# Trabalho de Estrutura de Dados – Jogo de Dominó

## Informações

O objetivo do trabalho é instigar e treinar os alunos sobre lógica e implementação de algoritmos que trabalham com estrutura de dados com lista duplamente encadeada.

## Regras

- **Deve ser feito em trios.**
- Valor: 40 pontos
- Data de entrega: combinado na aula com a turma.
  - A cada dia de atraso na entrega, a nota será reduzida em 5 (cinco) pontos.
- A entrega deve ser feita via Github. Cada equipe deve ter seu próprio repositório.
- A pontuação será feita da seguinte forma:
  - Funcionamento do programa.
  - Um documento explicando o funcionamento e o uso da estrutura de dados.
- Cuidado com o plágio!
- Observação: é obrigatório o uso de **lista duplamente encadeada** (cada nó possui uma referência para o nó anterior e para o nó posterior).

## Descrição do problema – Jogo de Dominó

1. O programa deve ser um jogo de Dominó.
2. O *player* 1 é o próprio computador (ramdon) e o *player* 2 é um humano.
3. Cada peça deve conter 2 números inteiros, representando os dois lados da peça.

```
public class Peca {
	private int numero1;
	private int numero2;
```

4. Todo o código que imprime algo na tela deve ser separado numa classe de **Output**.
5. O programa deve possuir 3 listas (duplamente encadeadas):
   1. uma lista com as peças do *player* do computador
   2. uma lista com as peças do *player* humano.
   3. uma lista com as peças que foram jogadas (mesa).
6. Ao iniciar, programa deve fazer a distribuição aleatória de todas peças.
7. A cada jogada feita o programa deve imprimir a lista de peças jogadas.
8. Criar uma classe separada (**Input**) para ler do teclado das opções disponíveis no jogo:
   1. Ler a peça que o humano escolhe jogar – para isso é necessário imprimir todas as peças disponíveis do humano.
   2. Escolher a opção para o humano passar a vez.
9. Caso o computador não tiver peça disponível, ele deve indicar que passou a vez.
10. O programa deve imprimir o *player* vencedor, quando chegar ao final do jogo.
11. Criar classes separadas para: Peça, Lista, Input, Output, Menu e lógica do jogo.
12. Fazer as validações necessárias para o jogo não entrar em um estado inconsistente.
13. Anotar no arquivo README.md as decisões não especificadas nesta descrição.

(Extra) Pensar em uma estratégia para deixar o computador mais “inteligente” para tentar ganhar o jogo, sem que “ele” saiba as peças que o humano possui.