletude da ferramenta. Será a ferramenta útil na presença deste problema? Justifique.

c) A ferramenta ESC/Java consegue por vezes provar a terminação de ciclos, mas precisa de ajuda. Utilize uma cláusula **decreases** para ajudar o ESC/Java a provar a terminação deste ciclo. Explique qual a verificação que o ESC/Java tenta efectuar neste caso.

```
for (int i = 0; i < n; i++)
    if (a[i] < m) {
        mindex = i;
        m = a[i];
    }
```

Grupo 4. [Verificação de modelos; 2,5 valores] Considere o seguinte modelo Promela que descreve um sistema composto por três leitores e dois escritores.

```
1 int nr = 0; bool busy = false;
2 active[3] proctype Reader() {
3     do
4         :: !busy ->
5             nr = nr + 1;
6 read: skip;
7         nr = nr - 1;
8     od;
9 }
10 active[2] proctype Writer() {
11     do
12         :: nr = 0 && !busy ->
13             busy = true;
14 write: skip;
15             busy = false;
16     od;
17 }
```

a) Apresente uma fórmula em lógica temporal linear que descreva a propriedade “Sempre que o leitor número um está a ler nenhum escritor está a escrever”.

b) Descreva por palavras suas o significado da fórmula $\Box \Diamond (\text{Writer}[1] @ \text{write} \vee \text{Writer}[2] @ \text{write})$

c) Será esta uma propriedade de segurança ou de animação? Justifique.

d) A ferramenta Spin pode ser utilizada em modo de simulação e em modo de verificação. Explique a diferença. Algum dos modos garante a correcção do modelo face à especificação? Justifique.

e) Descreva sucintamente como procederia para verificar, com a ferramenta Spin, se o modelo acima satisfaz a propriedade da alínea b).



Grupo 1. [Lógica de Hoare; 2,5 valores]

a) Considere o seguinte programa.

```
while x < 0 do x := x + 1
```

Suponha que pretende verificar que o programa deixa o valor 0 na variável x. Escreva uma pré e uma pos condição adequadas.

b) Mostre que o triplo obtido na alínea anterior está correcto.

c) Estará o seguinte triplo correcto? Justifique.

```
{true} while true do z := 5 {z = 9}
```

d) O resultado acima assegura que o programa deixa o valor 9 na variável z. Como explica a aparente contradição?

Grupo 2. [Java Modeling Language; 2,5 valores]

a) Considere o seguinte excerto de uma classe Java com anotações JML.

```
//@ requires n >= 0;  
//@ ensures \result * \result == n;  
static int sqrt (int n) {return n == 4 ? 2 : 0;}  
public static void main (String[] a) {sqrt(0);}
```

a) Explique o que acontece quando compilar o programa acima com `jmlc`; justifique. Se achar a compilação traz problemas sugira alterações ao programa de modo a que o programa compile sem problemas.

b) Explique o que acontece quando correr com `jmlrac` o programa acima (eventualmente alterado de acordo com a alínea anterior). Justifique.

c) Considere o seguinte método extraído de uma classe `ContaÀOrdem` que descreve uma conta bancária à ordem tradicional.

```
//@ requires amount > 0;  
//@ requires balance() >= amount;  
//@ ensures balance() == \old(balance()) - amount;  
public void withdraw (int amount)
```

Escreva o contrato para o mesmo método desta vez para a classe `ContaOrdenado`, subclasse de `ContaÀOrdem`, e que permite levantamentos que deixam o saldo negativo até um montante pré-determinado (geralmente o valor do ordenado). Junte todos os métodos que achar necessário para completar o seu exercício.

d) Imagine agora que a classe `ContaOrdenado` é uma subclasse directa da classe `Object` (em vez de herdar de `ContaÀOrdem`). Reescreva o contrato para o método acima.

e) Usamos baralhos de cartas em mais jogos do que imaginamos. Para este exercício estamos interessados em baralhos cujas cartas sejam identificadas por um número natural. Por exemplo se decidirmos que cada naipe está associado a um número entre 0 e 3, e que Paus é o naipe dois, então um Nove de Paus pode ser representado pela carta número $2 \times 13 + 9 = 35$. Para o efeito vamos assumir um método `dimensão()` que nos dá o número de cartas num baralho novo; cada carta é então representada por um número entre um inclusivé e `dimensão()` inclusivé.

Entre as várias operações do baralho considere: i) um método para obter uma carta do baralho, sem a retirar (por exemplo, para escolher o trunfo) `int` `umaCarta()`, ii) um método para retirar uma carta do baralho `int` `retirarUma()`, iii) outro para devolver ao baralho uma carta retirada anteriormente `void` `devolverCarta(int c)`, iv) um método para devolver uma série de cartas ao baralho `void` `devolverCartas(int[] cs)`.

Escreva uma interface Java que descreva um baralho. Escreva um especificação JML para esta interface utilizando variáveis de modelo. Junte mais métodos se achar adequado.

Grupo 3. [ESC/Java; 2,5 valores]

a) A ferramenta ESC/Java não é completa. Explique o significado desta afirmação; identifique uma fonte de incompletude.

b) Explique como funciona a ferramenta, utilizando para o efeito o seguinte método.

```
//@ requires x > y;
//@ ensures y < x;
void m () {y = -y;}
```

c) Considere o seguinte programa.

```
public class MultiConjunto {
    //@ invariant (\forall int i;
    //@           n <= i & i < a.length; a[i] == null);
    Object [] a; int n;
    MultiConjunto(int l) {
        n = 0;
        a = new Object[l];
    }
    void adicionar(Object o) {
        if (n == a.length)
            fazerCrescer();
        a[n++] = o;
    }
    private void fazerCrescer () {
        Object [] novo = new Object [a.length * 2];
    }
}
```

```

    for (int i = 0; i < a.length; i++)           16
        novo[i] = a[i];                          17
    a = novo;                                     18
}                                                  19
}                                                  20

```

Correu-se o programa abaixo com a ferramenta ESC/Java; parte do *output* está indicado abaixo. Indique como tentaria anular cada um dos avisos.

- ```

1. MultiConjunto.java:10: Warning: Possible attempt to
 allocate array of negative length (NegSize)
 a = new Object[1];
 ^
2. MultiConjunto.java:13: Warning: Possible null
 dereference (Null)
 if (n == a.length)
 ^
3. MultiConjunto.java:15: Warning: Possible negative array
 index (IndexNegative)
 a[n++] = o;
 ^
4. MultiConjunto.java:15: Warning: Array index possibly too
 large (IndexTooBig)
 a[n++] = o;
 ^
5. MultiConjunto.java:15: Warning: Type of right-hand side
 possibly not a subtype of array element type (ArrayStore)
 a[n++] = o;
 ^
6. MultiConjunto.java:18: Warning: Possible null
 dereference (Null)
 Object[] novo = new Object[a.length * 2];
 ^

```

**Grupo 4.** [Verificação de modelos; 2,5 valores] Considere o seguinte modelo Promela que descreve um sistema composto por três processos: um produtor, um meio de comunicação e um consumidor

```

chan sch = [0] of { byte };
chan cch = [0] of { byte };
int smsg, rmmsg;
active proctype sender() {
 smsg = 0;
 do
 :: smsg = (smsg + 1) % 4; /* produce */
 sch ! smsg;
 od
}

```

```

active proctype medium() {
 int msg;
 do
 :: sch ? msg ->
 if
 :: true -> cch ! msg;
 :: true -> skip;
 fi
 od
}
active proctype receiver() {
 do
 :: cch ? rmsg ->
 printf("%d", rmsg); /* consume */
 od
}

```

- a) O que modela o processo `medium`? Justifique.
- b) Apresente uma fórmula em lógica temporal que traduza a propriedade “sempre que for produzida, a mensagem número 3 será entregue, mais cedo ou mais tarde”. Note que todas as mensagens 3 produzidas por `sender` são iguais e portanto não distinguíveis.
- c) Relembrando o significado de computação com justiça fraca: “se uma instrução está sempre executável, então será eventualmente executada como parte da computação”, indique justificando se espera que o modelo verifique a propriedade da alínea **b**).
- d) Descreva sucintamente como procederia para verificar, com a ferramenta Spin, se o modelo acima satisfaz a propriedade da alínea **b**).



**Grupo 1.** [Hoare Logic; 2.5 valores]

a) The following program computes the larger square smaller than a given positive number  $n$ .

```
z := 0;
while (z + 1) * (z + 1) <= n do
 z := z + 1;
```

Show that  $\{n > 0\} S \{n - 2z \leq z^2 \leq n\}$ , where  $S$  is the program above.

b) Show that the program terminates under the assumption that  $n > 0$ .

c) While loops test their guards at the beginning of the loop. In contrast **repeat-until** loops test the guard at the end. As consequence, the body of a **repeat-until** is executed at least once. Furthermore, the **until** keyword suggests that the loop is exited when the condition becomes false, in contrast with while loops. Then, a program

```
repeat S until p
```

can be understood as

```
S; while !p do S
```

Suggest a rule for the **repeat** command. Justify your choice.

**Grupo 2.** [Java Modeling Language; 2.5 valores]

Consider the two classes below.

```
class A {
 //@ requires i >= 0;
 //@ ensures \result >= 100;
 int m (int i) { ... }
}
class B extends A {
 //@ also
 //@ ensures \result >= 0;
 int m (int i) { ... }
}
```

a) What is the precondition to method `m` in class B?

b) Suggest a possible value for `m(0)` in class B which cannot possibly be a value for the same method in class A.

c) Provide a possible implementation for the two `m()` methods.

d) The *disjoint set union* data structure involves disjoint sets (no pair of sets have elements in common) and the operations `setContaining()` and `union()`. The general scenario for its use is that you are given a set of items. At any given time, the items are partitioned into subsets such that each element belongs to exactly one subset. Each subset is called an *equivalence class*, and the collection of equivalence classes is called a *partition* of the set. From time to time, two equivalence classes are joined to form a new class (the `union()` operation). Since the equivalence classes are changing, the equivalence class of an item can change so the `setContaining()` operation is included to determine the class that currently has a specific element.

Given a Java type `Item` we consider the Java type `Partition` that describes a collection of equivalence classes of items. To refer to the equivalence classes we choose one and only one element of each equivalence class to represent the class. That element is called a *class representative*. Then, method `setContaining(Item e)` returns the class representative of `e`, which is itself an `Item`. This operation is particularly useful to determine whether two items, `e1` and `e2`, are in the same equivalence class: `p.setContaining(e1) == p.setContaining(e2)` where `p` is an object type `Partition`.

Method `union` joins two sets, given any two items (not necessarily the representatives) of two classes. Method `singleton (Item e)` adds a new class to the partition. If we need the set of all elements in the equivalence class of a given `Item e` (not necessarily the representative), we may use method `setOf (Item e)`.

```
interface Partition {
 Item setContaining (Item e);
 void union (Item e1, Item e2);
 void singleton (Item e);
 java.util.Set<Item> setOf (Item e);
}
```

Add a *model-based JML* specification to the above interface; consider more methods if needed; write comprehensive contracts to all your methods.



**Grupo 3.** [ESC/Java; 2.5 valores]

a) Write a short comment on the following statement.

Unlike Hoare Logic, ESC/Java is a tool for finding errors in a program, not for proving statements about programs.

b) The ESC/Java language includes an **assume** *pragma* allowing to add unverified facts to a program. Explain under which conditions should it be used as well as the consequences arising from its misuse.

c) Consider the following Java class.

```
class ObjectStack { 1
 Object [] a; int n; 2
 //@ invariant (\forall int i; 3
 //@ n <= i & i < a.length ==> a[i] == null); 4
 ObjectStack(int l) { 5
 n = 0; 6
 a = new Object[l]; 7
 } 8
 void Pop() { 9
 --n; 10
 a[n] = null; 11
 } 12
 ... 13
} 14
```

The class was submitted to ESC/Java producing an output, part of which is shown below. Describe how would you address each of the four warnings.

1. ObjectStack.java:7: Warning:  
Possible attempt to allocate array of negative length  
a = new Object[l];
2. ObjectStack.java:11: Warning: Possible null dereference  
a[n] = null;
3. ObjectStack.java:11:  
Warning: Possible negative array index  
a[n] = null;
4. ObjectStack.java:21: Warning:  
Array index possibly too large  
a[n] = null;

**Grupo 4.** [Model Checking; 3.5 valores] Consider the below solution to the critical problem using semaphores. The wait operation is **atomic** to ensure that the value of the semaphore is decremented only when positive. The signal operation needs no special implementation because the assignment statement is atomic.

```

byte sem = 1; 1
active proctype P() { 2
 do :: 3
 printf("Non critical section P\n"); 4
 atomic { 5
 sem > 0; 6
 sem-- 7
 } 8
 printf("Critical section P\n"); 9
 sem++; 10
 od 11
} 12
active proctype Q() { 13
 -- replace character 'P' with 'Q' 14
} 15

```

a) Suppose that you suspect that variable `sem` may take a value greater than 1 on line 10 of the above model. Using the Spin tool, explain how would you confirm your suspicion and, in case it becomes true, how would you obtain the computation (the sequence of states) that leads to the situation.

b) Using Spin (either from the command line or via the interactive iSpin tool), detail all the steps necessary to verify that the program fulfills the correctness property of mutual exclusion: at most one process is executing its critical section at any time. Do not forget to explain how to check that the property holds or not.

c) Same for the absence of starvation: if any process is trying to execute its critical section, then eventually that process is successful.

d) Spin treats safety properties differently from liveness properties. For safety properties, the tool looks for a counterexample consisting of one state where the formula is false. How does the tool behave with respect to liveness properties? Why?



