Verificação e Validação de Software (VVS)
Departamento de Informática, Faculdade de Ciências da Universidade de Lisboa
Exame de 2ª época — 22 de Junho de 2013
Duration: **2:30 + 30 min. tolerance**

Start each section (A to E) in a separate page. For each set of tests requested in the exam, do not forget to identify for each test the corresponding test requirements that are covered. Do not also forget to indicate any infeasible test requirements you find.

Consider the following Java method (indexOfMin) that computes the index of the first occurrence of the minimum value in an array of integers.

```
1  public static int indexOfMin(int[] v) {
2    if (v == null || v.length == 0) {
3      throw new IllegalArgumentException();
4    }
5    int iMin = 0;
6    for (int i = 1; i < v.length; i++) {
7      if (v[i] < v[iMin]) {
8        iMin = i;
9      }
10   }
11   return iMin;
12 }
```

A. Mutation testing [3.0 valores]

Consider the following mutations over indexOfMin:

m_1 (l. 2)	v == **null**	$\longrightarrow$	v != **null**
m_2 (l. 6)	i = 1	$\longrightarrow$	i = 0
m_3 (l. 6)	i < v.length	$\longrightarrow$	i >= v.length
m_4 (l. 7)	v[i] < v[iMin]	$\longrightarrow$	v[i] <= v[iMin]
m_5 (l. 7)	v[i] < v[iMin]	$\longrightarrow$	v[i] >= v[iMin]

(1) Characterize tests that kill the mutants defined by each mutation above, if possible. Explain your choice of tests for the killed mutants and why the remaining mutants can not be killed.

(2) For one of the killed mutants in **(1)**, characterize an additional test that executes the associated mutation but does not lead to state infection. Explain why.

(3) For one of the killed mutants in **(1)**, characterize an additional test that leads to state infection but not failure. Explain why.

B. Logic coverage [2.0 valores]

(1) Identify the predicates and clauses in indexOfMin.

(2) Identify the test requirements over indexOfMin using the CC (Clause Coverage) criterion.

(3) Write JUnit test methods that satisfy the previous test requirements.

(repeated code of indexOfMin, to avoid turning page)

```
1  public static int indexOfMin(int[] v) {
2    if (v == null || v.length == 0) {
3      throw new IllegalArgumentException();
4    }
5    int iMin = 0;
6    for (int i = 1; i < v.length; i++) {
7      if (v[i] < v[iMin]) {
8        iMin = i;
9      }
10   }
11   return iMin;
12 }
```

C. Graph coverage [3.0 valores]

(1) Draw the control flow graph (CFG) of indexOfMin.

(2) Identify all definitions, uses, and du-paths in the CFG.

(3) Characterize tests that satisfy the ADUPC ("All du-paths coverage") criterion.

(4) Explain briefly why the ADUPC criterion subsumes the AUC ("All-Uses Coverage") criterion. Present tests requirements that are in accordance with the AUC criterion for indexOfMin but not with the ADUPC criterion, and explain your choice.

D. Input space partitioning [2.5 valores]

Consider the following sketch of two characteristics and corresponding blocks to partition the input space of indexOfMin.

C1: Length of input vector
— NULL OR EMPTY: v is null or an empty array.
— LENGTH 1: The length of v is 1.
— LENGTH >= 1: The length of v is greater or equal to 1.

C2: Number of repetitions of minimum value
— NULL OR EMPTY: v is null or an empty array.
— ONE: The minimum value in v occurs only once.
— TWO: The minimum value in v occurs twice.

(1) Explain what is wrong with the above definition of characteristics and propose a valid reformulation.

(2) Take into account the reformulation you did and characterize test requirements applying the MBCC ("Multiple Base Choice Coverage") criterion with **two** base choices, **without using the** NULL OR EMPTY **blocks in the definition of those base choices.**

(3) Characterize tests that satisfy the previous test requirements.

E. Complementary aspects [2 valores]

Provide brief and clear answers to the following questions.

(1) What are integration tests?

(2) What are regression tests?

(3) In what way does program-based mutation help us "test"/refine a set of tests?