QUESTÃO 2 - TDD

Dado as sentenças abaixo, marque V para àquelas que são verdadeiras, ou F para as falsas. **(1.0 PT)**

[**F**] Podemos dizer que o exemplo a seguir é um bom exemplo de TDD?

> *Recebemos um código legado bastante grande de um projeto anterior, desenvolvido sem nenhum teste, e refatoramos o mesmo, criando testes. È iniciado pelo desenvolvimento de testes triviais, passando por testes simples, testes de unidade, até chegar em testes maiores, com o objetivo de nos certificarmos de que o código funciona e posteriormente permitir a evolução e manutenção desse código.*
>
> ***COMENTÁRIO: Isso não é TDD, isso é apenas teste. O código já estava pronto, não houve desenvolvimento de baby steps de acordo com o ciclo de TDD: 1 criar o teste e ver o mesmo falhar, 2 implementar a solução e fazer o teste passar, e 3 refatorar se necessário.***

[***V***] TDD supõe uma serie de ferramentas de desenvolvimento. A comparação do TDD versus um desenvolvimento não-TDD seria muito menos favorável se não existissem ferramentas e IDEs "bonitinhas" para automatizar testes, inclusive facilitar a leitura dos resultados dos mesmos, verificar rapidamente o que passou e não passou, facilitar inclusive varias refatoracoes comuns, e ferramentas de diff e controle de versão para reverter eventuais erros e/ou encontrar as últimas mudanças com data e responsável. Inclusive podemos considerar isso como uma das razoes porque o TDD demorou algumas décadas para aparecer, e não apareceu nos primórdios da computação.

COMENTÁRIO: As IDEs facilitam o TDD e o ambiente automatizando de testes é crucial para o seu desenvolvimento. Seria muito difícil fazer TDD com cartões perfurados.

[**F**] Refatorações no TDD são relativamente infrequentes, acontecem apenas quando é detectado algum erro que deve ser corrigido. Uma refatoração é sempre retrabalho e o resultado de algum erro humano.

COMENTÁRIO: Refatoração é para melhorar a leitura do código e não tem nada a ver com erro.

[***V***] Há alguns casos limite tão comuns que praticamente sempre devemos testar pelo menos vários deles, especialmente quando se usam estruturas de dados. Caso vazio, cheio, apenas um elemento, ultimo e primeiro, usar o índice zero versus índice 1, etc. Para algumas estruturas de dados, pode também ser importante testar os casos de número de elementos par e ímpar, ou entrada ordenada e

desordenada. Quando se implementa uma pilha, por exemplo, testar pelo menos algumas dessas condições deve ser um reflexo automático para o programador TDD.

COMENTÁRIO: Casos triviais devem ser testados primeiramente, seguindo-se os Baby Steps.