TRABALHO FINAL – TF

INFORMAÇÕES GERAIS

O objetivo desse trabalho é utilizar na prática os conceitos e ferramentas de teste de software estudados durante o semestre.

- Pontuação: Essa avaliação vai **valer 50% da minha parte da nota**.
- Entrega: 17/11/2021, 17:30 (***Hard deadline***)
- Trabalho pode ser realizado sozinho, duplas ou em trios, sem possibilidade de uma configuração diferente.

ESPECIFICAÇÃO DO PROBLEMA

Vocês devem testar um programa que avalia o desempenho de diversos algoritmos de ordenamento, quais sejam: *selection sort*, *insertion sort*, *shell sort*, *heap sort* e *merge sort*.

Informações de como funcionam tais algoritmos de ordenamento, podem ser encontrados em:

- https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao/
- https://www.geeksforgeeks.org/selection-sort/
- https://www.geeksforgeeks.org/insertion-sort/
- https://www.geeksforgeeks.org/heap-sort/
- https://www.geeksforgeeks.org/merge-sort/

RECURSOS UTILIZADOS

Vocês devem utilizar como base o projeto criado disponível em https://github.com/rafaelgaribotti/TCS-TF-2021-2. Note que a função que deve ser testada é a ***sort_array*** que se encontra dentro da função *main*. Atenção: não precisam criar testes para o *quick_sort*, pois o mesmo aparenta estar instável (contém um bug). Este projeto usa **gcc** como compilador, **make** para automatizar a compilação, e usa **Travis CI** como ferramenta de *Continuous Integration*. Além disso, vocês devem utilizar também o **gcov** como ferramenta de análise de cobertura de código, além das seguintes ferramentas voltas a teste: **cppcheck**, **valgrind** e **sanitizer**.

Também devem usar uma ferramenta para descrever os testes. Eu sugiro o uso do **Unit**, já mostrado em aula, mas este ponto fica de livre escolha caso alguns grupos queiram pesquisar e usar outras ferramentas. Neste sentido, deixo a critério do grupo decidir qual ferramenta utilizar para este propósito. Algumas alternativas mais conhecidas incluem: **gtest**, **cpptest**, **catch**. Entretanto existem dezenas de outras opções.

Os alunos **podem propor ferramentas adicionais** e isso vai ser altamente valorizado na avaliação.

ENTREGÁVEIS

- Um arquivo zip com o código fonte do repositório e o relatório, **ambos postados no Moodle ANTES do prazo**.

- Os grupos devem se identificar pelo Fórum do Moodle.
- Os grupos terão que **apresentar** o seu ambiente de projeto, seus testes e os resultados.
- O repositório deve ser totalmente automatizado com make em termos de: compilação, execução do relatório de cobertura, e execução dos testes. A cada *commit* o Travis CI deve ser executado para verificar se os testes passaram.
- O relatório deve ser no formado PDF, e conter:
  - Uma tabela com as classes de equivalências e valores limites e os testes resultantes do uso destes 2 critérios.
  - Ao descrever os testes no relatório, especifiquem no seguinte formato:

Número do Teste	Nome do Teste	Casos de Teste
1	<nome do teste1>	[{entrada},{saida esperada}]
2	<nome do teste2>	[{entrada},{saida esperada}]
...	...	...
N	<nome do testeN>	[{entrada},{saida esperada}]

  - Especifiquem separadamente os testes adicionais, por exemplo, testes incluídos para aumentar a cobertura de código. Neste caso, além de adotar o mesmo formato para descrever o teste, inclua também uma frase justificando a inclusão desse teste. Por exemplo, ele cobre que parte do código? Que caso relevante que vocês perceberam que é necessário, mas que os critérios de equivalência e valor limite não contemplavam?
- Cobertura: o **gcov** suporta cobertura de linhas e de *branch*, mas como o programa é muito pequeno, quero que também considerem *cobertura de caminho* e *cobertura de predicados*. Para a cobertura de predicados, usem a técnica que mostrei em aula para montar a tabela verdade e definir quais os testes devem ser incluídos. Inclua os resultados obtidos de cobertura no relatório.
- Incluam no relatório as ferramentas adicionais de verificação utilizadas, se este for o caso.
- Evitem modificações no código fonte original. Mudanças pequenas e pontuais são permitidas com o intuito de facilitar a execução do teste. Tais mudanças se ocorrerem, devem ser relatadas no código fonte com comentários e também no relatório.

AVALIAÇÃO

- [3 pontos] Relatório e apresentação
- [2 pontos] Automação dos scripts
- [3 pontos] Qualidade e completude dos testes
- [2 pontos] Uso adequado das ferramentas indicadas

PS.: A **apresentação é obrigatória!** Logo, não apresentação do trabalho será considerado como **trabalho não entregue**.