**Roteiro para Execução de Trabalhos de Implementação na Disciplina de IA**

Os trabalhos de implementação da disciplina de IA serão avaliados a partir de quatro componentes (cada um valor ¼ da nota):

1. Cobertura dos testes TDD (Test-Driven-Development). Neste item já está inclusa a exigência de que será necessário criar testes. A pontuação correspondente a este item será a porcentagem da cobertura dos testes. Por exemplo, se houver 100% de cobertura e todos os testes passarem, a nota será 10.
2. Relatório de Think Aloud. Será fornecido um conjunto mínimo de questões que deverão ser respondidas aplicando a metodologia Think Aloud. A nota será de acordo com a qualidade das respostas.
3. Testes estáticos de código.
4. Métricas estatísticas de funcionamento do código. Exemplo: Acurácia, Precisão, Revocação, F-Measure. Será definido um limiar mínimo para as métricas em cada exercício. Pode haver implementações que não se enquadrem em algumas métricas. Isso será adaptado caso a caso.

**Test-Driven-Development**

* Usar [pytest](https://testdriven.io/blog/modern-tdd/) para TDD;
* Usar pytest-cov para cálculo da cobertura;

**Relatório de Think Aloud**

Protocolos de reflexão em voz alta em programação envolvem a verbalização de pensamentos e raciocínios pelo programador enquanto ele escreve ou depura o código. Isso ajuda a identificar potenciais problemas no código, entender o processo de pensamento do programador e aprimorar o processo geral de desenvolvimento. **Perguntas básicas durante uma sessão de reflexão em voz alta incluem esclarecer o problema, explicar a abordagem da solução e justificar as escolhas de design**.

**Conceitos Principais**

* Verbalização de Pensamentos. O programador expressa em voz alta seus pensamentos, observações e raciocínios durante o processo de codificação.
* Identificação de Problemas. A reflexão em voz alta ajuda a revelar potenciais erros, mal-entendidos ou áreas de confusão no código.
* Compreensão do Processo de Pensamento. Fornece insights sobre como um programador aborda um problema, toma decisões e implementa uma solução.
* Aprimoramento do Design. Ao analisar os pensamentos verbalizados, os desenvolvedores podem identificar áreas onde o design do código pode ser aprimorado para maior clareza, eficiência ou manutenibilidade.

**Sugestão de Conjuntos de Perguntas a Serem Feitas**

1. Compreensão do Problema:
   1. Qual problema estamos tentando resolver?
   2. Quais são as entradas e saídas esperadas?
   3. Existem restrições ou limitações?
2. Abordagem da Solução:
   1. Como devemos abordar este problema?
   2. Quais estruturas de dados ou algoritmos podem ser adequados?
   3. Quais são algumas abordagens alternativas?
3. Explicação do Código:
   1. O que esta linha de código faz?
   2. Por que escolhemos esta implementação específica?
   3. Quais são os potenciais efeitos colaterais deste código?
4. Depuração:
   1. Onde você acha que pode estar o erro?
   2. Como podemos testar este código para encontrar o bug?
   3. O que estamos tentando alcançar com este bloco de código específico?
5. Decisões de Design
   1. Por que escolhemos este padrão de design específico?
   2. Como este design impacta a arquitetura geral?
   3. Existem potenciais compensações com este design?
6. Esclarecimento e Reflexão:
   1. Este código atende aos requisitos?
   2. Há algo obscuro ou ambíguo neste código?
   3. O que estamos aprendendo com esse processo?

Para os trabalhos de implementação desta disciplina, cada equipe deverá selecionar um subconjunto da implementação em que teve maior dificuldade e aplicar o Think Aloud. Não é necessário fazer para o projeto inteiro, mas se quiserem fazer podem fazer. Importante: não são os testadores, nem os programadores que farão as perguntas e criarão o relatório Think Aloud, mas sim as duas pessoas responsáveis pela tarefa que atuarão como entrevistadores. Os entrevistadores deverão, se necessário, gravar o áudio das conversas e fazer transcrições (pode ser auxiliado por ferramentas speech-to-text). Além de produzir as respostas para os áudios acima, os entrevistadores deverão fornecer uma lista dos 10 itens (tópicos, sentenças, frases, problemas, soluções, etc) mais importantes ocorridos nas entrevistas.

**Testes estáticos de código**

Gerar um relatório de no máximo uma página com o resultado da aplicação do [Pylint](https://www.pylint.org) e explicações e análise dos resultados.

**Métricas estatísticas de funcionamento do código**

Quando estudarmos algoritmos que podem ser mensurados por métricas como acurácia, revocação, precisão e f1-measure. Essas métricas deverão ser aplicadas. Em outras situações, deverá ser escrito um relatório de cerca de no máximo uma página analisando os seguintes fatores: complexidade do algoritmo, tempo de processamento (com variações de carga), intervalos de confiança para o desempenho do software (quando aplicável).