Métodos Formais em Engenharia de Software
Teste, 13 de Janeiro de 2023

Número: _____ Nome: _____

1. SAT *(4 pontos)*

Considere que tem um conjunto de cubos de cores: amarelos, vermelhos e brancos. Queremos empilhar 3 cubos respeitando as seguintes regras:

1. Em cada posição da pilha tem de estar um único cubo.
2. Os 3 cubos têm que ter cores diferentes, excepto se o cubo de baixo for branco.
3. O cubo que fica no topo nunca pode ser branco.

Codifique este problema em lógica proposicional. Assinale claramente o que denota cada variável proposicional que introduzir, e escreva um conjunto de fórmulas proposicionais adequado à sua modelação.

Métodos Formais em Engenharia de Software
Teste, 13 de Janeiro de 2023

Número: _____ Nome: _____

2. SMT *(2 pontos)*

Considere o seguinte programa C sobre inteiros:

```
x = x + y;
y = x - y;
x = x - y;
```

1. Faça a codificação lógica deste programa na teoria de inteiros.
2. Diga, justificando, como poderia usar um SMT *solver* para verificar que este programa faz a troca de valores entre as variáveis x e y. (Não precisa de usar a sintaxe específica do SMT *solver*).

Métodos Formais em Engenharia de Software
Teste, 13 de Janeiro de 2023

Número: _____ Nome: _____

3. Modelação estrutural com Alloy *(4 pontos)*

Considere o seguinte modelo de um sistema de gestão de conferências.

```
open util/ordering[Date]
sig Person {
  attends : set Conference
}
sig Conference {
  organizers : some Person,
  location : one Location,
  date : one Date
}
sig Location {}
sig Date {}
```

Especifique as seguintes propriedades *(para cada um dou duas possíveis soluções)*:

a) Todas as conferências tem que ter alguém a assistir.
b) Os organizadores de uma conferência tem que assistir à mesma.
c) Uma pessoa só pode assistir a duas conferências no mesmo dia se decorrerem no mesmo local.
d) Não é possível haver conferências no mesmo local em dias consecutivos.

Métodos Formais em Engenharia de Software
Teste, 13 de Janeiro de 2023

Número: _____ Nome: _____

4. Modelação comportamental com Alloy *(4 pontos)*

Considere o seguinte conceito de notificação. Um utilizador pode subscrever eventos sobre os quais pretende ser notificado sempre que acontecem. Quando um evento ocorre todos os subscritores são notificados do mesmo. Um utilizador pode a qualquer momento ler todas as notificações que tem pendentes (sempre que tiver alguma notificação).

```
sig Event {}
sig User {
  var subscriptions : set Event,
  var notifications : set Event
}
pred stutter { ... }
pred occur [e : Event] { ... // Occurrence of an event }
pred read [u : User] { ... // Read all notifications }
pred subscribe [u : User, e : Event] { ... // Subscribe an event }
pred unsubscribe [u : User, e : Event] { ... // Unsubscribe from a event }
fact {
  no subscriptions and no notifications
  always {
    stutter or (some e : Event | occur[e]) or (some u : User | read[u]) or
    (some u : User, e : Event | subscribe[u,e] or unsubscribe[u,e])
  }
}
```

a) Especifique as ações `read` e `occur`.
b) Especifique os seguintes princípios operacionais:
   (1) se um evento ocorre duas vezes de seguida a segunda ocorrência já não gera notificações adicionais;
   (2) depois de ler as suas notificações um utilizador só as pode voltar a ler depois de ocorrer algum evento que ele subscreva.

Métodos Formais em Engenharia de Software
Teste, 13 de Janeiro de 2023

Número: _____ Nome: _____

5. Why3 *(6 pontos)*

Nas questões que se seguem poderá utilizar funções existentes na biblioteca do tipo list, em particular:

```
function num_occ (x: 'a) (l: list 'a) : int
```

a) Considere a seguinte função sobre listas de inteiros. Escreva uma ou mais pós-condições que descrevam o seu comportamento relativamente ao número de ocorrências de cada elemento.

```
let rec function remove (x:int) (l:list int) : list int
  ensures  { . . . }
= match l with
  | Nil -> Nil
  | Cons h t -> if h=x then t
                else Cons h (remove x t)
  end
```

Métodos Formais em Engenharia de Software
Teste, 13 de Janeiro de 2023

Número: _____ Nome: _____

b) O módulo seguinte destina-se a implementar o conceito de notificação da Questão 4. O estado será um record com dois campos: `subscribes` é um map de eventos para listas de utilizadores (contendo os utilizadores que subscrevem cada evento), e `notif` é um map de utilizadores para listas de eventos (contendo os eventos notificados a cada utilizador, e ainda não lidos).

```
module Label
  use bool.Bool   use int.Int   use list.ListRich
  type user   type event

  clone fmap.MapImp as Subs with type key = event
  type subsmap = Subs.t (list user)

  clone fmap.MapImp as Notif with type key = user
  type notifmap = Notif.t (list event)

  type state_type = { subscribes : subsmap ; notif : notifmap }
    invariant { . . . }
    by { . . . }

  val state : state_type
  ...
```

Escreva um ou mais invariantes de tipo apropriados para o tipo `state_type.` Deverá incluir os seguinte factos: um utilizador não tem notificações de eventos que não subscreva; cada utilizador ocorre no máximo uma vez em cada lista de subscritores de eventos; utilizadores que subscrevam algum evento devem constar no domínio do map de notificações.