# Software Fiável

Mestrado em Engenharia Informática e Mestrado em Informática

## Exame I

17 de Janeiro de 2014

**Duração** 3h

**20 valores**

### GRUPO I

**[(0.5+2.5)+1]**

1. Considere o programa  $P$  apresentado abaixo.

```
sum = 1;
i = 1;
while i < n do
 sum = sum + 2*i + 1;
 i = i + 1;
```

- (a) Explique o que significa dizer que  $P$  é parcialmente correto relativamente à condição inicial  $\{n \geq 1\}$  e à condição final  $\{sum = n^2\}$
- (b) Recorrendo ao cálculo de Hoare, prove a correção parcial de  $P$  enunciada em (a).

2. Suponha que a linguagem de programas é estendida com o comando repetitivo

`repeat S until p;` equivalente a `S; while  $\neg p$  do S;`

Deduz a regra do cálculo de Hoare para este novo tipo de comandos.

### GRUPO II

**[(3+1.5)+ (1+1+1+0.5+1)]**

1. Considere a especificação algébrica abaixo e o refinamento para o tipo Java `JoinableList<E>`.

```
specification List[Element]
 sorts
 List[Element]

 constructors
 make: --> List[Element];
 addLast: List[Element] Element --> List[Element];

 observers
 size: List[Element] --> int;
 get: List[Element] int -->? Element;

 others
 concat: List[Element] List[Element] --> List[Element];

 domains
 L: List[Element]; I: int;
 get (L, I) if 0 <= I and I < size(L);

 axioms
 L, L2: List[Element]; I: int; E: Element;

 size (make ()) = 0;
 size (addLast (L, E)) = 1 + size (L);

 get (addLast (L, E), I) = E when I = size(L)
 else get (L, I);

 concat (L, make ()) = L;
 concat (L2, addLast (L, E)) = addLast(concat(L2, L), E);

end specification

refinement<E>
 Element is E

 List[Element] is JoinableList<E> {
 make: --> List[Element]
 is
 JoinableList();

 addLast: List[Element] e:Element --> List[Element]
 is
 JoinableList<E> make(E e);

 concat: List[Element] l:List[Element] --> List[Element]
 is
 JoinableList<E> make(JoinableList<E> l);

 size: List[Element] --> int
 is
 int size();

 get: List[Element] i:int -->? Element
 is
 E get(int i);
 }
end refinement
```

- (a) Apresente a correspondente especificação baseada em propriedades em JML tão completa quanto possível para o tipo `ObjectJoinableList`, uma versão não genérica de `JoinableList` definida como se mostra abaixo. Deve especificar também as propriedades do tipo `ObjectJoinableList` que interessam aos seus clientes e que, por falta de poder expressivo, não foram especificadas na abordagem algébrica.

```
/* Immutable; implements a list where it is combine them; null values are NOT allowed. */
public final class ObjectJoinableList {
 /* The number of items stored in this list.
 * Invariant: count == (pre!=null ? pre.count : 0) +
 * (item!=null ? 1 : 0) + (post!=null ? post.count : 0) */
 private final int count;

 /* The list of items before "this.item"; may be null. */
 private final ObjectJoinableList pre;

 /* The list of items after "this.item"; may be null. */
 private final ObjectJoinableList post;

 /* If nonnull, it stores an item. */
 private final Object item;

 public ObjectJoinableList() { ... }

 public ObjectJoinableList make(ObjectJoinableList that) { ... }

 public ObjectJoinableList make(Object newItem) { ... }

 public Object get(int i) { ... }

 public int size() { ... }
}
```

- (b) Especifique formalmente, também em JML, a informação veiculada no comentário assinalado e explique que vantagens isso pode trazer.

## 2. Sujeitou-se a classe `Sqrt` à ferramenta Esc/Java2.

- (a) De acordo com o output da ferramenta, mostrado abaixo, explique em que consiste cada um dos problemas encontrados e a razão por que ocorrem.
- (b) Sem alterar a especificação, sugira uma forma de eliminar o problema específico que é reportado pelo Esc/Java2 na linha 13.
- (c) Suponha que com a sugestão fornecida em (b) o Esc/Java2 deixa de assinalar qualquer problema no método `intSqrt`. Indique, justificando, se isso permite concluir que o código é correto face à especificação.
- (d) A especificação JML dada endereça apenas a correção parcial do método `intSqrt`. Indique como é possível em JML também endereçar a questão da terminação.
- (e) Explique o que acontece se executarmos a classe `Sqrt` com o JMLrac (não se limite a indicar o `output`, explique também o processo).

```
1 public class Sqrt {
2
3 /*@
4 @ requires n >= 0;
5 @ ensures \result * \result <= n;
6 @ ensures n < (\result + 1) * (\result + 1);
7 @*/
8 public static int intSqrt (final int n){
9 int act = 0;
10 while ((act+1)*(act+1) <= n)
11 act++;
12 return act;
13 }
14
15
16 /**
17 * @param args
18 */
19 public static void main(String[] args) {
20 System.out.println(intSqrt(-10));
21 System.out.println(intSqrt(10));
22 }
23
24 }
```

```

/Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java:13: Warning: Postcondition possibly not established (Post)
 }
 ^
MARKER /Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java 13 Warning: Postcondition possibly not established (Post)
Associated declaration is "/Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java", line 5, col 3:
 @ ensures \result * \result <= n;
 ^
Adding extra marker info: /Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java 5 51
Suggestion [13,1]: none
Execution trace information:
 Reached top of loop after 0 iterations in "/Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java", line 10, col 2.
 Executed return in "/Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java", line 12, col 2.

/Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java:20: Warning: Precondition possibly not established (Pre)
 System.out.println(intSqrt(-10));
 ^
MARKER /Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java 20 Warning: Precondition possibly not established (Pre)
Associated declaration is "/Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java", line 4, col 3:
 @ requires n >= 0;
 ^
Adding extra marker info: /Users/antoniolopes/Documents/workspaceMobius/ExpAndre/src/Sqrt.java 4 31
Suggestion [20,28]: none
```

**GRUPO III****[(1.75+1+2+1)+1.25]**

- I. Considere o modelo Promela fornecido, o qual implementa o crivo de Eratóstenes — um algoritmo para encontrar números primos até um certo valor.

- (a) Indique uma possível execução deste modelo alterado de forma a que  $N$  seja 5.
- (b) Procedeu-se a uma análise básica do sistema, tal como mostrado abaixo e a uma simulação do *trail* resultante. Interprete os resultados obtidos e indique como, face à natureza do problema, este poderia ser eliminado.
- (c) Indique como procederia para verificar com o Spin (na consola ou através de um dos seus interfaces gráficos) se o sistema tem as seguintes propriedades:

*Mais tarde ou mais cedo o valor de value do Generator será sempre igual a  $N+1$*

*Até executar a primeira leitura de inp na linha 24, o valor de value de um processo Sieve será sempre igual a 0*

- (d) Suponha que no caso da verificação da primeira propriedade o Spin reporta que explorou o espaço todo e não encontrou nenhum erro. Indique justificando se é possível concluir que não existe mesmo nenhuma execução do sistema onde a propriedade é violada.

2. Indique, justificando, as afirmações que são falsas:

- (a) O Uppaal é um *model checker* aplicável a modelos definidos por autómatos que incorporam relógios os quais permitem modelar sistemas com requisitos de tempo real.
- (b) O Bandera é um *model checker* aplicável diretamente a programas Java.
- (c) O AlloyAnalyzer é um *model checker* aplicável a modelos escritos em Alloy.
- (d) O SLAM implementa técnicas automáticas de abstração e refinamento guiada por contraexemplos, os quais são encontrados através de *model checking*, e aplica-se diretamente a código (por exemplo, escrito em C).
- (e) O Jennisys utiliza *model checking* para sintetizar programas numa linguagem OO a partir de especificações.

```

1 #define N 10
2
3 mtype = { nextValue };
4
5 proctype Generator (int max){
6 chan next = [1] of {mtype, int};
7 int value = 2;
8 run Sieve (next);
9 do
10 :: (value <= max) ->
11 next!nextValue(value);
12 value = value + 1
13 :: (value > max) ->
14 break
15 od
16 }
17
18 proctype Sieve (chan inp){
19 chan next = [1] of {mtype, int};
20 int value, myPrime;
21 inp?nextValue(myPrime);
22 run Sieve (next);
23 do
24 :: inp?nextValue(value) ->
25 if
26 :: (value % myPrime) != 0 ->
27 next!nextValue(value)
28 :: else ->
29 skip
30 fi
31 od
32 }
33
34 init{
35 run Generator (N)
36 }

```

```

spin -a sieve.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w
-o pan pan.c
./pan -m10000
Pid: 6632
pan:1: invalid end state (at depth 32)
pan: wrote sieve.pml.trail

```

```

(Spin Version 6.2.5 -- 3 May 2013)
Warning: Search not completed
+ Partial Order Reduction

```

```

Full statespace search for:
 never claim - (not selected)
 assertion violations +
 cycle checks - (disabled by -DSAFETY)
 invalid end states +

```

State-vector 204 byte, depth reached 33, errors: 1

```

Starting Generator with pid 1
1: proc 0 (:init:) sieve.pml:35 (state 1)
 [(run Generator(5))]
Starting Sieve with pid 2
2: proc 1 (Generator) sieve.pml:8 (state 1)
 [(run Sieve(next))]
....

```

```

spin: trail ends after 33 steps
#processes: 6
33: proc 5 (Sieve) sieve.pml:21 (state 1)
33: proc 4 (Sieve) sieve.pml:23 (state 10)
33: proc 3 (Sieve) sieve.pml:23 (state 10)
33: proc 2 (Sieve) sieve.pml:23 (state 10)
33: proc 1 (Generator) sieve.pml:16 (state 10)
33: proc 0 (:init:) sieve.pml:36 (state 2)
6 processes created
Exit-Status 0

```



# Software Fiável

Mestrado em Engenharia Informática e Mestrado em Informática

## Exame 2

3 de Fevereiro de 2014

Duração 3h

20 valores

### GRUPO I

[3+1]

1. Considere o programa  $P$  apresentado abaixo.

```
act = 1;
ant = 0;
i = 0;
while i < n do
 act = act + ant;
 ant = act - ant;
 i = i + 1;
```

Recorrendo ao cálculo de Hoare, prove a correção parcial de  $P$  relativamente à condição inicial  $\{n \geq 0\}$  e à condição final  $\{ant = \text{Fib}(n)\}$ , onde  $\text{Fib}(n)$  é o  $n$ -ésimo número de Fibonacci, uma sucessão definida recursivamente da seguinte forma:  $\text{Fib}(0)=0$ ;  $\text{Fib}(1)=1$ ;  $\text{Fib}(n)=\text{Fib}(n-1)+\text{Fib}(n-2)$  se  $n \geq 2$ .

2. Suponha que a linguagem de programas é estendida com o comando repetitivo

do  $S$  while  $p$ ;      equivalente a       $S$ ; while  $p$  do  $S$ ;

Deduz a regra do cálculo de Hoare para este novo tipo de comandos.

