Trabalho 2 - Implementação de um jogo em Racket

1. Implementar um jogo em linguagem **Racket**.
2. Implementar 2 algoritmos que resolvam o jogo.
3. O jogo deve levar em consideração:
    a. variáveis do jogo;
    b. jogador(es), permitir definir o nome do jogador;
    c. caso seja possível: jogador versus software, multiplayer;
    d. embutir alguma inteligência no jogo;
    e. placar final com o nome do jogador;
    f. salvar lista com os 10 melhores jogadores.
4. Alguns exemplos de jogos:
   Tetris, Bejeweled, Bomberman, Columns, Same, Sokoban, Campo Minado (com gerador de campo aleatório), Corrida (estilo Enduro), PacMan, Flappy Bird, Space Invaders, Frogger, Arkanoid, Batalha Naval.
5. Escrever um mini-relatório descrevendo o jogo e realizando uma **análise** da performance dos dois algoritmos (Ex.: quem foi melhor, uso de memória, tempo de execução, etc). Obs.: a parte de análise tem grande peso na nota do trabalho.
6. Pode ser utilizado código de terceiros, como estrutura base do jogo, entretanto, deve ser indicado nas referências o site do jogo utilizado, e todo o código adicional implementado pelo aluno deve estar localizado em um arquivo de código fonte chamado **trabalho.rkt** (O aluno será avaliado pelo conteúdo existente neste arquivo).
7. O trabalho é **individual**.

Ex.: Se for implementado o Sudoku, deve ser definida uma estrutura de dados como:

```
(define board '

((  0  0  0  0  0  0  0  0  0  )
(  0  2  5  8  0  0  4  1  0  )
(  0  3  0  4  0  0  0  6  0  )
(  0  0  0  0  2  0  6  5  0  )
(  0  0  0  6  0  9  0  0  0  )
(  0  1  9  0  8  0  0  0  0  )
(  0  7  0  0  0  5  0  8  0  )
(  0  5  8  0  0  1  2  4  0  )
(  0  0  0  0  0  0  0  0  0  )))
```

o exemplo de dados anterior foi gerado por um site de Sudoku (http://www.extremesudoku.info/sudoku.html), que gerou o tabuleiro inicial em dificuldade *extreme*.

e chamado cada um dos métodos implementados para resolver o Sudoku:

(resolveSudokuAlgorimto1 board)

(resolveSudokuAlgorimto2 board)

e que devem resultar numa possível solução como:

SOLUTION:
'(4 9 6 3 1 2 8 7 5)
'(7 2 5 8 9 6 4 1 3)
'(8 3 1 4 5 7 9 6 2)
'(3 8 7 1 2 4 6 5 9)
'(5 4 2 6 7 9 1 3 8)
'(6 1 9 5 8 3 7 2 4)
'(2 7 4 9 6 5 3 8 1)
'(9 5 8 7 3 1 2 4 6)
'(1 6 3 2 4 8 5 9 7)

No cenário do Sudoku apresentado, foi testado um algoritmo X, o qual apresentou tempo de execução em torno de 1 segundo. Para questões de avaliação do algoritmo, como o algoritmo é relativamente rápido, o ideal seria executá-lo várias vezes de forma automática e avaliar o tempo médio de execução.

CRITÉRIOS DE AVALIAÇÃO:

Documentação: os códigos devem ser documentados.

Clareza lógica: a lógica nos programas deve ser coerente e fazer sentido.

Utilização das receitas de projeto: as funções devem ser escritas utilizando as receitas de projeto.

Testes: Criar testes unitários para todas as funções.

Corretude e completude: criar um teste principal que execute todos os teste unitários, e o jogo deve contemplar os requisitos definidos.

Boas práticas de programação: o código deve estar bem escrito e organizado; os recursos da linguagem devem ser usados corretamente.

Complexidade: irá avaliar o grau de complexidade do jogo e sua respectiva codificação (número de linhas de código, quantidade de funções, etc).

Paralelização: o jogo deve apresentar paralelização de código para que elementos do jogo funcionem de forma independente.

Apresentação: deverá ser realizada uma demonstração do jogo.