**PROJETO DA DISCIPLINA PARADIGMAS DE PROGRAMAÇÃO**

**Diretrizes Gerais:**

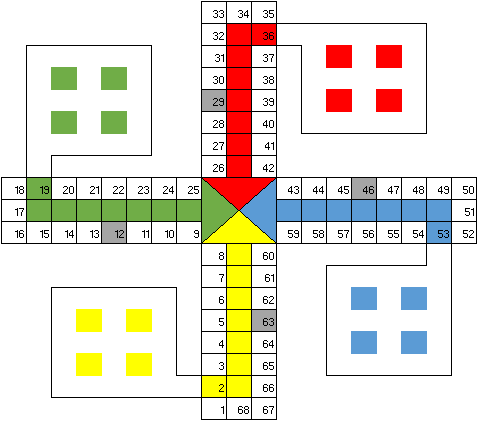
* Em quintetos
  + C# (com Unity): <link para pasta>
    - BRITNEY ISABEL GONCALVES NASCIMENTO
    - BRUNO DA SILVA BARBOSA
    - FERNANDO DE PAIVA ALMEIDA FERREIRA
    - IAN VASCONCELOS BERNARDO
    - LUCAS SANTOS ANDRADE
  + Kotlin: <link para pasta>
    - ALEXSANDRO DA SILVA LIMA
    - ANTONIO LEOMAR FERREIRA
    - GABRIEL ALVES DA COSTA E SILVA
    - MATEUS CORDEIRO VIANA
    - MATEUS SILVA SOUZA
  + PHP: <link para pasta>
    - CAMILA SABINO DOS SANTOS
    - JOSÉ KELVYN DOS SANTOS SILVA
    - LUIZ CARLOS DE LEMOS JUNIOR
    - MAIARA DA SILVA PAZ
    - ?
  + Haskell: <link para pasta>
    - CAIO VINICIUS ARAGAO DUARTE
    - JOANDERSON PEREIRA DE SOUZA
    - MIKAELLE DOS SANTOS OLIVEIRA
    - THIAGO DIAS GALDINO
    - WILTON LIMA SILVA SATURNINO
  + Go: <link para pasta>
    - LUANA VIDAL DE NEGREIROS NÓBREGA
    - MARCELO VICTOR ARAUJO PEDROSA
    - NATALIA MAITE GUIMARAES SANTOS
    - VICTOR EMANOEL MARTINS CABRAL
    - ?
* Escolher uma linguagem (multiparadigma)
  + A outra linguagem será JavaScript
* Desenvolver um jogo para praticar a programação na linguagem escolhida
  + Mesmo jogo para todos os grupos
  + Será apresentado ao final da disciplina
* Identificação dos conceitos, abordados na disciplina, na linguagem escolhida e em JavaScript
  + Serão apresentados em algumas etapas
    - Etapa 1 (final da 1ª unidade) [20/06 - Relatório completo (das 2 linguagens)]
    - Etapa 2 (final da 2ª unidade) [xx/xx]

**Especificações do Jogo:**

* Jogo definido: Ludo
* O jogo deve ter interface gráfica (mesmo que simples)
  + Interface via texto aceitável apenas caso a linguagem não possibilite facilmente desenvolver interface gráfica.
* Apresentar Top5 com, no mínimo, estas informações: nome, pontuação e data/horário da partida. OBS.: deve-se armazenar a pontuação dos jogadores (mesmo que seja em um txt).
  + A forma de pontuação deve ser definida pelo grupo (por exemplo, considerando o tempo para vencer uma partida).
  + Sugestão: buscar a possibilidade de usar o SQLite
* Dois jogadores (humano e “máquina”)
  + A “máquina” deve fazer jogadas “inteligentes”. Jogo puramente aleatório é passível de Nota Zero.
* Regras do Ludo (adaptado) a seguir...

**Objetivo de uma partida**

Cada jogador possui 4 peões da mesma cor e uma zona de início chamada: Base. O tabuleiro é composto de 68 casas numeradas (como ilustração a seguir), sendo 8 seguras (4 marcadas de cinza e as outras 4 sendo as casas iniciais), por onde podem se deslocar qualquer peão.



Além disso, cada jogador dispõe de 7 casas de chegada e uma Meta (a de maior tamanho em forma de triângulo, no centro do tabuleiro) marcadas com sua cor, onde só ele pode se deslocar.

O objetivo do jogo, ao jogar em “modo tradicional”, com uma cor (amarelo=“humano” e vermelho=“máquina”), é conseguir levar, antes que o adversário, os 4 peões da Base à Meta, passando por todo o tabuleiro. Ao jogar em “modo rápido” (amarelo&vermelho=“humano” e verde&azul=“máquina”), o objetivo é conseguir levar primeiro, da Base à Meta, um peão de qualquer uma das duas cores daquele jogador.

**Como jogar**

Nesta versão, é jogado um dado com 10 lados a cada jogada, movimentando 1 peão com o valor obtido no dado (sentido horário). Se o jogador tira o número 0 (zero) ou 9 (nove), ele poderá tirar um peão da Base.

Todos os jogadores começam o jogo com 3 peões em suas respectivas Bases e 1 em jogo, situado em sua casa de início correspondente.

