Departamento de Informática
Faculdade de Ciências
Universidade de Lisboa

Verificação Validação Software
Exame Um
23 de Junho de 2009

Inicie cada grupo numa nova página. Duração: três horas.

Grupo 1. [1,5 valores] Distinguish “defect”, “error” and “failure”. Term “bug” is often used in the computer science jargon; relate it to the previous three terms.

Grupo 2. [2 valores]

> Do invalid inputs really matter? From the perspective of program correctness invalid inputs are simply those outside the precondition of a specified function. Formally speaking, a software implementation of that function can exhibit any behavior on inputs that do not satisfy the precondition. This includes failure to terminate, run time exceptions, and “bus error, core dumps.”
>
> *Introduction to Software Testing*, Ammann and Offutt, 2008

Discuss the need to produce test cases that contain invalid inputs.

Grupo 3. [3 valores] Consider a class describing an unbounded stack of generic elements E with the usual semantics, and including the following public method signatures.

```
public Stack ();
public void push (E element);
public void pop ();
public E top();
public boolean isEmpty();
```

a) List the conditions for the inputs. Derive a set of equivalence classes.

b) Prepare a test suit. For each test specify the associated equivalence class, the driving (input) code, the expected output and the code necessary to check the result.

Grupo 4. [4 valores]

```
 /** ****************************************************
  * Finds and prints n prime integers
  * Jeff Offutt, Spring 2003
  *******************************************************/
private static void printPrimes (int n)
{
```

```
7    int curPrime; // Value currently considered for primeness
8    int numPrimes; // Number of primes found so far.
9    boolean isPrime; // Is curPrime prime?
10   int [] primes = new int [MAXPRIMES]; // The list of prime numbers.
11
12   // Initialize 2 into the list of primes.
13   primes [0] = 2;
14   numPrimes = 1;
15   curPrime = 2;
16   while (numPrimes < n)
17   {
18     curPrime++; // next number to consider ...
19     isPrime = true;
20     for (int i = 0; i <= numPrimes-1; i++)
21     { // for each previous prime.
22       if (isDivisible (primes[i], curPrime))
23       { // Found a divisor, curPrime is not prime.
24         isPrime = false;
25         break; // out of loop through primes.
26       }
27     }
28     if (isPrime)
29     { // save it!
30       primes[numPrimes] = curPrime;
31       numPrimes++;
32     }
33   } // End while
34
35   // Print all the primes out.
36   for (int i = 0; i <= numPrimes-1; i++)
37   {
38     System.out.println ("Prime: " + primes[i]);
39   }
40 } // end printPrimes
```

a) Prepare a test suit that ensures 100% branch coverage. For each test list in tabular form the branches that it covers.

b) Classify each occurrence of each parameter and local variable as **def** or **use**. Construct tables for each variable identifying they **def-use** paths. From the tables generate tests covering as many **def-use** pairs as possible.

c) Compare the two white-box approaches, considering the two generated test suits above.

d) Consider the two mutants given below

```
19 isPrime = false;
36 for (int i = 1; i <= numPrimes - 1; i++)
```

For each mutant separately, if possible, find a test case that *does not* reach the mutant.

e) For each mutant separately, find a test case that kills it.

Grupo 5. [2 valores]
a)

> The definitions for testing outline analysis objectives that relate to evaluation, (revealing) defects, and quality. We can use two approaches to achieve these objectives:
>
> 1. static analysis methods where the software artifact is examined manually, or with a set of tools, but not executed;
> 2. dynamic analysis methods where the software artifact is executed using a set of input values, and its output behavior is then examined and compared to what is expected.
>
> *Practical Software Testing*, Burnstein, 2002

Discuss the different software artifacts that are amenable to static analysis methods and to dynamic analysis methods.
b)

> A review is a group meeting whose purpose is to evaluate a software artifact or a set of software artifacts.
>
> *Practical Software Testing*, Burnstein, 2002

Summarize the main benefits of establishing a review program.