### GRUPO II

[(2.5+1)+(2+1.5+1)]

```
/**
 * A polynomial in one variable (i.e., a univariate polynomial)
 * with constant integer coefficients.
 */
specification
 sorts
 Polynomial
 constructors
 make: --> Polynomial;
 addMonomial: Polynomial int int --> Polynomial;
 others
 coefficient: Polynomial int --> int;
 sum: Polynomial Polynomial --> Polynomial;
 order: Polynomial --> int;
 domains
 P: Polynomial; A: int; N: int;
 addMonomial (P, A, N) if N >= 0;
 coefficient (P, N) if N >= 0;
 axioms
 P, Q: Polynomial; A, A1, A2: int; N, N1, N2: int;

 coefficient (make (), N) = 0;
 coefficient (addMonomial (P, A, N1), N2) =
 A + coefficient (P, N1) when N1 = N2
 else coefficient (P, N2);

 sum (make (), P) = P;
 sum (addMonomial (Q, A, N), P) =
 addMonomial (sum (Q, P), A, N);

 addMonomial (addMonomial (P, A1, N1), A2, N2) =
 addMonomial (P, A1 + A2, N1) when N1 = N2
 else addMonomial (addMonomial (P, A1, N1), A2, N2);

 order (make ()) = 0;
 order (P) >= N if coefficient (P, N) != 0;
 coefficient (P, N) != 0 if N = order (P) and N > 0;
end specification
```

```
refinement
 Polynomial is Poly {
 make: --> Polynomial
 is Poly();
 addMonomial: Polynomial a:int n:int --> Polynomial
 is Poly addMonomial(int a, int n);
 coefficient: Polynomial n:int --> int
 is int coefficient(int n);
 order: Polynomial --> int
 is int order();
 sum: Polynomial p:Polynomial --> Polynomial
 is Poly sum(Poly p);
 }
end refinement
```

1. Considere a especificação algébrica dada e o refinamento para o tipo Java Poly.

- (a) Apresente a correspondente especificação baseada em propriedades em JML tão completa quanto possível para o tipo Poly (destinada a clientes da classe). Indique quais as propriedades especificadas algebricamente não conseguiu especificar em JML e quais as propriedades que especificou em JML que não estão cobertas pela especificação algébrica.
- (b) Tendo em conta os detalhes da implementação revelados, que invariantes JML seria útil especificar?

```
/** Immutable polynomials with integer coefficients.*/
public class Poly {

 // array with coefs of the poly for each i <= degree()
 private int coefs [];

 /** Creates the polynomial zero.
 */
 public Poly () { coefs = new int [1] }

 /** Creates a new polynomial: c.x^n.
 * @param c The coefficient
 * @param n The exponent
 */
 public Poly (int c, int n) { coefs = new int [n+1]; coefs[c] = n; }

 /**
 * @return the degree of the polynomial (the largest exponent).
 */
 public int degree() {
 int i = coefs.length-1;
 while (i>0 && coefs[i]==0)
 i--;
 return i;
 }

 /**
 * Returns the coefficient of the term whose exponent is d
 * (0 if it does not exist).
 * @param d The exponent
 * @return the coefficient of the term whose exponent is d
 */
 public int coeff(int d) { ... }

 /** Returns this+c.x^n
 * @param c The coeff of the monomial
 * @param n The degree of the monomial
 * @return this+c.x^n.
 */
 public Poly addMonomial(int c, int n) {...}

 /** Returns this+other
 * @param other The other polynomial
 * @return this+other.
 */
 public Poly add(Poly other) {
 int max = degree() > other.degree() ? degree() : other.degree();
 result.coefs = new int [max + 1];
 ...
 }
}
```

2. Sujeitou-se a classe *InsertionSort* à ferramenta Esc/Java2.

- (a) De acordo com o output da ferramenta, mostrado abaixo, explique em que consiste cada um dos problemas assinalados, a razão por que ocorrem e que forma poderiam ser resolvidos.
- (b) Explique o que teria de fazer para, recorrendo primeiro ao **Esc/Java2** e depois ao **Jmlrac**, poder ganhar confiança de que o método *sort* executa corretamente a ordenação *inplace* do vector que lhe é passado como argumento.
- (c) Indique as razões pelas quais nem o Esc/Java2 nem o Jmlrac podem dar garantias de correção do método.

```
1 public class InsertionSort {
2 static class Node {
3 public int key;
4 public Object value;
5 }
6 public Node(int key, Object value) {
7 this.key = key; this.value = value;
8 }
9 }
10
11 public static void sort(Node[] arr) {
12 for(int i = 1; i <= arr.length; i++) {
13 for(int j = i; j >= 0; j--) {
14 if (arr[j].key >= arr[j-1].key) {
15 Node tmp = arr[j];
16 arr[j] = arr[j-1];
17 arr[j-1] = tmp;
18 }
19 }
20 }
21 }
22
23 /**
24 * @param args
25 */
26 public static void main(String[] args) {
27 sort(new Node[] {new Node(23, "Jose"), new Node(18, "Maria")});
28 }
29 }
```

```

InsertionSort: sort(InsertionSort$Node[]) ...

/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java:13: Warning: Possible null dereference (Null)
 for(int i = 1; i <= arr.length; i++) {
 ^
MARKER /Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java 13 Warning: Possible null dereference (Null)
Suggestion [13,25]: perhaps declare parameter 'arr' at 12,32 in /Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java with 'non_null'
Execution trace information:
 Reached top of loop after 0 iterations in "/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java", line 13, col 2.
 Reached top of loop after 0 iterations in "/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java", line 14, col 3.

/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java:15: Warning: Array index possibly too large (IndexTooBig)
 if (arr[j].key >= arr[j-1].key) {
 ^
MARKER /Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java 15 Warning: Array index possibly too large (IndexTooBig)
Suggestion [15,11]: none
Execution trace information:
 Reached top of loop after 0 iterations in "/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java", line 13, col 2.
 Reached top of loop after 0 iterations in "/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java", line 14, col 3.

/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java:15: Warning: Possible null dereference (Null)
 if (arr[j].key >= arr[j-1].key) {
 ^
MARKER /Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java 15 Warning: Possible null dereference (Null)
Suggestion [15,14]: none <intimidating expression>
Execution trace information:
 Reached top of loop after 0 iterations in "/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java", line 13, col 2.
 Reached top of loop after 0 iterations in "/Users/antonialopes/Documents/workspaceMobius/ExpAndre/src/InsertionSort.java", line 14, col 3.

```

### GRUPO III

[(2+2.5+1+1.25)+1.25]

#### I. Considere o modelo Promela fornecido.

(a) Indique uma possível execução deste modelo e explique sumariamente o que faz o sistema modelado.

(b) Indique como procederia para verificar com o Spin (na consola ou através de um dos seus interfaces gráficos) se o sistema tem as seguintes propriedades:

*O valor escrito em result pelo consumer é verdadeiro sse todos os elementos de str são iguais.*

*Se C é o i-ésimo comparador a ser criado então, depois de executar a primeira leitura de inC, o valor da sua variável first é sempre str[i-1]*

*Se C é o i-ésimo comparador a ser criado então, mais tarde um mais cedo o valor da sua variável y é str[i]*

(c) Para cada uma das propriedades anteriores, indique, justificando, se se trata de uma propriedade de segurança (safety), de animação (liveness) ou mista.

(d) Explique sucintamente de que forma o Spin procede para verificar um assert embebido no modelo.

2. Indique, justificando, as afirmações que são falsas:

(a) A técnica de verificação de um OS kernel que foi alvo de uma das apresentações baseava-se em model checking.

(b) O PRISM é um model checker aplicável a modelos onde se podem associar probabilidades às transições de estado.

(c) O Modex é uma ferramenta que extrai modelos Promela a partir de código C.

(d) O CodeContracts permite apenas a verificação de contratos em tempo de execução.

```

1 #define MAX 3
2 #define MAXINDEX MAX-1
3
4 bool result;
5 byte str[MAX];
6
7 proctype producer(chan outC){
8 int i;
9 do
10 :: (i <= MAXINDEX) -> outC!str[i]; i++;
11 :: (i > MAXINDEX) -> break;
12 od;
13 }
14
15 proctype comparator(chan inC; chan outC){
16 bit first,y;
17 bool r;
18 chan child = [1] of { bit };
19 chan c = [1] of { bool };
20
21 outC!1;
22 if
23 :: inC?first -> outC!true;
24 run comparator(child, c);
25 do
26 :: inC?y -> child!y; c?r; outC!(r && (first==y));
27 :: timeout -> break;
28 od
29 :: timeout -> skip
30 fi
31 }
32
33 proctype consumer(chan inC){
34 bool r;
35 int i=0;
36 do
37 :: (i <= MAX) -> inC?r; printf("received %d",r);i++;
38 :: (i > MAX) -> break;
39 od;
40 result = r;
41 }
42
43 init
44 {
45 chan child = [1] of { bit };
46 chan c = [1] of { bool };
47
48 str[0] = 1;
49 str[1] = 0;
50 str[2] = 1;
51
52 run producer(child);
53 run consumer(c);
54 run comparator(child, c);
55 }

```



- 
- (e)**    ☐ Rubicon permite fazer a verificação de propriedades de programas Ruby escritas numa linguagem baseada no RSpec (uma plataforma de teste para Ruby).



# Software Fiável

Mestrado em Engenharia Informática e Mestrado em Informática

## Exame I

16 de janeiro de 2015

**Duração** 3h

**20 valores**

### GRUPO I

**[(2+1)+1]**

1. Considere o programa  $P$  apresentado abaixo.

```
i := n;
while i >= 2 do
 i := i - 2;
b := (i==0);
```

(a) Recorrendo ao cálculo de Hoare, prove a correção parcial de  $P$  relativamente a

$\{ n \geq 1 \} P \{ b = n \text{ é par} \}.$

(b) Para provar a terminação de  $P$  quando é executado num estado inicial satisfazendo  $n \geq 1$  o que é preciso fazer mais?

2. Suponha que a linguagem de programas é estendida com a seguinte atribuição condicional

$x := p ? e1 : e2;$  equivalente a  $\text{if } p \text{ then } x := e1 \text{ else } x := e2;$

Deduz a regra do cálculo de Hoare para este novo tipo de comandos.

### GRUPO II

**[2.5+1.5+ (2+0.75+2+1.25)]**

1. Considere a especificação algébrica de um Intervalo  $o$  e refinamento abaixo.

```
specification Interval[TotalOrderWithSuc]
 sorts
 Interval[Successorable]

 constructors
 interval: Successorable Successorable --> Interval[Successorable];

 observers
 sup: Interval[Successorable] --> Successorable;
 inf: Interval[Successorable] --> Successorable;

 others
 before: Interval[Successorable] Interval[Successorable];

 domains
 E, F: Successorable;
 interval(E,F) if geq(F,E);

 axioms
 E,F: Successorable; I,J: Interval[Successorable];

 sup(interval(E, F)) = F;
 inf(interval(E, F)) = E;

 before(I, J) iff geq(inf(J), sup(I));
end specification
```

```
specification TotalOrderWithSuc
 sorts
 Successorable

 constructors
 suc: Successorable --> Successorable;

 others
 geq: Successorable Successorable;

 axioms
 E, F, G: Successorable;

 E = F if geq(E, F) and geq(F, E);
 E = F if suc(E) = E and suc(F) = F;

 geq(E, F) if E = F;
 geq(E, F) if not geq(F, E);
 geq(E, G) if geq(E, F) and geq(F, G);
 geq(suc(E), F) if E = F;
 geq(E, suc(F)) if geq(E,F) and not (E = F);
end specification
```

```

refinement <F>
Interval[TotalOrderWithSuc] is MyInterval<F> {
 interval: e1:Successorable e2:Successorable -->? Interval[Successorable]
 is MyInterval(F e1, F e2);
 sup: Interval[Successorable] --> Successorable is F max();
 inf: Interval[Successorable] --> Successorable is F min();
 before: Interval[Successorable] e: Interval[Successorable] is boolean isBefore(MyInterval<F> e);
}

TotalOrderWithSuc is F {
 geq: Successorable e:Successorable is boolean isGreaterEq(F e);
 suc: Successorable --> Successorable is F suc();
}
end refinement

```

Apresente a correspondente especificação em JML tão completa quanto possível para o tipo `DateInterval`, resultado de edição de uma cópia do `MyInterval`. Não se esqueça que se optar por uma especificação baseada em modelos tem de definir também a função de abstração.

```

/* Immutable; implements an interval of dates. */
public class DateInterval {

 private final Date min;

 private final Date max;

 public DateInterval(Date min, Date max){ ... }

 public Date min() { ... }

 public Date max() { ... }

 public boolean isBefore(DateInterval other) { ... }

 public boolean equals(Object other){ ... }

 public String toString(){ ... }
}

```

2. Especifique formalmente, também em JML, as relações entre os dois atributos da classe `FrequencyQueue` que viram no trabalho.

```

/** The FrequencyQueue implemented with a max heap and a map. */
/**
 * The ArrayList that stores the heap.
 * Includes only entries with frequency > 0
 */
private ArrayList<Entry<E>> entries;

/**
 * The Map that stores the positions in the heap of items in the queue.
 */
private HashMap<E, Integer> map;

```

Recorde que a classe interna `Entry<E>` tem os seguintes atributos

```

private E item;
private int frequency;

```

3. Sujeitou-se a classe `InsSort` à ferramenta `Esc/Java2`.
  - (a) De acordo com o output da ferramenta, mostrado abaixo: explique em que consiste cada um dos problemas encontrados, a razão por que ocorrem e uma forma de os eliminar.
  - (b) Suponha que com as alterações por si indicadas o `Esc/Java2` deixa de assinalar qualquer problema no método `sort`. Indique, justificando, se isso permite concluir que a execução deste método não pode ter erros (ex., `NullPointerException`).

```

serie4.InsSort: sort(serie4.InsSort$Node[]) ...

--
workspaceMobius/Exame/src/InsSort.java:11: Warning: Possible null dereference (Null)
 for(int i = 1; i <= arr.length; i++) {
 ^
workspaceMobius/Exame/src/InsSort.java:13: Warning: Array index possibly too large
(IndexTooBig)
 if (arr[j].key >= arr[j-1].key) {
 ^
workspaceMobius/Exame/src/InsSort.java:13: Warning: Possible null dereference (Null)
 if (arr[j].key >= arr[j-1].key) {
 ^

```

```

1
2
3 public class InsSort {
4
5 class Node{
6 public int key;
7 public Object value;
8 }
9
10 public static void sort(Node[] arr) {
11 for(int i = 1; i <= arr.length; i++) {
12 for(int j = i; j >= 0; j--) {
13 if (arr[j].key < arr[j-1].key) {
14 Node tmp = arr[j];
15 arr[j] = arr[j-1];
16 arr[j-1] = tmp;
17 }
18 }
19 }
20 }
21 }

```

- (c) Escreva em JML um contrato apropriado para o método, i.e., que exprima que o método faz a ordenação *inplace* dos elementos do vetor dado como argumento.
- (d) Explique de que forma pode ganhar confiança de que o código do método implementa corretamente esse contrato.

