Tarea MFI, Auto-Test Primer Parcial 2017

$$\underline{\text{EVALUACIÓN}}: \quad \text{NOTA} = \frac{\text{Aciertos} \; - \frac{\text{Fallos}}{3}}{\text{Número de preguntas}} \times 10$$

1. En el Common Criteria:

   A se define un estándar de seguridad base que ha de especializarse después para dominios específicos (ferrocarriles, automoción, aeronáutica, etc)

   B se distinguen distintos niveles de seguridad, denominados *Safety Integrity Levels* (SIL)

   C se define un conjunto de criterios aceptado internacionalmente para evaluar el nivel de seguridad de productos informáticos (hardware, software y firmware)

   D los niveles de seguridad más bajos garantizan mayor confianza

2. Indica cuál de las siguientes afirmaciones es cierta:

   A Los métodos formales no se aplican a programas imperativos

   B No existe ningún producto comercial Java con componentes certificadas a un nivel más alto que el 3 del CC

   C El nivel 7 es el más alto y seguro del estándar Common Criteria (CC)

   D La confidencialidad y la inocuidad (condición de ser inofensivo) son propiedades de seguridad (en el sentido de *security*)

3. ¿Cuál de las siguientes propiedades **NO** es un atributo ecuacional en Maude?

   A asociatividad

   B conmutatitividad

   C multiplicidad

   D identidad

4. Dada la siguiente regla Maude "`crl pippo(X) => X * N if s(s(N)):= X * X`" el término de entrada `pippo(3)` se reduce a:

   A `99`

   B `63`

   C `21`

   D `3`

5. Completa el siguiente programa en Maude que define una función `capicua` que comprueba si una lista es un palíndromo:

```
fmod LISTQID is
  pr QID .      pr BOOL .

  sorts ListQid .
  subsorts Qid < ListQid .

  op nil : -> ListQid [ctor] .
  op _._ : ListQid ListQid -> ListQid [ctor assoc id: nil] .
  op capicua : ListQid -> Bool .

  var Q : Qid .      var L : ListQid .

  eq capicua(nil) = true .
  eq capicua(Q) = true .
  eq [          ] = capicua(L) .

  eq capicua(L) = false [owise] .

endfm
```

A `capicua(_ . L . _)`

B `capicua(Q . L . Q)`

C `capicua(Q L Q)`

D `capicua(L . Q . L)`

6. Completa la siguiente definición en Maude de la función `reverse` para las listas definidas en la pregunta anterior:

```
op reverse : ListQid -> ListQid .
eq reverse(nil) = nil .
eq reverse(Q . L) = [          ] .
```

A `L . Q`

B `reverse(L) . Q`

C `reverse(L) . [Q]`

D `reverse(L) (Q . nil)`

7. Indica cuál de las siguientes lógicas tiene cuantificadores no clásicos:

A lógica temporal

B lógica *fuzzy*

C lógica ecuacional

D lógica lineal

8. Indica cuál de las siguientes lógicas es una lógica modal:

   A lógica epistémica

   B lógica *fuzzy*

   C lógica ecuacional

   D lógica de orden superior

9. ¿Cuál de las siguiente técnicas **NO** se representa mediante un hiperarco en la Trilogía del Software?

   A refactorización

   B depuración declarativa o racional

   C certificación (*proof-carrying code*)

   D model checking

10. ¿Cuál de las siguientes propiedades **NO** es una característica del model checking?

    A mejora el rendimiento del programa

    B rápido

    C automático

    D el usuario no necesita realizar demostraciones matemáticas

11. Indica cuál de las siguientes **NO** es una técnica de transformación de programas:

    A fragmentación (*slicing*)

    B refactorización

    C derivación formal de programas

    D evaluación parcial o especialización de programas

12. Indica cuál de los siguientes tipos de herramientas genera contraejemplos:

    A intérprete simbólico

    B certificador de programas

    C model checker

    D analizador sintáctico

13. ¿Cuál de las siguientes parejas de procesos formales operan en sentidos "opuestos" dentro del esquema de la trilogía del software (es decir, uno de los procesos lleva de componentes de tipo A a B, mientras el otro lo hace de tipo B a A)?:

    A análisis estático - análisis semántico

    B generación de juegos de datos - generación de escenarios

    C síntesis de especificaciones a partir del código - derivación formal de programas

    D transformación automática de código - prototipado

14. En el esquema de la Trilogía del Software, la técnica de *model checking* se representa usando varios arcos. Indica cuál de los siguientes arcos no aparece en dicha representación:

    A de programas a especificaciones

    B de programas a programas

    C de programas a datos

    D de especificaciones a datos

15. Indica cuál de las siguientes afirmaciones es **FALSA**:

    A El lenguaje JML es parte del estándar UML

    B Tanto en JML como en OCL la evaluación de las aserciones no causa efectos laterales

