Vitor Hugo Klein

RA:2577895

Prova 3

Questão 1 – Cálculo da pré-condição mais fraca

$\{x > 0\}$

```
if (x < y) then
    z := y – x;
else
    z := x - y;
```

$\{ z \geq 0 \land (x = y \Rightarrow z = 0) \}$

wp(if (x < y) then z := y - x else z := x - y, $z \geq 0 \land (x = y \Rightarrow z = 0)$)

wp = $(x < y \Rightarrow (y - x \geq 0 \land (x = y \Rightarrow y - x = 0))) \land$
$(x \geq y \Rightarrow (x - y \geq 0 \land (x = y \Rightarrow x - y = 0)))$

Como x > 0 e as subtrações são não negativas, a pré-condição mais fraca é satisfeita.

Questão 2 – Cálculo da pré-condição mais fraca

$\{x > 0\}$

```
a := x + 1;
b := a * 2;
c := b - 4;
```

$\{ (c > 0 \Rightarrow x \geq 2) \land (c = 0 \Rightarrow x = 1) \}$

wp(c := b - 4, Q) -> (b - 4 > 0 => x ≥ 2) ∧ (b - 4 = 0 => x = 1)
wp(b := a * 2, ...) -> ((a * 2 - 4 > 0 =>x ≥ 2) ∧ (a * 2 - 4 = 0 => x = 1))
wp(a := x + 1, ...) -> (((x + 1) * 2 - 4 > 0 =>x ≥ 2) ∧ ((x + 1) * 2 - 4 = 0 => x=1))

$(2x - 2 > 0 \Rightarrow x \geq 2) \wedge (2x - 2 = 0 \Rightarrow x = 1)$

Como $x > 0$ implica essa expressão, o programa é correto.

Questão 3 – Especificações JML

(a) Especificação JML para `ehQuadradoPerfeito`:

```
/*@ requires n >= 0;
  @ ensures \result <==> (\exists int i; i*i == n);
  @*/
boolean ehQuadradoPerfeito(int n);
```

(b) Especificação JML para `ehMatrizIdentidade`:

```
/*@ requires matrix != null &&
  @         (\forall int i; 0 <= i && i < matrix.length; matrix[i] != null &&
matrix[i].length == matrix.length);
  @ ensures \result <==>
  @        (\forall int i; 0 <= i && i < matrix.length;
  @          (\forall int j; 0 <= j && j < matrix.length;
  @            (i == j ==> matrix[i][j] == 1) &&
  @            (i != j ==> matrix[i][j] == 0)));
  @*/
boolean ehMatrizIdentidade(int[][] matrix);
```

Questão 4

a) Um problema de decisão é aquele que pode ser respondido com "sim" ou "não", para cada entrada.

b) Um problema é decidível se existe um algoritmo que resolve corretamente todas as entradas; é indecidível se não existe tal algoritmo.

c) A classe de linguagens Recursivas contém linguagens para as quais existe uma máquina de Turing que sempre termina (aceita ou rejeita).

d) A classe de linguagens Recursivamente Enumeráveis contém linguagens para as quais existe uma máquina de Turing que aceita as palavras da linguagem, mas pode não parar para outras.

e) A técnica da Redutibilidade transforma um problema em outro, sendo usada para provar indecidibilidade ao mostrar que um problema indecidível pode ser reduzido ao problema em questão.