### GRUPO III

[(2+0.75+0.75+1+0.5)+1]

- I. Pretende-se modelar em Promela uma pequena parte de uma rede sem fios. Neste modelo, um determinado número de aparelhos, modelados por processos do tipo *Aparelho* competem para ter acesso à rede que tem capacidade limitada. Um processo *Controlador* é quem faz o controlo de acesso, concedendo ou não acesso aos pedidos que lhe chegam dos diversos aparelhos.

```

1 #define NUM_APARELHOS 5
2 #define MAX 3
3
4 byte numUsers = 0;
5 chan ch = [0] of { byte , bool };
6
7 proctype Aparelho (byte i) {
8 bool resposta;
9 do
10 // a completar
11 od
12 }
13
14 active proctype Controlador () {
15 byte id;
16 do
17 :: ch?id,_ ->
18 if
19 :: numUsers < MAX -> ch!id,true
20 :: else -> ch!id,false
21 fi
22 od
23 }
24
25 init {
26 byte i = 0;
27 atomic {
28 do
29 :: (i >= NUM_APARELHOS) -> break
30 :: else -> run Aparelho(i); i++
31 od
32 }
33 }

```

- (a) Complete o modelo fornecido de forma a que quando um aparelho vê o seu pedido de acesso à rede concedido incrementa *numUsers* de uma unidade e imprime uma mensagem

---

de sucesso e quando vê o seu pedido recusado imprime uma mensagem de fracasso. Quando um aparelho ligado à rede se desliga decrementa *numUsers* de uma unidade. Cada aparelho tenta recorrentemente ter acesso à rede.

(Recorde que na leitura de um canal pode usar a expressão *eval(id)* para condicionar a haver um *pattern-matching* do valor lido com o valor de *id*)

- (b) Explique como, sem recorrer à LTL, pode verificar com o Spin que o número de utilizadores (*numUsers*) **nunca** excede o limite.
- (c) Como poderia confirmar que a solução que escreveu não impede que estejam o máximo número de utilizadores ligados?
- (d) Indique como procederia para verificar com o Spin (na consola ou através de um dos seus interfaces gráficos) se o sistema tem a seguinte propriedade:  
*O valor de resposta do Aparelho 1 é recorrentemente verdadeiro*
- (e) Suponha que no caso da verificação dessa propriedade o Spin reporta que explorou o espaço todo e não encontrou nenhum erro. Indique justificando se é possível concluir que não existe mesmo nenhuma execução do sistema onde a propriedade é violada.

2. Indique, justificando, as afirmações que são falsas:

- (a) O Uppaal é um *model checker* que permite verificar requisitos de tempo real escritos em LTL.
- (b) O Bandera é um *model checker* aplicável diretamente a programas Java.
- (c) O PRISM é um *model checker* aplicável a modelos definidos por autómatos cujas transições têm associadas probabilidades e que permitem modelar sistemas com comportamento estocásticos.
- (d) O verificador de propriedades associado à linguagem Dafny é baseado em *model checking*.



## Software Fiável

Mestrado em Engenharia Informática e Mestrado em Informática

### Exame 2ª Época

2 de fevereiro de 2015

Duração 3h

20 valores

#### GRUPO I

[3]

Considere o programa  $P$  apresentado abaixo.

```
prx = 5;
act = 1;
i = 0;
while i < n do
 prx = prx + 2 * act;
 act = prx - 2 * act;
 i = i + 1;
```

Recorrendo ao cálculo de Hoare, prove a correção parcial de  $P$  relativamente à condição inicial  $\{n \geq 0\}$  e à condição final  $\{act = H(n)\}$ , onde  $H(n)$  é uma sucessão definida recursivamente da seguinte forma:  $H(0)=1$ ;  $H(1)=5$ ;  $H(n) = H(n-1) + 2 * H(n-2)$  se  $n \geq 2$ .

#### GRUPO II

[(3+1.5+1)+(2+1+1.5)]

- I. Considere a especificação algébrica do tipo `Tab` fornecida, que representa um tabuleiro tipo batalha naval, onde as casas podem estar ocupadas e/ou reveladas. Considere ainda o refinamento para o tipo Java `EstadoTabuleiro`.

```
specification Tab
 sorts
 Tab
 constructors
 empty: int -->? Tab;
 ocupa: Tab int int -->? Tab;
 revela: Tab int int -->? Tab;
 observers
 dim: Tab --> int;
 ocupada: Tab int int;
 revelada: Tab int int;

domains T: Tab; i,j,n: int;
 empty(n) if n>=1;
 ocupa(T,i,j) if i>=1 and i<=dim(T) and j>=1 and j<=dim(T);
 revela(T,i,j) if i>=1 and i<=dim(T) and j>=1 and j<=dim(T);

axioms T: Tab; n,i,j,k,l: int;

 dim(empty(n)) = n;
 dim(ocupa(T,i,j)) = dim(T);
 dim(revela(T,i,j)) = dim(T);

 not ocupada(empty(n),k,l) if n>=1 and k>=1 and k<=n and l>=1 and l<=n;
 ocupada(ocupa(T,i,j),k,l) iff
 i>=1 and i<=dim(T) and j>=1 and j<=dim(T) and k>=1 and k<=dim(T) and
 l>=1 and l<=dim(T) and ((i=k and j=l) or ocupada(T,k,l));
```

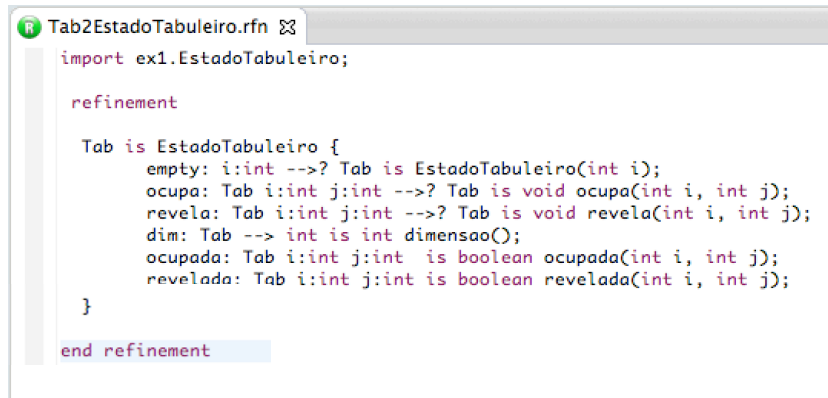
```

ocupada(revela(T,i,j),k,l)
 iff i>=1 and i<=dim(T) and j>=1 and j<=dim(T) and k>=1 and k<=dim(T) and
 l>=1 and l<=dim(T) and ocupada(T,k,l);

not revelada(empty(n),k,l)
 if n>=1 and k>=1 and k<=n and l>=1 and l<=n;
revelada(revela(T,i,j),k,l)
 iff i>=1 and i<=dim(T) and j>=1 and j<=dim(T) and k>=1 and k<=dim(T) and
 l>=1 and l<=dim(T) and ((i=k and j=l) or revelada(T,k,l));
revelada(ocupa(T,i,j),k,l)
 iff i>=1 and i<=dim(T) and j>=1 and j<=dim(T) and k>=1 and k<=dim(T) and
 l>=1 and l<=dim(T) and not(i=k and j=l) and revelada(T,k,l);
end specification

```

- (a) Apresente a correspondente especificação JML baseada em propriedades para o tipo EstadoTabuleiro. A especificação deve ser tão completa quanto possível e não esqueça que se destina aos clientes da classe.



```

Tab2EstadoTabuleiro.rfn
import ex1.EstadoTabuleiro;

refinement

 Tab is EstadoTabuleiro {
 empty: i:int -->? Tab is EstadoTabuleiro(int i);
 ocupa: Tab i:int j:int -->? Tab is void ocupa(int i, int j);
 revela: Tab i:int j:int -->? Tab is void revela(int i, int j);
 dim: Tab --> int is int dimensao();
 ocupada: Tab i:int j:int is boolean ocupada(int i, int j);
 revelada: Tab i:int j:int is boolean revelada(int i, int j);
 }

end refinement

```

- (b) Tendo em conta os detalhes da implementação revelados abaixo que invariantes ou outras propriedades JML seria útil especificar?

```

public class EstadoTabuleiro {
 /*
 * Constantes que representam o estado das casas do tabuleiro
 * (um enum seria mais adequado, mas o JML não sabe o que são)
 */
 public static final int LIVRE_ESCONDIDA = 0;
 public static final int OCUPADA_ESCONDIDA = 1;
 public static final int LIVRE_REVELADA = 2;
 public static final int OCUPADA_REVELADA = 3;

 //elementos do tabuleiro são necessariamente um dos quatro valores acima
 private final int[][] tabuleiro;

 public EstadoTabuleiro(int dim) { ... }
 ...
}

```

- (c) Explique com detalhe de que forma poderia usar as propriedades JML que escreveu para ganhar confiança na correção do código da classe.

## 2.

- (a) A classe *ObjectStack* foi sujeita a verificação com o *Esc/Java2* sendo que o output da verificação inclui o que é mostrado ao lado. Explique a razão de ser de cada problema identificado e como deve ser endereçado.
- ObjectStack.java:7: Warning: Possible attempt to allocate array of negative length  
a = new Object[1];
  - ObjectStack.java:11: Warning: Possible null dereference  
a[n] = null;
  - ObjectStack.java:11: Warning: Possible negative array index  
a[n] = null;
  - ObjectStack.java:21: Warning: Array index possibly too large  
a[n] = null;
- (b) Explique porque é que os problemas identificados pelo *Esc/Java2* são classificados pela ferramenta como *warnings*.
- (c) A linguagem do *Esc/Java2* inclui o *pragma assume* que permite juntar factos não verificados ao programa. Explique (i) o que justifica a existência deste *pragma*, (ii) sob que condições este deve ser usado e (iii) as consequências da sua má utilização.

```

class ObjectStack {
 Object [] a; int n;
 //@ invariant (\forall int i;
 //@ n <= i & i < a.length ==> a[i] == null);
 ObjectStack(int l) {
 n = 0;
 a = new Object[l];
 }
 void Pop() {
 --n;
 a[n] = null;
 }
 ...
}

```

### GRUPO III

[(2.5+1+1.25+1.25)+1]

- I. O algoritmo apresentado abaixo foi proposto para resolver o problema da exclusão mútua. Considere um sistema com três processos concorrentes executando este algoritmo

```

int array[1..n] tok ← [0,..., 0]

Process i
 loop forever
 non-critical section
 tok[i] ← max(tok) + 1
 for all other processes j
 await (tok[j] = 0 || tok[i] < tok[j])
 critical section
 tok[i] ← 0

```

- (a) Apresente um modelo do sistema em Promela que permita facilmente alterar o número de processos do sistema e que leve em consideração que a operação *max* usado no algoritmo é atômica.
- (b) Explique como, sem recorrer à LTL, pode verificar com o Spin que dois processos nunca podem estar simultaneamente na região crítica.
- (c) Analise, no seu modelo, até onde podem crescer os valores das “senhas” *tok[i]* obtidas pelos processos quando querem entrar na secção crítica. Analise nomeadamente se pode voltar a ser obtida a senha 0 por algum processo. De que forma isso afecta a propriedade de exclusão mútua?
- (d) Indique como procederia para verificar com o Spin (na consola ou através de um dos seus interfaces gráficos) se o sistema tem a seguinte propriedade:

*Se em determinado momento o processo 1 está à espera para aceder à região crítica, então não é possível que o processo 2 ou 3 entre duas vezes na região crítica antes do 1*

2. Indique, justificando, as afirmações que são falsas:

- (a) O Uppaal é um *model checker* que permite verificar requisitos de tempo real definidos numa lógica temporal ramificada.
- (b) O CMC é um *model checker* aplicável diretamente a programas C e C++ em que os processos comunicam por memória partilhada e seguem um protocolo *event-driven*.
- (c) A linguagem de input do PRISM permite modelar sistemas com comportamento estocástico.
- (d) A linguagem Dafny permite escrever programas OO que incorporam especificações, cuja correção é verificada em tempo de compilação.