Para capturar um peão de um adversário, basta cair na mesma casa que ele, desde que não seja uma casa segura. O peão “capturado” retornará à Base e o jogador que conseguiu capturar avançará com um de seus peões 10 casas, sempre que seja possível.

Se um peão conseguiu dar a volta completa ao tabuleiro poderá entrar na Meta. Para conseguir introduzir um peão na casa de Meta o jogador deverá tirar o número exato no dado, se não tem o número exato o peão não se movimenta. Na casa de chegada, um peão não poderá ser capturado pelo adversário.

Se um peão chega à Meta, o jogador deverá, obrigatoriamente, avançar 10 casas com qualquer de seus peões, sempre que seja possível.

Ganha o jogador que conseguir introduzir os 4 peões na Meta (no “modo tradicional”) ou 1 peão de alguma de suas cores (no “modo rápido”).

**Tempo por turno**

Deve-se ter um tempo limite pro jogador jogar no seu turno (sugestão: 20 segundos). Se o tempo termina, o peão do jogador se moverá automaticamente (baseado aleatoriamente nas jogadas possíveis).

**Relatório:**

* **Considerações Iniciais**
  + Classificação (nível, geração, paradigmas)
  + Breve Histórico (incluindo comentários sobre outras linguagens que a influenciaram)
  + Popularidade (em relação a rankings famosos: TIOBE, IEEE Spectrum, etc.)
  + Uso de compilador? Interpretador?
* **Conceitos**
  + Nomes
    - Caracteres especiais podem ser usados em nomes?
    - Os nomes são sensíveis à capitalização (fazem distinção entre minúsculo e maiúsculo)?
    - Qual é o tamanho máximo de um nome?
    - As palavras especiais são palavras reservadas ou palavras-chave?
  + Vinculações (*Bindings*)
    - Vinculação de tipos estática (explícita? implícita?) ou vinculação de tipos dinâmica?
    - Vinculações de Armazenamento e Tempo de Vida
      * Variáveis estáticas?
      * Variáveis dinâmicas da pilha?
      * Variáveis dinâmicas do monte (heap) explícitas?
      * Variáveis dinâmicas do monte (heap) implícitas?
  + Escopo
    - Variáveis globais? Variáveis locais?
    - Escopo estático? Escopo dinâmico?
  + Tipos de dados
    - Quais os tipos de dados suportados pela linguagem?
      * Quais são tipos primitivos e quais são tipos compostos?
      * Quais são e como se diferenciam os tipos numéricos?
      * É possível definir tipos de dados enumeráveis? E subfaixas?
      * Como a linguagem trata Strings?
        + Tipo primitivo? Tipo composto?
        + Tamanho estático? Tamanho dinâmico (limitado?)?
      * A linguagem suporta os seguintes tipos compostos? Como?
        + Produto cartesiano
        + Mapeamentos
        + Uniões disjuntas
        + Tipos recursivos
  + Checagem de Tipos
    - Tipagem estática ou dinâmica?
    - Fracamente ou fortemente tipada?
  + Equivalência de Tipos
    - Equivalência Nominal? Equivalência Estrutural?
  + Completude de Tipo
    - A linguagem adota valores de primeira classe? E valores de segunda classe?
    - A linguagem viola o Princípio da Completude de Tipo?
  + Expressões
    - Literais?
    - Agregados?
    - Chamadas de Função?
    - Expressões condicionais?
    - Expressões iterativas?
    - Acessos a constantes e variáveis?
  + Sistema de tipos
    - Sobrecarga?
    - Polimorfismo Paramétrico?
    - Polimorfismo por Inclusão?
    - Coerção?
  + Armazenamento
    - Variáveis
      * Variáveis Temporárias
      * Variáveis Persistentes
      * Storables
      * Variáveis Compostas
        + Atualização de variáveis

Atualização total? Atualização seletiva?

* + - * + Arrays

Estático? Dinâmico? Flexível?

* + - * Semântica de Cópia e Semântica de Referência
    - Comandos
      * Skip
      * Atribuição
        + Atribuição simples? Atribuição Múltipla? Atribuição Simultânea?
      * Chamada de procedimento?
      * Comando sequencial?
      * Comando colateral?
      * Comando condicional?
      * Comando iterativo?
      * Expressão de Comando (expressões com efeitos colaterais)?
      * Comando bloco e Expressão bloco?
  + Abstrações
    - Abstração de função? Abstração de procedimento?
    - Parametrização de abstrações
      * Passagem de parâmetro por cópia? (valor, resultado, valor/resultado?)
      * Passagem de parâmetro por referência?
    - Unidades de Programa
      * Pacotes?
      * Tipos Abstratos de Dados?
      * Objetos e Classes?
    - Unidades Genéricas?
  + Sequenciadores
    - Desvios Incondicionais?
    - Escapes?
    - Exceções?
* **Considerações Finais**
  + Dificuldades e Lições Aprendidas com o desenvolvimento do jogo e do relatório em geral
  + Análise da linguagem com base em critérios de avaliação
    - Legibilidade
    - Capacidade de Escrita
    - Confiabilidade
    - Custo
* **Referências**