


Universidade de Brasília
Departamento de Ciência da Computação
Disciplina: Métodos de Programação
Código da Disciplina: 201600

Métodos de Programação - 201600

Trabalho 1

O objetivo deste trabalho é utilizar o desenvolvimento orientado a testes (TDD) para resolver o problema de verificar o jogo da velha. As regras do jogo da velha são dadas em: <http://portaldoprofessor.mec.gov.br/fichaTecnicaAula.html?aula=28141> 

O jogo da velha é representado como uma matriz 3x3 de inteiros.

O valor 0 significa que a posição está vazia

O valor 1 significa que a posição está com um X

O valor 2 significa que a posição está com um O

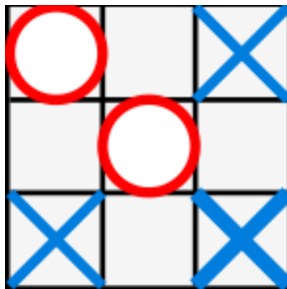
Ex. A matriz


[2, 0, 1]

[0, 2, 0]

[1, 0, 1]

Representa o jogo:



O objetivo é fazer e testar uma função que: 

Tem como parâmetro a matriz 3x3 inteiros

Retorna 1 se o vencedor foi o X

Retorna 2 se o vencedor foi o O

Retorna 0 se o jogo está empatado

Retorna -1 se o jogo está indefinido (ex. tem apenas um X)

Retorna -2 se o jogo é com certeza impossível pelas regras (ex. todas as posições são X)

O desenvolvimento deverá ser feito passo a passo seguindo a metodologia TDD. A cada passo deve-se pensar qual é o objetivo do teste e o significado de passar ou não no teste.

1) O programa deverá ser dividido em módulos e desenvolvido em C/C++ ou Python.

No caso de ser em C/C++, deverá haver um arquivo `velha.c` (ou `.cpp`) e um arquivo `velha.h` (ou `.hpp`). Deverá haver também um arquivo `testa_velha.c` (ou `.cpp`) cujo objetivo é testar o funcionamento da biblioteca de verificação do jogo da velha.

No caso de ser em Python deve ser dividido em `velha.py` (implementação) e `testa_velha.py` (teste).

2) Se for em C/C++, utilize o padrão de codificação dado em:

<https://google.github.io/styleguide/cppguide.html>

quando ele não entrar em conflito com esta especificação. O código deve ser claro e bem comentado. O código deve ser verificado se está de acordo com o estilo usando o `cpplint` (<https://github.com/cpplint/cpplint>).

Se for em Python utilize o padrão de codificação dado em:

<https://google.github.io/styleguide/pyguide.html>

quando ele não entrar em conflito com esta especificação. O código deve ser verificado se está de acordo com o estilo usando o `pylint`

<https://pypi.org/project/pylint/>

Utilize o `cpplint` ou `pylint` desde o início da codificação pois é mais fácil adaptar o código no início.

3) Faça um documento (`.txt` ou `.pdf`) dizendo quais testes você fez a cada passo e o que passar neste teste significa.

4) O desenvolvimento deverá ser feito utilizando um destes frameworks de teste:

C/C++

`gtest` (<https://code.google.com/p/googletest/>)

`catch` (<https://github.com/philsquared/Catch/blob/master/docs/tutorial.md>)

Python

`Robot` (<https://robotframework.org/>)

`PyTest` (<https://docs.pytest.org/en/7.1.x/>)

5) Deverá ser entregue o histórico do desenvolvimento orientado a testes feitos através do git (<https://git-scm.com/docs/gittutorial>)

```
git config --global user.name "Your Name Comes Here"
git config --global user.email you@yourdomain.example.com
git init
git add *
git commit -m "teste 1"
git log
```

Compactar o diretório “.git” ou equivalente enviando ele junto.

Devem ser enviados para a tarefa no aprender3.unb.br um arquivo zip onde estão compactados todos os diretórios e arquivos necessários. Todos os arquivos devem ser enviados compactados em um único arquivo (.zip) e deve ser no formato matricula_primeiro_nome.zip. ex: 06_12345_Jose.zip. Deve ser enviado um arquivo dizendo como o programa deve ser compilado e rodado.

Deve ser enviado o diretório “.git” compactado junto com o “.zip”

Data de entrega:

12/ 7 /22



Pela tarefa na página da disciplina no aprender3.unb.br