**Material for the exam:** You need only a pen and your student ID card. Bags and backpacks, including mobile phones, should be left by the blackboard. Violations to this norm will invalidate your test.

**On exam writing:** Start each new group on a new page.

**Duration:** Two hours and thirty minutes.

**Exercise 1.** [Hoare Logic; 2.5 points]

a) Hoare Logic uses triples to reason about program correctness. Is the following triple correct? Justify.

$$\{true\} \textbf{ while true do } x := 5 \{x \neq 5\}$$

b) Show that the following triple holds.

$$\{x > -5\} \textbf{ while } x > 0 \textbf{ do } x := x - 1 \{-5 < x \wedge x \leq 0\}$$

c) Given the pre-condition  $x > -5$ , suppose you want to make sure that the above program leaves the value 0 in variable  $x$ , if it terminates. Will the invariant  $x > -5$  be adequate for the effect? If yes explain why. Otherwise propose an appropriate invariant. In any case explain how to proceed in proving the result.

d) Consider the following variant of the **while** rule.

$$\frac{\{\varphi\} S \{\varphi\}}{\{\varphi\} \textbf{ while } p \textbf{ do } S \{\varphi\}}$$

Does it allow to infer correct assertions only? If yes explain why; if not give a counterexample.

**Exercise 2.** [Java Modelling Language; 3 points]

a) Consider the following Java class equipped with JML annotations.

```
class Neg {
 /*@
 @ requires n != 0;
 @ ensures \result == -n;
 @*/
 static int neg (int n) {return 2 * n;}
 public static void main (String[] a) {neg (0);}
}
```

Explain what happens when one compiles the class with the JML compiler (jmlc). Justify. If you think that the class does not compile, suggest changes so that it may compile smoothly.

b) Explain what happens when one runs the above program with `jmlrac` (possibly amended in accordance with the previous exercise). Justify.

c) A computer disk, in its simplest form, is a container of fixed capacity capable of taking in a variable number of items (usually called files). Each item is characterized by its length, a non-negative integer number. The sum of the lengths of the items in a disk cannot exceed the disk capacity. The disk capacity is itself a non-negative integer number, and remains constant throughout the lifetime of the disk. Disk access is index based; indices start at zero (as for Java arrays). In order to simplify our model, the items in the disk are represented as (non-negative) integers describing the length of the item. There are usually multiple operations that manipulate disks; we are interested in a small subset of these, including the following operations, which are part of **interface** `Disk`.

- `/*@ pure @*/ int get(int index)` returns the element at the specified position in the disk. The index must be in range, that is, it must be non-negative and smaller than the number of elements in the disk;
- `/*@pure @*/int available ()` returns the free remaining space in the disk;
- `void store(int item, int index)` inserts the specified item at the specified position in the disk. Shifts the item currently at that position (if any) and any subsequent elements to the right (adds one to their indices). There must be space available on the disk, and the index must be non-negative and smaller or equal to the number of elements in the disk.

Write a *model-based* JML specification for an interface comprising the above methods; consider more methods if needed; write comprehensive, model-based, contracts to all your methods.

Note: In case you do not remember the exact names of the operations in the JML model you have chosen, pick new names, write the method signatures, and explain their behaviours.

d) Now consider a list-based implementation of the above interface.

```
class ListDisk implements Disk {
 private java.util.List items;
 ...
}
```

Write the *abstraction function(s)* for the model you have chosen. Add more fields and methods if necessary for the purpose of the describing the abstraction function. Do **not** implement the class; I am only interested in the abstraction function.

### Exercise 3. [ESC Java; 2.5 points]

a) The ESC/Java tool is not complete. Explain the significance of this statement. Identify a source of incompleteness. Why is the tool useful in spite of this limitation?

**b)** Consider the following class, intended to describe a multiset of Object values.

```

1 class MultiSet {
2 Object[] a;
3 int n;
4 //@ invariant (\forall int i; n<=i && i<a.length; a[i]==null);
5 MultiSet(int l) {
6 n = 0;
7 a = new Object[l];
8 }
9 void add(Object o) {
10 if (n == a.length)
11 grow();
12 a[n++] = o;
13 }
14 private void grow() {
15 Object[] b = new Object[a.length * 2 + 1];
16 for (int i = 0; i < a.length; i++)
17 b[i] = a[i];
18 a = b;
19 }
20 }

```

We ran ESC/Java against the MultiSet class; a portion of the output is shown below. For each of the five problems encountered a) explain its root, and b) propose a fix.

1. MultiSet.java:7: Warning: Possible attempt to allocate array of negative length (NegSize)  
a = new Object[1];  
                  ^
2. MultiSet.java:10: Warning: Possible null dereference (Null)  
if (n == a.length)  
          ^
3. MultiSet.java:12: Warning: Possible negative array index (IndexNegative)  
a[n++] = o;  
      ^
4. MultiSet.java:12: Warning: Array index possibly too large (IndexTooBig)  
a[n++] = o;  
      ^
5. MultiSet.java:12: Warning: Type of right-hand side possibly not a subtype of array element type (ArrayStore)  
a[n++] = o;  
          ^

c) Suppose that you eventually manage to annotate your program in such a

way that ESC/Java reports no errors or warnings. What can you conclude about the correctness of the code against its specification?

**Exercise 4.** [Model Checking; 3 points]

The following incorrect solution to the mutual exclusion problem was published in the January 1966 issue of Communications of the ACM. The algorithm is for two processes.

```
Boolean array $b(0; 1)$ integer k, i, j ,
comment This is the program for computer i , which may be
either 0 or 1, computer $j \neq i$ is the other one, 1 or 0;
 $C0$: $b(i) := \text{false}$;
 $C1$: if $k \neq i$ then begin
 $C2$: if not $b(j)$ then go to $C2$;
 else $k := i$; go to $C1$ end;
 else critical section;
 $b(i) := \text{true}$;
 remainder of program;
 go to $C0$;
 end
```

- a) Model the algorithm in Promela.
- b) Explain in detail how one may prove or disprove the claim that the two processes cannot both enter their critical sections simultaneously.
- c) Suggest a temporal logic formula expressing the idea that whenever one process enters its critical region, then eventually the other will enter its critical region.
- d) Spin can be used in simulation mode or in verification mode. Explain the difference. Does any of the modes ensure the correctness of the model with respect to its specification? Justify.

**Exercise 5.** [Advanced Topics; 1 points]

Which of the following claims are false? Justify.

- a) CMC is a model checker that works directly on C and C++ code, and can be used to prove properties of concurrent programs communicating via shared memory.
- b) UPPAL is a model checker specifically designed to check properties of fault tolerant computer networks.
- c) Alloy is a specification language for expressing structural constraints and behaviour and a tool to prove the correctness of models.
- d) JavaPathFinder is a tool able to detect deadlocks and unhandled exceptions in Java bytecode.



**Material for the exam:** You need only a pen and your student ID card. Bags and backpacks, including mobile phones, should be left by the blackboard.

Violations to this norm will invalidate your test.

**On exam writing:** Start each new group on a new page.

**Duration:** Two hours and thirty minutes.

**Exercise 1.** [Hoare Logic; 2.5 points]

a) Show that the following program computes the product of two numbers  $m$  and  $n$  when  $m$  is non-negative.

```
{m ≥ 0}
i := m;
p := 0;
while i != 0 do
 p := p + n;
 i := i - 1;
{p = m × n}
```

b) Prove that the above program terminates.

c) “Partial”, “hypothetical”, and “imprecise” are three adjectives often used when referring to Hoare Logic. In which sense is the logic partial, hypothetical, and imprecise?

**Exercise 2.** [Java Modelling Language; 3 points]

Consider the following Java classes equipped with JML annotations.

```
class A {
 //@ requires !contains(x);
 //@ ensures size() == \old(size()) + 1;
 void add (E x) { ... }
 ...
}

class B extends A {
 //@ also
 //@ requires contains(x);
 //@ ensures size() == \old(size());
 @Override
 void add (E x) { ... }
 ...
}
```

a) Under which conditions can one call method `add` at class B? What is ensured in this case?

**b)** Suppose that a code refactoring process determined that class A has become obsolete and that class B should now inherit directly from class Object. Rewrite the contract for method `add` in the new B class.

**c)** Decks of cards are used in many games. For the purpose of this exercise we are interested in decks whose cards are identified by natural numbers. We assume a method `int dimension()` that returns the number of cards in a given deck; each card is then represented by a number between 0 and `dimension() - 1`. Decks are not supposed to contain duplicated cards. Among the various operations usually available on a deck of cards, consider the following:

1. A method to peek the card at the top of deck, `int peekTopCard()`. The card is supposed to remain in the same position in the deck;
2. A method to discard the card at the top, `void removeTopCard ()`;
3. A method to return a card to the deck, `void returnCard ()`. The card must have been taken from the deck before.
4. A method to return a series of cards to the deck, `void returnCards (int[] cards)`. Each card in the array must have been previously taken from the deck.

Write a *model-based* JML specification for an interface comprising the above methods; consider more methods if needed; write comprehensive, model-based, contracts to all your methods.

Note: In case you do not remember the exact names of the operations in the JML model you have chosen, pick new names, write the method signatures, and explain their behaviours.

**d)** Now consider a list-based implementation of the above interface.

```
class ArrayDeck implements Deck {
 private int [] cards;
 ...
}
```

Write the *abstraction function(s)* for the model you have chosen. Add more fields and methods if necessary for the purpose of the describing the abstraction function. Do **not** implement the class; I am only interested in the abstraction function.

### Exercise 3. [ESC Java and Dafny; 2.5 points]

Consider the following code snippet, as part of some method in some class.

```
y = x * x + 2 * x + 1;
//@ assume y >= 0;
z = new int[y];
```

- a) Suppose that we run the code with the ESC/Java tool. What is the expected output?
- b) Is there a value for  $x$  that will force the code to fail? Which? What is the expected behaviour of the Java program in this case?
- c) From the answers to the above two exercises, what can one conclude about the soundness and the completeness of the ESC/Java tool? Why?
- d) Consider the following method written in the Dafny language.

```

0 method divide (n: int) {
1 var k : int := 0;
2 var d : int := -3;
3 while k < n {
4 k := k + 1;
5 d := d + 1;
6 }
7 var x : int := 100 / d;
8 }

```

Running the Dafny compiler on the above code produces the following output:

```
divide.dfy(7,22): Error: possible division by zero
```

Explain why.

- e) The method `divideByZero` can be easily translated into Java. We ran the thus obtained code through ESC/Java, to obtain the following output:

```

Divide: divide(int) ...
 [0.053 s 33172080 bytes] passed

```

Explain why. How should one call, in this case, the ESC/Java tool in order to obtain a behaviour similar to that of Dafny?

#### Exercise 4. [Model Checking; 3 points]

Consider the following candidate algorithm for mutual exclusion.

```

bool c1 := True
bool c2 := True

```

| <b>While</b> ( <b>True</b> ){ |                           | //process 1 | <b>While</b> ( <b>True</b> ){ |                           | //process 2 |
|-------------------------------|---------------------------|-------------|-------------------------------|---------------------------|-------------|
| 0                             | c1 := <b>False</b>        |             | 0                             | c2 := <b>False</b>        |             |
| 1                             | <b>While</b> (!c2){}      | //busy wait | 1                             | <b>While</b> (!c1){}      | //busy wait |
| 2                             | <i>critical section 1</i> |             | 2                             | <i>critical section 2</i> |             |
| 3                             | c1 := <b>True</b> }       |             | 3                             | c2 := <b>True</b> }       |             |

- a) Model the algorithm in Promela.

- b)** Explain in detail how, using the Spin model checker, one may prove or disprove the claim that the two processes cannot both enter their critical sections simultaneously.
- c)** Suggest a temporal logic formula expressing absence of deadlock, i.e., that it is impossible to reach a state in which one process is trying to enter its critical section, but never succeeds.
- d)** Again, explain in detail how, using the Spin model checker, one may prove or disprove absence of deadlocks for the given algorithm.
- e)** The above algorithm cannot ensure the mutual exclusion property. Exhibit a counter-example, in the form of a computation.

**Exercise 5.** [Advanced Topics; 1 points]

Which of the following claims are false? Justify.

- a)** CMC is a model checker that works directly on C and C++ code, and can be used to prove properties about distributed systems communicating via message passing.
- b)** UPPAL is a model checker tool for real-time systems that uses Linear Temporal Logic.
- c)** The Alloy Analyzer is a constraint solver that provides fully automatic simulation and checking.
- d)** JavaPathFinder is a tool designed to verify executable Java bytecode.



# Software Fiável

Mestrado em Engenharia Informática e Mestrado em Informática

## Exame I

13 de janeiro de 2017

**Duração** 3h

**20 valores**

### GRUPO I

[1+2+3]

1. Suponha que a linguagem de programas para a qual foi definido o cálculo de Hoare é estendida com um novo tipo de atribuição da forma

$x ::= \text{exp};$  que, como usualmente, é equivalente a  $x := x + \text{exp};$

A partir do axioma do cálculo de Hoare para a atribuição, deduza um axioma que capture o significado da nova instrução de atribuição.