    C Los lenguajes JML y OCL permiten especificar pre/post-condiciones e invariantes

    D Las especificaciones JML se escriben como comentarios anotados dentro del código fuente Java

16. Indica cuál de los siguientes resolutores de *constraints* se utiliza en el intérprete simbólico PEX y en el verificador deductivo Dafny (ambos de Microsoft):

    A Z3

    B CVC4

    C Yices

    D CORAL

17. Indica cuál de las siguientes no es una utilidad ofrecida por la herramienta PEX:

    A interpretación simbólica

    B runtime assertion checking

    C análisis de código

    D generación de casos de prueba

18. Elige la opción correcta en relación a la siguiente restricción JML (que afirma que, cuando termina la ejecución del bucle `for`, no hay ninguna componente nula en el *array* `a`):

```
...for (n = 0; n < a.length; n++)
     if (a[n]==null) break;
/*@ assert (\forall int i; 0 <= i && i < n;
                              a[i] != null);
  @*/
```

A La aserción es cierta

B En general, dicha aserción es falsa ya que no se cumple si se entra a ejecutar `break` y el bucle termina abruptamente

C El aserto causa que, en cada iteración del bucle, JML compruebe si hay alguna componente nula en el *array* `a`

D Se trata de un invariante de bucle que afirma que el *array* `a` contiene componentes no nulas en todo momento de la ejecución

19. Dado el siguiente fragmento de código:

```
void testme (int x, int y) {
           int v[y] ;
           for (int i = 0; i < y; i++) {
            v[i] = 0;
           }
           if (x > y)
              System.out.println(v[x]);  // Execution ERROR: ArrayIndexOutOfBound!
}
int main() {
           int x = sym_input();
           int y = sym_input();
           testme (x,y);
           return 0;
}
```

indica cuál de los siguientes pares se obtendría en una rama del árbol de ejecución simbólica del método main:

A `< x = y, 0 >`

B `< x > y, 0 >`

C `< x ≤ y, 0 >`

D `< !(x > y), ERROR >`

20. En el contexto de las metodologías de desarrollo ágiles, es habitual (y resulta especialmente útil) disponer de una herramienta de:

A refactorización

B *theorem proving*

C *proof-carrying code*

D ninguna de las anteriores

21. Indica cuál de las siguientes lógicas tiene cuantificadores no clásicos:

    A lógica lineal

    B lógica *fuzzy*

    C lógica de reescritura

    D lógica dinámica

22. Indica cuál de las siguientes lógicas es una lógica modal:

    A lógica de reescritura

    B lógica *fuzzy*

    C lógica ecuacional

    D lógica dinámica

23. ¿Cuál de las siguientes propiedades **NO** es una característica del model checking?

    A el usuario no necesita realizar demostraciones matemáticas

    B rápido

    C automático

    D reduce el tamaño del código

24. ¿Cuál de las siguiente técnicas se representa mediante un hiperarco en la Trilogía del Software?

    A refactorización

    B inferencia de programas

    C transformación de modelos

    D model checking

25. Estrictamente, la evaluación parcial de programas no debería considerarse como una técnica de transformación de programas pura, en el sentido de que requiere dos tipos de entradas (en vez de sólo una), que son:

    A programas y datos

    B especificaciones y programas

    C datos y especificaciones

    D dos programas

26. Completa el siguiente programa Maude que define la ordenación de listas de naturales de forma 100 % declarativa:

```
fmod SORT is
   include NAT .
   sort ListNat . subsort Nat < ListNat .
   op nil : -> ListNat .
   op __ : ListNat ListNat -> ListNat [assoc id: nil] .

   vars NL1 NL2 NL3 : ListNat .
   vars M N : Nat .

   op  sort : ListNat -> ListNat .
  ceq [          ] = sort(NL1 N NL2 M NL3) if N < M .
   eq sort(NL1) = NL1 [ owise ] .
endfm
```

A `sort(M N NL1 NL2 NL3)`

B `sort(NL1 NL2 NL3 M N)`

C `sort(M NL1 NL2 NL3 N)`

D `sort(NL1 M NL2 N NL3)`

27. Asume que añadimos un nuevo operador "`op foo : ListNat -> Nat`" al programa de ordenación de la pregunta anterior, definiéndolo con la siguiente regla:

$$\texttt{crl foo(N L1)} \Rightarrow \texttt{M if (M NL2)} := \texttt{sort(N L1)}.$$

En el nuevo programa, el término de entrada `foo(3 6 2 8)` se reescribe a:

A 2

B 3

C 6

D 8

28. Indica cuál de las siguientes afirmaciones es **falsa** en relación a la certificación del software y los estándares de seguridad:

A En España, las empresas interesadas en certificar la calidad de un producto software deben someter éste al dictámen de AENOR (Asociación Española de Normalización y Certificación)

B El Common Criteria distingue distintos niveles de seguridad denominados *Evaluation Assurance Levels* (EAL)

C Los niveles de seguridad más altos del Common Criteria garantizan mayor confianza

D No existe un conjunto de criterios aceptado internacionalmente para evaluar el nivel de seguridad de productos informáticos

29. ¿Cuál de las siguientes propiedades **NO** es un atributo de operador en Maude?

