Universidade Estadual de Maringá – Departamento de Informática
11917/31 – Programação Funcional – Informática
Prof. Marco Aurélio

Trabalho 02

1) Em um campeonato de atletismo os atletas são separados nas categorias: Menores, Sub23 e Adultos. O custo de inscrição de uma equipe depende da categoria da equipe e da quantidade de atletas. Para os Menores, a inscrição de cada atleta custa R$ 30, para os Sub23, R$ 40, e para os Adultos, R$ 50. A inscrição para equipes com mais de 10 atletas tem 10% de desconto.

   a) Projete um tipo de dado para representar a categoria de uma equipe.

   b) Projete um tipo de dado para representar a inscrição de uma equipe.

   c) Projete uma função que calcule o custo de inscrição de uma equipe.

2) Para implementar um editor de linha (um editor de texto que permite editar apenas uma linha por vez), é preciso representar o estado do editor, isto é, qual é a linha que está sendo editada e qual é a posição do cursor na linha. Para evitar estados inválidos (a posição do cursor ficar fora da linha), você desenvolveu a seguinte representação:

```
/// O estado de um editor de linha.
/// - esquerda é o conteúdo da linha a esquerda do cursor
/// - direita é o conteúdo da linha a direita do cursor
type Editor {
    Editor(esquerda: String, direita: String)
}
```

   A partir essa representação é possível implementar diversos comandos de edição. Por exemplo, o comando para inserir o caractere "o" transformaria o estado `Editor("Exempl", " de teste")` para `Editor("Exemplo", " de teste")`, e o comando para mover o cursor para o início da linha transformaria `Editor("Exemplo", " de teste")` para `Editor("", "Exemplo de teste")`.

   a) Projete um tipo de dado para representar um comando de edição que pode ser: mover o cursor uma posição para a direita, mover o cursor uma posição para a esquerda ou inserir um caractere qualquer após o cursor.

   b) Implemente uma função que receba como entrada o estado do editor e um comando de edição, e atualize o estado do editor executando o comando.

Referência

```
/// Devolve uma substring de *s* começando em *inicio* e pegando os próximos
/// *tam* caracteres ou até o fim de *s*, o que vier primeiro.
/// Se *inicio* é negativo, começa a partir do fim de *s*.
/// Se *tam* é negativo, devolve "".
fn string.slice(s: String, inicio: Int, tam: Int) -> Int

fn string_slice_examples() {
  check.eq(string.slice("funcional", 2, 3), "nci")
  check.eq(string.slice("funcional", 2, -1), "")
  check.eq(string.slice("funcional", 3, 50), "cional")
  check.eq(string.slice("funcional", -3, 2), "na")
}
```