2. Considere o programa  $P$  apresentado abaixo.

```
i := n;
while i >= 3 do
 i := i - 3;
b := (i==0);
```

Recorrendo ao cálculo de Hoare, prove a correção parcial de  $P$  relativamente a

$\{n \geq 1\} P \{b \equiv (n \text{ é múltiplo de } 3)\}.$

3. Considere o programa Dafny mostrado ao lado, guardado no ficheiro `exame1.dfy`. O programa calcula a divisão de  $x$  por  $y$ , colocando o quociente e o resto nas variáveis  $q$  e  $r$ , respetivamente.

- (a) Explique, em termos de triplos de Hoare, que verificação é realizada quando se executa

`$ dafny exame1.dfy`

- (b) Considere o output obtido, mostrado abaixo. Explique cada um dos erros reportados e a sugestão fornecida para resolver o primeiro erro.

- (c) Como modificaria o programa de forma a eliminar os dois erros.

```
1 method IntDivision () {
2 var x, y, q, r: int;
3
4 q := 0;
5 r := x;
6
7 while r >= y
8 invariant q * y + r == x;
9 invariant r >= 0 {
10 r := r - y;
11 q := q + 1;
12 }
13 assert q * y + r == x;
14 assert r >= 0;
15 assert r < y;
16 }
```

```
$ dafny exame1.dfy
Dafny program verifier version 1.9.8.30829, Copyright (c) 2003-2016,
Microsoft.
exame1.dfy(7,2): Error: cannot prove termination; try supplying a decreases
clause for the loop
[...]
exame1.dfy(9,15): Error BP5004: This loop invariant might not hold on entry.
[...]
Dafny program verifier finished with 1 verified, 2 errors
```

- I. Considere a classe `RobotState` fornecida de seguida, cujos objetos representam o estado de robôs que vão ser utilizados para explorar cavernas onde há tesouros e monstros, produzindo mapas completos das mesmas.

Um objeto `rbt` do tipo `RobotState` está preparado para receber e processar informação relativamente ao local em que o robô se encontra: há um tesouro? há um monstro? ou o local está livre? O objeto `rbt` pode ainda receber instruções para dar um passo numa determinada direção, desde que esse movimento seja possível e já tenha lido sido fornecida informação relativamente à sua posição atual. Quando a caverna está totalmente explorada, `rbt` deverá receber uma instrução para ser teletransportado de volta para sua posição inicial. Porque o teletransporte é uma atividade que consome muita energia, esta instrução só é possível quando `rbt` já terminou a sua tarefa de exploração.

Neste exercício pretende-se que, recorrendo ao JML, equipe a classe fornecida com especificações que expressem as propriedades informalmente descritas acima e também nos comentários existentes no código.

- (a) Comece por equipar a classe com invariantes que traduzam as propriedades que caracterizam os estados válidos de um objeto do tipo `RobotState`.
- (b) Recorrendo a uma especificação baseada em propriedades, equipe os métodos públicos da classe com contratos JML apropriados. Não se esqueça de indicar também os *pragmas* JML necessários para que o `jmlc` não assinalar erros de sintaxe.

```
public class RobotState {
 // The robots internal map of the world to explore
 // 0 - the location not yet visited,
 // 1 - the location has been visited
 // 2 - the location has been visited and there is a treasure
 // 3 - the location has been visited and there is a monster
 private int[][] world;
 // Current x,y coordinates, a location in the map
 private int x, y;

 public RobotState(int w, int h){
 world = new int[w][h];
 for(int i=0; i<w;i++) // nothing visited
 for(int j=0; j<h;j++)
 world[i][j] = 0;
 x = 0; y = 0; // starting location
 }
 // Provides information about a location in this world
 public int getInfo(int x1, int y1){ return world[x1][y1];}

 // Provides an array with the current x, y coordinates.
 public int[] getCurrentLocation(){ return new int[]{x,y};}

 // Holds if location (x1,y1) is a valid location in this world.
 public boolean isAllowedLoc(int x1, int y1){ }

 // Holds if the whole world is explored.
 public boolean isCompletelyExplored(){ }

 //Robot updating current location with info from sensors.
 //input==1 means current location is empty
 //input==2 means there is a treasure here!
 //input==3 means there is a monster here!
 public void processSensorReading(int input) { }

 // Robot moves one step south (i.e it decreases its y-coordiante by 1),
 // provided that is possible.
 // If the new location has not been visited before, the robot should update it
 // and set its status as "visited" (i.e to 1).
 // If the new location has been visited, its status should remain the same.
 public void moveSouth() { }

 // Robot teleports back to the starting location (0,0),
 // provided that it has finished exploring the cave.
 public void teleportHome() { }

 // other methods not shown
 // void moveNorth(), void moveEast(), void moveWest()
}
```

2. Sujeitou-se a classe `RobotState` à ferramenta `Esc/Java2`. A parte relevante do output da ferramenta é mostrada abaixo.
- (a) Explique cada um dos problemas reportados.
- (b) Algum destes problemas desaparece quando juntarmos ao programa os contratos JML que escreveu para os elementos envolvidos? E quando juntarmos ao programa os invariantes JML que escreveu para a classe? Indique como poderia eliminar os problemas remanescentes (se existirem).
- (c) Suponha que com as alterações indicadas em (b) o `Esc/Java2` deixa de assinalar qualquer problema na classe. Indique, justificando, se isso permite concluir que a execução do método `getInfo` nunca vai dar origem a um `NullPointerException`.

```

10 public RobotState(int w, int h){
11 world = new int[w][h];
12 for(int i=0; i<w; i++){
13 for(int j=0; j<h; j++){
14 world[i][j] = 0;
15 }
16 }
17 x = 0; y = 0; // starting location
18 // Provides information about a location in this world ...
19 public int getInfo(int x1, int y1){ return world[x1][y1]; }
20 }

```

RobotState ...

Prover started:0.028 s 92756000 bytes  
[0.337 s 93015328 bytes]

AulasSF1617/src/RobotState.java:12: Warning: Possible attempt to allocate array of negative length (NegSize)

```

 world = new int[w][h];
 ^

```

RobotState: getInfo(int, int) ...

AulasSF1617/src/RobotState.java:19: Warning: Possible null dereference (Null)

```

 public int getInfo(int x1, int y1){ return world[x1][y1]; }
 ^

```

### GRUPO III

[6]

Considere o seguinte algoritmo de exclusão mútua. O algoritmo, que foi proposto por T.Anderson, recorre a uma fila de *locks*.

```

const
 has_lock = 0 ;
 must_wait = 1

shared variable
 Slots: array[0..N - 1] of {has_lock, must_wait};
 Next_slot: integer initially 0

initially
 Slots[0] = has_lock ∧ (∀ k : 0 < k < N :: Slots[k] = must_wait)

process p /* 0 ≤ p < N */
private variable
 my_place: integer

while true do
1: Noncritical Section;
2: my_place := fetch_and_inc(Next_slot);
3: if my_place = N - 1 then
4: atomic add(Next_slot, -N)
 fi;
5: my_place := my_place mod N;
6: await Slot[my_place] = has_lock; /* spin */
7: Slots[my_place] := must_wait;
8: Critical Section;
9: Slot[my_place + 1 mod N] := has_lock
od

```

- (a) Modele o algoritmo em Promela. O modelo deve contemplar a hipótese de um processo, a partir de certa altura, não querer entrar mais na seção crítica (i.e., ficar indefinidamente na seção não crítica).
- (b) Explique como poderíamos verificar, com o Spin, se o algoritmo garante efetivamente a exclusão mútua (em qualquer altura, há no máximo um processo na seção crítica).

- (c) Indique se o sistema modelado tem a seguinte propriedade:

*Há recorrentemente algum processo na seção crítica.*

Se a propriedade não se verificar, indique um contraexemplo.

- (d) Explique como poderíamos, com o Spin, verificar se o sistema modelado tem a propriedade acima.
- (e) O Spin pode ser usado em modo de simulação ou de verificação. Explique a diferença entre os dois modos. Algum dos modos permite concluir que um modelo é correto face à especificação?

#### **GRUPO IV**

**[1.5]**

**Responda apenas a UMA das seguintes questões:**

**I a.** *FacebookInfer*

- (a) Explique sumariamente o que é o *FacebookInfer* e em que técnicas se baseia.
- (b) Dê exemplo de 3 tipos de problemas que são detetados pela ferramenta

**I b.** *JavaPathFinder*

- (a) Explique sumariamente o que é o *JavaPathFinder*.
- (b) Dê exemplo de 3 tipos de problemas que são detetados pela ferramenta

# Software Fiável

Mestrado em Engenharia Informática e Mestrado em Informática

## Exame 2

30 de janeiro de 2017

**Duração** 3h

**20 valores**

### GRUPO I

[1+2+3]

1. Suponha que a linguagem de programas para a qual foi definido o cálculo de Hoare é estendida com um novo tipo de atribuição da forma

$x := x * \text{exp};$  que, como usualmente, é equivalente a  $x := x * \text{exp};$

A partir do axioma do cálculo de Hoare para a atribuição, deduza um axioma que capture o significado da nova instrução de atribuição.

2. Considere o programa  $P$  apresentado abaixo.

```
q := 0;
r := x;
while r >= y do
 r := r - y;
 q := q + 1;
```

Recorrendo ao cálculo de Hoare, prove a correção parcial de  $P$  relativamente a

$$\{y \geq 0\} P \{x = q*y + r \wedge r < y\}.$$

3. Considere o programa **Dafny** mostrado ao lado, guardado no ficheiro `exame2.dfy`.

- (a) Explique, em termos de triplos de Hoare, que verificação é realizada quando se executa

```
$ dafny exame2.dfy
```

- (b) Considere o output obtido, mostrado abaixo. Explique, com detalhe, o que significa o erro reportado, porque acontece, e de que forma o programa deve ser modificado de forma a eliminar esse erro.

```
$ dafny exame2.dfy
Dafny program verifier version
1.9.6.21116, Copyright (c) 2003-2015,
Microsoft.
exame2.dfy(13,16): Error BP5004: This
loop invariant might not hold on entry.
Execution trace:
(0,0): anon0
Dafny program verifier finished with 2
verified, 1 error
```

- (c) Explique porque razão o Dafny não reporta nenhum problema relativamente à prova da terminação.

```
1 function pow (n: nat) : nat {
2 if n == 0
3 then 1
4 else 2 * pow (n - 1)
5 }
6
7 method power(){
8 var n, k, j :int;
9
10 k := 0;
11 j := 1;
12 while k < n
13 invariant k <= n
14 invariant j == pow(k) {
15 k := k + 1;
16 j := 2 * j;
17 }
18 assert j == pow(n);
19 }
```

1.

- (a) Recorrendo ao **JML**, equipe o interface Vector com especificações que capturem o que representa este tipo (nomeadamente as propriedades informalmente descritas nos comentários).

```
/**
 * A mutable sparse vector (i.e., a vector with few non-zero components).
 */
public interface Vector {

 /** number of components of this vector */
 int dim();

 /** number of non-zero components of this vector */
 int els();

 /** returns the value of the component i of this vector */
 double get(int i);

 /** sets the value of the component i of this to val */
 void set(int i, double val);

 /** if has only zero components */
 boolean isZero();

 /** multiplies every component of this by val */
 void scalarProd(double val);

 /** sums other to this, assuming the other vector
 * has the same dimension */
 void sum(Vector other);

 /** returns the dot product of this and other,
 * assuming the other vector has the same dimension */
 double dotProd(Vector other);
}
```

```

/**
 * A mutable implementation of sparse vector,
 * based on a hash table.
 */
public class HTSparseVector implements Vector{

 private static class Component {
 private int index; /** The key */
 private double value; /** The value */
 private Component(int index, double f) {
 this.index = index; this.value = f;
 }
 }

 //The array holding the non-zero components of the vector.
 private Component[] hashTable;
 // The dimension of the vector.
 private final int dimension;
 //The number of non-null components in this vector.
 private int nonZeroComponents;
 //The number of deleted components in the table,
 //since its inception or since the last rehash operation.
 private int numDeletes;

 //The initial capacity of the table.
 private static final int INITIAL_CAPACITY = 11;
 //The maximum load factor allowed.
 private static final double LOAD_THRESHOLD = 0.75;

 //A marker for a deleted entry.
 private static Component DELETED = new Component(-1, 0);

 public HTSparseVector(int dim){
 hashTable = new Component[INITIAL_CAPACITY];
 dimension = dim;
 nonZeroComponents = 0;
 numDeletes = 0;
 }
}

```

- (b)** Considere agora o excerto da classe `HTSparseVector` fornecida acima, baseada numa tabela de dispersão, eficiente para representar vetores com uma percentagem pequena de componentes não nulas (vetores esparsos).

Equipe esta classe com invariantes que traduzam as propriedades que caracterizam os estados válidos de um objeto do tipo `HTSparseVector`.