A trans

B ctor

C ditto

D iter

30. Indica cuál es el significado de la función "pippo" definida en Maude sobre multiconjuntos de números naturales:

```
fmod NAT-MSET is
  protecting NAT .
  sort NatMSet .
  subsort Nat < NatMSet .
  op mt : -> NatMSet [ctor] .
  op _ _ : NatMSet NatMSet -> NatMSet [assoc comm id: mt] .
  op pippo : NatMSet -> Nat .
  vars N M : Nat.    var S : NatMSet.
  eq pippo(N N S) = pippo(N S).
 ceq pippo(N M S) = pippo(N S) if N < M .
  eq pippo(N) = N .
endfm
```

A pippo elimina duplicados del multiconjunto

B pippo computa el elemento más pequeño del multiconjunto

C pippo ordena el multiconjunto

D pippo devuelve el primer elemento del multiconjunto

31. Indica cuál de las siguientes es una técnica de transformación de programas:

A síntesis de programas

B fragmentación

C model checking

D transformación de modelos

32. Indica cuál de las siguientes herramientas tiene, en la Trilogía del software, un arco que llega a datos:

A model checker

B certificador de programas

C resolutor de restricciones aritméticas

D analizador sintáctico

33. La lógica temporal es un tipo de lógica:

A dinámica

B *fuzzy*

C modal

D de orden superior

34. En la Trilogía del Software se representan distintas técnicas de síntesis (o inferencia) que conectan dos componentes de la trilogía; por ejemplo, las que:

A derivan juegos de datos a partir de programas

B analizan propiedades a partir de programas

C generan contraejemplos a partir de modelos

D aprenden programas a partir de ejemplos

35. Indica cuál de las siguientes técnicas **NO** es 100 % automática:

A análisis estático

B run-time checking

C model checking

D theorem proving

36. El SMT solver CORAL se usa en:

A el analizador Java PathFinder

B el model checker NuSMV

C el intérprete simbólico PEX

D el evaluador parcial Indus

37. Indica cuál de las siguientes afirmaciones es cierta:

A A nivel industrial, y en particular de las grandes compañías del sector TIC, no existe ningún interés en el avance de los métodos formales

B El compilador del lenguaje Clight (subconjunto de C) está escrito en el lenguaje del demostrador de teoremas Coq y su corrección ha sido probada también en Coq

C El Common Criteria (CC) es un estándar internacional ISO/IEC

D Las propiedades de seguridad (en el sentido de *security*) se formulan en términos de que ciertos eventos no pueden suceder

38. Elige la opción correcta en relación a la siguiente restricción JML que afirma que, cuando termina la ejecución del bucle `for`, no hay ninguna componente nula en el (segmento visitado del) *array* `a`:

```
...for (n = 0; n < a.length; n++)
     if (a[n]==null) break;
/*@ assert (\forall int i; 0 <= i && i <= n;
                              a[i] != null);
  @*/
```

A En general, dicha aserción es falsa ya que no se cumple cuando se ejecuta el `break` y el bucle termina abruptamente

B El aserto causa que, en cada iteración del bucle, JML compruebe si hay alguna componente nula en el *array* `a`

C Se trata de un invariante de bucle que afirma que el *array* `a` contiene componentes no nulas en todo momento de la ejecución

D La aserción es cierta

39. Dado el siguiente fragmento de código:

```
void testme (int x, int y) {
  int v[] = new int[y];
  for (int i = 0; i < y; i++) {
    v[i] = 0;
  }
  if (x > y) System.out.println(v[x]);  // ArrayIndexOutOfBound Error
}

int main() {
  int x = sym_input();
  int y = sym_input();
  testme (x,y);
  return 0;
}
```

indica cuál de los siguientes pares se obtendría en una rama del árbol de ejecución simbólica del método main:

A `< x = y, 0 >`

B `< x > y, 0 >`

C `< x ≤ y, 0 >`

D `< !(x > y), ERROR >`

40. ¿Qué está especificando la siguiente restricción en JML para el método `contratar` que utiliza un campo `num_empleados` de una clase `Empresa`?

```
//@ ensures num_empleados = \old(num_empleados) + 1;
public void contratar() { num_empleados = num_empleados + 1; }
```

A el campo `num_empleados` antes de la ejecución y después de la ejecución se diferencian en el valor entero `1`

B el campo `num_empleados` se incrementa en uno después de la ejecución del método `contratar`

C al empezar la ejecución del método, el campo `num_empleados` debía ser mayor que 0

D el campo `num_empleados` debe ser mayor que 0 al terminar