Lista de Exercícios N.03 e 04 (Valor: 5 pontos) Entrega: Domingo, 19 de junho de 2022 às 23:59

1. (Slides de análise semântica 5 a 14) Considerando o código abaixo, quais das opções denotam linhas que estão "'ligadas"' entre si? Lembre-se da regra de declaração aninhada mais próxima.

```
1   class Foo {
2     f(x: Int): Int {
3       {
4         let x: Int <- 4 in {
5           x;
6           let x: Int <- 7 in
7             x;
8           x;
9         };
10        x;
11      };
12    };
13    x: Int <- 14;
14  }
```

( ) Linha 5 liga-se com a linha 2

( ) Linha 8 liga-se com a linha 6

( ) Linha 10 liga-se com a linha 2

( ) Linha 10 liga-se com a linha 13

2. (Slides de análise semântica 19 a 28) Em regras de inferência 'boas', sempre que é provável que *'e'* é do tipo *'T'*, *'e'* deve sempre ser avaliado para um valor de tipo *'T'*. Sabendo disso, escolha as regras de inferência abaixo que são consideradas 'boas':

( ) $$\frac{\begin{array}{c}\vdash e_1 : T_1 \\ \vdots \\ \vdash e_n : T_n\end{array}}{\vdash \{e_1; ...; e_n;\} : T_n}$$ Sequência

( ) $$\frac{\vdash e_1 : Int \quad \vdash e_2 : Int}{\vdash e_1 < e_2 : Int}$$ Comparação

( ) $$\frac{\vdash e_1 : Int \quad \vdash e_2 : Int}{\vdash e_1 / e_2 : Bool}$$ Divisão

( ) $$\frac{\vdash e_1 : T_1}{\vdash is\_void(e_1) : Bool}$$ Vazio

3. (Slides de análise semântica 33 a 36) Considere a regra de inferência a seguir, e escolha as substituições corretas para O_1 e O_2 :

$$\frac{\begin{array}{c}O_1 \vdash e_1 : T_1 \\ O_2 \vdash e_2 : T_2\end{array}}{\vdash let\ x : T_1 < -e_1\ in\ e_2 : T_2}$$ Let-Init

( ) $O_1 = O[T_1/x]$ $O_2 = O[T_1/x]$

( ) $O_1 = O[T_1/x]$ $O_2 = O[T_2/x]$

( ) $O_1 = O$ $O_2 = O[T_1/x]$

( ) $O_1 = O$ $O_2 = O[T_2/x]$

4. (Slides de análise semântica 37 a 40) Considere as definições de classes a seguir, e marque, à direita, quais dos 'limites superiores mínimos (lub)' são verdadeiros.

```
class Object
class Bool inherits Object
class Point inherits Object
class Line inherits Object
class Shape inherits Object
class Quad inherits Shape
class Circle inherits Shape
class Rect inherits Quad
class Square inherits Rect
```

( ) `lub(Point,Quad)=Object`

( ) `lub(Square,Rect)=Quad`

( ) `lub(Square,Rect)=Rect`

( ) `lub(Square,Circle)=Object`

5. (Slides de análise semântica 41 a 45) Dadas as definições de classes abaixo, com a declaração de método, quais são os tipos válidos para as variáveis na seguinte chamada:

```
z <- x.setCenter(y)
```

```
class Object
class Bool inherits Object
class Point inherits Object
class Line inherits Object
class Shape inherits Object {
  setCenter(p: Point): Bool {
    ...
  };
  ...
};
class Quad inherits Shape
class Circle inherits Shape
class Rect inherits Quad
class Square inherits Rect
```

( ) x: Rect; y: Object; z: Bool

( ) x: Circle; y: Point; z: Bool

( ) x: Object; y: Object; z: Object

( ) x: Shape; y: Point; z: Bool

6. (Slides de análise semântica 47 a 49) Dado o código abaixo, escolha os pares de tipos estáticos/dinâmicos que estão coretos. Assuma, para o tipo dinâmico, que a execução tenha terminado na linha 14.

```
1  class Animal {...}
2  class Pet inherits Animal {...}
3  class Cat inherits Pet {...}
4  class Dog inherits Pet {...}
5  class Lion inherits Animal {...}
6  class Main {
7    w: Animal <- new Animal;
8    x: Animal <- new Pet;
9    y: Animal <- new Pet;
10   z: Pet <- new Pet;
11   w <- new Lion;
12   y <- new Dog;
13   z <- new Cat;
14   ...
15 };
```

	Var	Tipo estático	Tipo dinâmico
( )	w	Animal	Lion
( )	x	Animal	Pet
( )	y	Pet	Dog
( )	z	Pet	Pet

7. (Slides de análise semântica 50 a 53) Dada as definições de classes a seguir, escolha as relações de subtipos que são verdadeiras. Lembre-se das regras para subtipos com SELF_TYPE.

```
class Object
class Bool inherits Object
class Point inherits Object
class Line inherits Object
class Shape inherits Object
class Quad inherits Shape
class Circle inherits Shape
class Rect inherits Quad
class Square inherits Rect
```

( ) $Square \leq SELF\_TYPE_{Shape}$

( ) $SELF\_TYPE_{Circle} \leq Quad$

( ) $SELF\_TYPE_{Shape} \leq Shape$

( ) $SELF\_TYPE_{Rect} \leq Shape$

8. Considere os trechos de código abaixo escritos em COOL:

```
1     class A {
2         x: A; -- line 2
3         baz(): A {{x <- new A; x;}}; -- line 3
4         bar(): A {new A}; -- line 4
5         foo(): String {"COMPILADORES!"};
6     };
7     class B inherits A {
8         foo() : String {" "};
9     };
10    class C inherits A {
11        foo() : String {"A melhor disciplina: "};
12    };
13    class Main {
14        main (): Object {
15            let io : IO <- new IO, b : B <- new B, c : C <- new C in
                 {{
16                io.out_string(c.baz().foo());
17                io.out_string(b.baz().foo());
18                io.out_string(b.bar().baz().foo());
19            }}
20        };
21    };
```

(a) Da forma como está, qual é a saída desse código? (b) Observe as linhas de 2 a 4, e veja como esse trecho pode ser alterado para que a saída do programa seja 'A melhor disciplina: COMPILADORES!'.

```
1     class Main {
2         main (): Object {
3             let io : IO <- new IO, x : Int <- 20 in {{
4                 io.out_int (x);
5                 let x : Int < - 1 in {{
6                     (* x <- SEU CODIGO ;*)
7                     io.out_int (x);
8                 }};
9                 if x == 21 then
10                    io.out_string("x")
11                else
12                    io.out_int(x)
13                fi;
14            }}
15        };
16    };
```

É possível trocar (* x < − SEU CODIGO ;*) na linha 6, por uma linha de código contendo uma atribuição para x que faça o código imprimir '2021x'? Se não for possível, explique.