2. Sujeitou-se a classe `HTSparseVector` à ferramenta **Esc/Java2**. Uma parte do output da ferramenta é mostrada abaixo.

HTSparseVector: get(int) ...

```
src/HTSparseVector.java:60: Warning: Possible
null dereference (Null)
 if (hashTable[pos] == null)
```

```
58 public double get(int i){
59 int pos = findPos(i);
60 if (hashTable[pos] == null)
61 return 0;
62 else
63 return hashTable[pos].value;
64 }
```

HTSparseVector: set(int, double) ...

```
src/HTSparseVector.java:71: Warning: Possible
negative array index (IndexNegative)
 if (hashTable[pos] == null)
 ^
```

```
66 public void set(int i, double f){
67 if (f == 0){
68 remove(i);
69 }
70 else {
71 int pos = findPos(i);
72 if (hashTable[pos] == null){
73 addComponent(pos, new Component(i, f));
74 }
75 else {
76 hashTable[pos].value += f;
77 }
78 }
79 }
```

- (a) Explique cada um dos problemas reportados.
- (b) Algum destes problemas desaparece quando juntarmos ao programa os contratos JML que escreveu para os elementos envolvidos? E quando juntarmos ao programa os invariantes JML que escreveu para a classe?
- (c) Explique o que teria de fazer para usar o **jmlrac** para ganhar confiança relativamente à correção da classe `HTSparseVector` face às propriedades com que anotou o interface `Vector`. Explique em concreto que verificações serão feitas e que tipo de output pode ser esperado.

### GRUPO III

[6]

Considere o seguinte algoritmo de exclusão mútua. O algoritmo, que foi proposto por L.Lamport, recorre a memória partilhada.

```
shared variable
 B: array[1..N] of boolean initially false;
 X: 1..N;
 Y : 0..N initially 0

process p /* 1 <= p <= N */

private variable
 j: 1..N

while true do
1: Noncritical Section;
2: B[p] := true;
3: X := p;
4: if Y != 0 then
5: B[p] := false;
6: await Y = 0 ; /* busy wait */
7: goto 1
 fi;
8: Y := p;
9: if X != p then
10: B[p] := false;
11: for j := 1 to N do
12: await ¬B[j] /* busy wait */
 od;
13: if Y != p then
14: await Y = 0 ; /* busy wait */
15: goto 1
 fi
 fi;
16: Critical Section;
17: Y := 0;
18: B[p] := false
od
```

- (a) Modele o algoritmo em **Promela**.
- (b) Explique como poderíamos verificar, com o **Spin**, se o algoritmo garante efetivamente a exclusão mútua (em qualquer altura, há no máximo um processo na seção crítica).
- (c) Explique como poderíamos verificar, com o **Spin**, se o algoritmo garante que  
*Sempre que um processo quer entrar na seção crítica, mais tarde ou mais cedo entra na seção crítica.*
- (d) Suponha que o algoritmo não garante a propriedade acima. Indique, justificando, se há garantia que o Spin reporte que a propriedade é violada. Se isso acontecer, que tipo de output será dado pela ferramenta?
- (e) Explique o que acontece se o problema conhecido por *state explosion* se manifestar quando estamos a usar o Spin e dê exemplo de uma técnica que o Spin implemente para aliviar este problema.

### GRUPO IV

[1.5]

Responda apenas a **UMA** das seguintes questões:

- a. *FacebookInfer*: Explique sumariamente em que diferem os triplos de Hoare  $\{pre\}prog\{post\}$  que são usados pelo *FacebookInfer* e porque foi necessário estender a lógica de programas subjacente.
- b. *JavaPathFinder*: Explique sumariamente em que medida o processo de verificação de modelos que pode ser conduzido recorrendo ao *JavaPathFinder* difere do processo suportado pelo *Spin*.



**Material for the exam:** You need only a pen and your student ID card.  
Bags and backpacks, mobile phones, and coats should be left by the blackboard. Violations to this norm will invalidate your test.

**On exam writing:** Start each new group on a new page.

**Duration:** Two hours and thirty minutes.

**Exercise 1.** [Hoare Logic; 2.5 points]

a) The program below calculates  $z = \sum_{i=0}^n i$ . Show that the following triple is correct.

```
{z = 0 ∧ m = 0}
while m ≤ n do
 z := z + m;
 m := m + 1
```

```
{z = $\frac{n(n+1)}{2}$ }
```

b) Show that the above program terminates under the given pre-condition.

c) The Hoare Calculus contains a rule for an **if–then–else** conditional statement. Imperative programming languages often include **if–then** statements. Imagine we add the following rule to the Hoare Calculus.

$$\frac{\{\varphi \wedge p\} S \{\varphi\}}{\{\varphi\} \text{ if } p \text{ then } S \{\varphi\}}$$

Would the calculus remain sound? Would it remain complete? In each case: if yes explain why, if not give a counterexample.

**Exercise 2.** [Dafny; 2.5 points]

Consider the following program written in the Dafny programming language.

```
1 method computeFibonacci (n: nat) returns (m: nat)
2 ensures m == fib(n)
3 {
4 var i := 0;
5 var k := 1;
6 m := 0;
7 while i < n
8 invariant m == fib(i) && k == fib(i + 1)
```



```

9 {
10 var x := k;
11 k := m + k;
12 m := x;
13 i := i + 1;
14 }
15 }
```

a) Explain in terms of Hoare triples the sort of verification performed by the Dafny compiler when method `computeFibonacci` is compiled. Assume that all variables in the Hoare Calculus are of integer type.

b) For the above program the Dafny compiler complains with the following error message:

```
exam1.dfy(2,14): Error: unresolved identifier: fib
```

The `fib` identifier refers to the Fibonacci function, defined on natural numbers as follows:

$$\text{fib}(n) = \begin{cases} n & \text{if } n < 2 \\ \text{fib}(n-2) + \text{fib}(n-1) & \text{otherwise} \end{cases}$$

Fix the error in such a way that the compiler stops complaining about `fib`.

c) Once the problem with the `fib` identifier is solved, Dafny complains as below. Explain how you would fix the error.

```
exam1.dfy(7,2): Error BP5003: A postcondition might not
hold on this return path.
```

## Exercise 2. [Java Modelling Language; 3 points]

a) Consider the following excerpt of a Java class with JML annotations.

```

public int pub;
private int priv;
/*@ requires i <= pub && i <= priv;
private void priv1 (int i) {...}
/*@ requires i <= pub && i <= priv;
public void pub2(int i) { ... }
```

Explain what happens when compiling the above program with `jmlc`. If you think that the compilation process fails suggest changes to the program so that it may compile without problems.

**b)** Consider the following function defined on the positive numbers.

$$f(n) = \begin{cases} (n^2 + 1)/2 & \text{if } n > 0 \text{ and } n \text{ is even} \\ n^2/2 & \text{if } n > 0 \text{ and } n \text{ is odd} \end{cases}$$

Write the signature of a Java method that implements the function; add a JML contract that describes the behavior of the method.

**c)** When defining a subtype of a Java type (class or interface) we often redefine methods. The Java language allows, under certain conditions, to change the types of one or more parameters and the type of the return value. Similarly, JML contracts in a subclass may change with respect to those in the superclass. Explain how JML implements the notion of behavioural subtyping. Do not forget to discuss invariants, pre- and post-conditions.

**d)** Consider the following excerpt of a Java class describing a multiset of integer elements.

```
class Multiset {
 /**
 * Creates a multiset with the elements present in a
 * given array.
 */
 Multiset (int [] v) { ... }
 /**
 * The number of elements in this multiset.
 */
 int size () { ... }
 /**
 * The smallest element in this multiset.
 * The multiset must not be empty.
 */
 int smallest () { ... }
 /**
 * The largest element in this multiset.
 * The multiset must not be empty.
 */
 int largest () { ... }
 /**
 * The number of occurrences of a given integer in
 * this multiset.
 */
 int occurrences (int n) { ... }
 /**
 * Remove an element from this multiset. The operation
```

```
 * has no effect if n is not in the multiset.
 */
void remove (int n) { ... }
/**
 * Add an element to this multiset.
 */
void add (int n) { ... }
}
```

Write a *property-based* specification for the class. Clearly identify the observer and the mutator methods. In case you need extra operations (say, on integer values or on arrays of integer values), add their signatures to the class. You need not write the body of any method.

#### Exercise 4. [Model Checking; 3 points]

The railroad crossing problem consists of three processes: a train, a crossing controller, and a gate. The controller should close the gate on receipt of a signal indicating that a train is approaching and only opens the gate once the train has signalled that it has crossed the road.

Consider the model below where the three processes communicate via message passing.

```
1 chan train_ch = [0] of {byte};
2 #define FAR 0
3 #define NEAR 1
4 #define IN 2
5 #define STATES 3
6 active proctype train() {
7 byte position = FAR
8 do
9 :: train_ch!position;
10 position = (position + 1) % STATES;
11 od
12 }
13
14 chan gate_ch = [0] of {bool};
15 #define UP true
16 #define DOWN false
17 active proctype gate() {
18 bool cmd
19 do
20 :: gate_ch?cmd -> skip
21 od
}
```

```
22 }
23
24 active proctype crossing () {
25 do
26 :: train_ch?NEAR -> gate_ch!DOWN
27 :: train_ch?FAR -> gate_ch?UP
28 :: train_ch?IN -> skip
29 od
30 }
```

For each of the three properties below:

- Use spin to prove or disprove the property;
- Change the Promela code as little as possible (and do not alter the model)
- Describe how you would check the property using the command line or one of the of the graphic interfaces. Do not forget to describe how to conclude whether the property holds or not.

Furthermore, you should solve one case using LTL and another without using LTL.

- a) “The gate is closed when the train is passing the crossing”
- b) “The gate receives a DOWN message when it is in the UP state, and receives a UP message when it is in the DOWN state”
- c) “The gate will certainly open”

**Exercise 5.** [Advanced Topics; 1 points]

Which of the following claims are false? Justify.

- a) Infer is a model checker that works directly on Java, C, C++, and Objective-C code.
- b) ThreadSafe is a static analysis tool to detect defects in concurrent code written in Java.
- c) PRISM is a tool able to detect deadlocks and unhandled exceptions in Java bytecode.
- d) UPPAL is a model checker specifically designed to verify properties of real-time systems.



**Material for the exam:** You need only a pen and your student ID card.  
Bags and backpacks, mobile phones, and coats should be left by the blackboard. Violations to this norm will invalidate your test.

**On exam writing:** Start each new group on a new page.

**Duration:** Two hours and thirty minutes.

**Exercise 1.** [Hoare Logic; 2.5 points]

The following program calculates the integer division of  $x1$  by  $x2$ .

```
1 y1 := 0;
2 y2 := x1;
3 while y2 >= x2 do
4 y1 := y1 + 1;
5 y2 := y2 - x2
```

a) Show that the program ends in a state satisfying condition

$x1 = y1 \times x2 + y2 \wedge y2 \geq 0 \wedge y2 < x2$  when starting from any state that satisfies condition  $x1 \geq 0 \wedge x2 > 0$ .

b) Show that the above program terminates under the given pre-condition.

c) Consider the following Hoare-style proof rules. Some of them are not sound, allowing one to prove incorrect properties about programs. Find which proof rules below are not sound and explain.

$$\frac{\{\varphi'\} S \{\psi\} \quad \varphi' \rightarrow \varphi}{\{\varphi\} S \{\psi\}} \quad \frac{\{\varphi\} S1 \{false\}}{\{\varphi\} S1;S2 \{\psi\}} \quad \frac{\{\varphi \wedge p\} S \{\phi_1\} \quad \{\varphi \wedge \neg p\} S \{\phi_2\}}{\{\varphi\} S \{\psi_1 \vee \psi_2\}}$$

**Exercise 2.** [Dafny; 2.5 points]

The program below performs a linear search on an array of integer values.

```
1 method find (a: array<int>, key: int) returns (index: nat)
2 requires a != null;
3 ensures 0 <= index <= a.Length;
4 ensures index < a.Length ==> a[index] == key;
5 {
6 index := 0;
7 while index < a.Length && a[index] != key
8 invariant forall k :: 0 <= k < index ==> a[k] != key; {
9 index := index + 1;
```

```
10 }
11 }
```

a) Write an Hoare triple corresponding to method `find`. Assume a version of the Hoare Calculus equipped with the necessary array operations.

b) For the above program the Dafny compiler complains with the following error message:

```
exam2.dfy(7,2): A postcondition might not hold on
this return path.
```

Explain how you would fix the error.

c) What happens when the key is not in array `a`? Add a post-condition for the effect.

### Exercise 3. [Java Modelling Language; 3 points]

A backpack is a fixed-sized data structure. The size of the data structure is not given by the number of elements it holds (which can be potentially unlimited), but by the sum of the size of its elements, as in a conventional backpack. This means that the elements that may be placed in a backpack must have a fixed size. For example, a backpack of 30 litres can hold a number of goods whose total volume can not exceed 30 litres.

The elements that may go in a backpack are of the following type.

```
1 interface Good {
2 int size();
3 }
```

The operations we are interested in can be summarised by the following excerpt of a Java class,

```
1 public class Backpack {
2 Backpack (int capacity) { ... }
3 int capacity () { ... }
4 int freeSpace () { ... }
5 boolean contains (Good e) { ... }
6 boolean fits (Good e) { ... }
7 void add (Good e) { ... }
8 void remove (Good e) { ... }
9 }
```

where the capacity of a `BackPack` is fixed upon creation and does not change, the `fits` predicate says whether the backpack has room for a given

good, method `add` puts a good in the backpack and is defined only when the good fits in, and method `remove` removes a good from a backpack if the good is in, otherwise the method has no effect.

**a)** Write a *property-based* specification for the class. Clearly identify the observer and the mutator methods. You need not write the body of any method; you must not add more constructors or observer methods.

**b)** Add a method that removes from a backpack all goods with size greater than a given size. Write its contract in terms of the observer methods identified above.

**c)** Consider the following code in JML

```
1 // @ ensures \result == 3;
2 int test() { return abs(3); }
3 // @ ensures \result >= 0;
4 int abs(int x) { return x > 0 ? x : -x; }
```

and in Dafny:

```
1 method test () returns (v: int) ensures v == 3 {
2 v := abs(3);
3 }
4 method abs (x: int) returns (y: int) ensures y >= 0 {
5 if x > 0 {y := x;} else {y := -x;}
6 }
```

Compare the behaviour of the `jmlc` compiler with that of the Dafny compiler when run on the code above. Describe what happens in *each* method. Then explain what happens when one executes the code generated by the compiler (if and when compilation succeeds).

#### Exercise 4. [Model Checking; 3 points]

Consider the following two-process mutual exclusion algorithm. Each process can be in one of three code regions: the *noncritical region*, the *trying region*, and the *critical region*. Initially, both processes are in their noncritical regions. Processes cycle through the three regions, in the order stated, as follows:

- A process indicates that it wants to enter its critical region by first entering its trying region;
- If one process is in its trying region and the other is in its noncritical region, the process can immediately enter its critical region;

- If both processes are in their trying regions, the value of a shared boolean variable `turn` is used to determine which process enters its critical region. If the value of `turn` is **false**, then process 0 can enter its critical region and turn the value of `turn` to **true**. If the value of `turn` is **true**, then process 1 can enter its critical region and turn the value of `turn` to **false**;
- Processes must eventually leave their critical regions.

a) Write a promela model of the mutual exclusion algorithm.

b) For each of the three properties below describe how you would check the property using the command line or one of the of the graphic interfaces. Do not forget to describe how to conclude whether the property holds or not.

- The two processes cannot be in their critical regions at the same time.
- A process which wants to enter its critical region will eventually be able to do so.
- It is impossible to reach a state in which both processes are trying to enter their critical sections, but no process succeeds.

#### Exercise 5. [Advanced Topics; 1 points]

Which of the following claims are false? Justify.

- a) Slam is a model-checking tool to check properties of device-drivers written in the C programming language.
- b) Uppal is Hoare-logic based tool specialised in the verification of real-time multi-threaded software.
- c) Infer is a tool to identify bugs in Java and C code based on separation logic.
- d) ThreadSafe is a static analysis tool that focus on the detection of concurrency bugs in Java code, based on specifications provided by programmers.





**Material for the exam:** You need only a pen and your student ID card.  
*Bags and backpacks, mobile phones, and coats should be left by the blackboard. Violations to this norm will invalidate your test.*

**On exam writing:** Start each new group on a new page.

**Duration:** Two hours and thirty minutes.

**Exercise 1.** [Hoare Logic; 2.5 points]

**a)** The sum of the first  $n$  natural numbers is  $n(n+1)/2$ . What is the sum of the first  $n$  even numbers?  $n(n+1)/4$ ? Not really, but you can show the following. Do it!

```
{n ≥ 0}
s := 0;
i := 0;
while i < n do
 i := i + 2
 s := s + i
{s ≥ n(n+1)/4}
```

**b)** Prove that the above program terminates under the given pre-condition.

**c)** While loops test their guards at loop entry. In contrast **repeat-until** loops test the guard at the end. As consequence, the body of a **repeat-until** loop is executed at least once. Furthermore, the **until** keyword suggests that the loop is exited when the condition becomes false, in contrast with while loops. Then, a program

```
repeat S until p
```

can be understood as

```
S; while !p do S
```

Suggest a rule for the repeat command. Justify your choice.

**Exercise 2.** [Dafny; 2.5 points]

The following method, written in the Dafny programming language, computes the largest value in a given array of integers.

```

method ArrayMax (a: array<int>) returns (z: int) 1
 ensures forall i :: 0 <= i < a.Length \Rightarrow z >= a[i] 2
{ 3
 z := a[0]; 4
 var i := 1; 5
 while i < a.Length 6
 { 7
 if a[i] > z { z := a[i]; } 8
 i := i + 1; 9
 } 10
} 11

```

a) Explain, in terms of Hoare triples, the sort of verification performed by the Dafny compiler when method `ArrayMax` is compiled.

b) For the above program, the Dafny compiler complains with two error messages:

- 1) Index out of range (4, 10)
- 2) A postcondition might not hold on this return path.  
Related location 1: Line 2, Col: 12 (6, 4)

Explain the meaning of the each error, why they happen, and how can the method can be adjusted in order to eliminate the errors.

c) Explain why Dafny does not report any termination-related problem.

### Exercise 3. [Verifast; 3 points]

a) Modular verification in the presence of aliasing is a challenging task. Verifast uses an ownership regime and sees the heap as a multiset of heap chunks. Explain the meaning of an assertion of the form  $C_f(o,v)$ .

b) Consider the Java class below annotated with Verifast assertions.

```

class C { 1
 int a; boolean b; 2
 int getA () 3
 //@ requires a != null; 4
 //@ ensures true; 5

```

```

{ return a; } 6
boolean getB () 7
 //@ requires b |-> ?v; 8
 //@ ensures b |-> v; 9
{ return b; } 10
public static void main (String [] args) 11
 //@ requires true; 12
 //@ ensures true; 13
{ 14
 C f4 = new C(); 15
 f4.getA(); 16
 C f5 = f4; 17
 f5.getA(); 18
} 19
int m(int i) 20
 //@ requires i >= 0; 21
 //@ ensures result >= i; 22
{ return i; } 23
} 24

```

Concentrate on method main. Would Verifast issue an error? What if we replace getA by getB in both lines 16 and 18? Explain.

c) Consider a class D extending C that *further* allows calls to method m such that i may be lower than or equal to 0. In this case the new method should return a value that is lower or equal to i. Write the code and the contract for class D.

#### Exercise 4. [Model Checking; 3 points]

Consider the following excerpt of a promela model that computes the  $n$ -th Fibonacci number in  $\mathcal{O}(n)$  message exchanges. Recall the first numbers in the sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.

```

#define N 10 1
chan c = [2] of {int}; 2
proctype Fib () { 3
 int i; int x; int y; 4
 do 5
 :: c?x -> 6
 i = i + 1; 7
 if 8

```

```

:: i == N -> 9
 printf("The %d-th fib number is %d\n", N, x); 10
 break 11
:: else -> 12
 c?y; c!y; c!(x + y); 13
fi 14
od 15
} 16

```

- a) Write the **init** process so as to complete the model.
- b) Does process Fib terminate? How would you prove that using Spin?
- c) How would you proceed in order to show that the above model actually computes the Fibonacci of N? Explain the changes (if any) to the model, and how to check the result using Spin from the command line. Note:  $\text{fib}(10) = 55$ .
- d) What happens if we reduce the buffer size? Would the above property still hold? And what if we increase the buffer size?
- e) The first two numbers in the Fibonacci sequence are equal. Is this fact true for all two consecutive values? How would you proceed in order to show that, soon or later, the values in the channel come in strict ascending order? Again, explain the changes (if any) to the model, and how to check the result using Spin from the command line.

#### Exercise 5. [Other tools; 1 points]

- a) Hifrog is a bounded model checker for the C programming language. In which sense does a conventional model checker such as Spin differ from a bounded-model checker?
- b) Which of the following claims are false? Justify.
  - i) PRISM is a validation and verification of real-time systems modeled as networks of timed automata.
  - ii) SLAM is a model checker that works directly on Java, C, C++, and Objective-C code.
  - iii) ThreadSafe is a static analysis tool for finding concurrency bugs in Java code.



**Material for the exam:** *You need only a pen and your student ID card. Bags and backpacks, mobile phones, and coats should be left by the blackboard. Violations to this norm will invalidate your test.*

**On exam writing:** *Start each new group on a new page.*

**Duration:** *Two hours and thirty minutes.*

**Exercise 1.** [Hoare Logic; 2.5 points]

**a)** No one needs the product operator to multiply two numbers. Show that the following triple holds.

```
{n ≥ 0}
s := 0;
i := n;
while i > 0 do
 s := s + m
 i := i - 1
{s = n * m}
```

**b)** Prove that the above program terminates under the given pre-condition.

**c)** Hoare triples ensure partial correctness. Explain the meaning of this statement.

**d)** Hoare triples are often called hypothetical because  $\{\varphi\} S \{\psi\}$  makes no claim that  $\varphi$  will actually be true when  $S$  is executed. Explain.

**Exercise 2.** [Dafny; 2.5 points]

The following method, written in the Dafny programming language, decides whether two given integer arrays contain the same elements in the same order.

```
method EqualArrays (a: array<int>, b: array<int>)
```

```

returns (eq: bool) 2
ensures eq \iff a.Length == b.Length && 3
 forall i :: 0<=i<a.Length \implies a[i]==b[i] 4
{
 if a.Length != b.Length { return false; } 5
 var i := 0; 6
 eq := true; 7
 while i < a.Length 8
 { 9
 if a[i] != b[i] { return false; } 10
 i := i + 1; 11
 } 12
 return true; 13
} 14

```

a) Explain, in terms of Hoare triples, the sort of verification performed by the Dafny compiler when method `ArrayMax` is compiled.

b) For the above program, the Dafny compiler complains with an error message of the form below.

```

A postcondition might not hold on this return path.
Related location 1: Line: 3, Col: 15

```

Explain the meaning of the error, why it happens, and how can the method be adjusted in order to eliminate the error. Remember that a statement of the form `return true` in reality means `eq := true; return`.

c) Explain why Dafny does not report any termination-related problem.

### Exercise 3. [Verifast; 3 points]

The following snippet of a Java class describes a stack when implemented with an array.

```

public class ArrayStack {
 private Object[] elements;
 private int size;
 private static final int DEFAULT_CAPACITY = 10;

```

```
...
}
```

a) Predicate `stack` captures some aspects of the `Stack` representation. i) Why are predicates important in VeriFast? ii) Explain in detail the meaning of the predicate.

```
/*@
predicate stack(list<Object> elems) =
 this.size |-> ?s &*&
 this.elements |-> ?e &*&
 e[0..s] |-> elems &*&
 e[s..e.length] |-> _ &*&
 s == length(elems);
@*/
```

b) Write a constructor that accepts an integer for the initial capacity; add a VeriFast contract based on the predicate above.

c) The `peek` method returns the element at the top of the stack, leaving the stack unchanged. Write method `peek`; add a VeriFast contract based on the predicate above.

d) Consider the following two methods as part of class `Stack`.

```
void push(Object x)
 //@ requires stack(?elems);
 //@ ensures stack(append(elems, cons(x, nil)));
{ ... }
public static void main(String[] args)
 //@ requires true;
 //@ ensures true;
{
 ArrayStack s = new ArrayStack(1);
 s.push(null);
 s.peek();
 s.peek();
}
```

Would the code in method `main` compile with VeriFast? Explain.

**Exercise 4.** [Model Checking; 3 points]

Consider the following algorithm to control access to the critical regions of two processes A and B. Initially, we have  $x := 0$  and  $y := 0$ . Then processes A and B evolve as follows.

|                                                                                                                                                              |                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Process A:</b><br>Loop forever<br>Work in noncritical region<br>$y := 1; t := 1;$<br>wait until $x = 0$ or $t = 0$<br>Work in critical region<br>$y := 0$ | <b>Process B:</b><br>Loop forever<br>Work in noncritical region<br>$x := 1; t := 0;$<br>wait until $y = 0$ or $t = 1$<br>Work in critical region<br>$x := 0$ |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|

- a) Model the algorithm in Promela.
- b) Adapt the model so that Spin may prove (or disprove) that “At most one process is executing its critical section at any time” (mutual exclusion). Explain how to check the result using Spin from the command line.
- c) Adapt the model so that Spin may prove (or disprove) that “If any process is trying to execute its critical section, then eventually that process is successful” (absence of starvation). Explain how to check the result using Spin from the command line.

**Exercise 5.** [Other tools; 1 points]

ThreadSafe is a commercial static analysis tool that focuses on the detection of defects in software. For which languages is ThreadSafe available? Which class of problems does Threadsafe address? How does ThreadSafe work?