PUC Minas

Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Curso de Engenharia de *Software*

Discipinas:	Fundamentos de Engenharia de *Software*
	Algoritmos e Estruturas de Dados I
Professores:	Ivan Luiz Vieira de Araújo
Entrega:	01/12/2024
Valor:	10 pontos

***Observações*:**

- O trabalho poderá ser feito em **grupos de até 4 alunos, desde que esses alunos estejam simultaneamente nas turmas das 2 disciplinas ou fazendo apenas 1 disciplina**.
- Cópias de trabalho receberão nota **ZERO**.
- O programa deve ser desenvolvido na linguagem de programação C.
- As informações manipuladas neste trabalho deverão ser armazenadas em arquivo(s) de **acesso direto**. Portanto, deverá ser feita leitura e escrita em arquivos com a utilização de **bibliotecas em C**.
- O trabalho deverá ser entregue pelo Canvas até o dia **01/12/2024** às **23:59 horas**.
- O grupo deve preparar uma apresentação gravada com a participação de todos os seus componentes. Essa apresentação também deverá ser entregue no Canvas em todas as disciplinas participantes do trabalho interdisciplinar.
- Deverá ser entregue o **projeto completo** do programa, a **documentação**, os **arquivos** contendo os testes realizados e a apresentação gravada em todas as disciplinas participantes do trabalho interdisciplinar.
- Em caso de dúvidas, entre em contato com seu professor.

Companhia Aérea Voo Seguro

A Voo Seguro é uma companhia aérea que visa garantir a satisfação de seus passageiros e promover sua fidelização. Ela opera em vários aeroportos pelo Brasil e conta com uma equipe de tripulação composta por pilotos, copilotos e comissários. Atualmente, o controle de voos, reservas e tripulação é realizado em planilhas e cadernos, o que tem gerado diversos problemas, como reservas duplicadas e falta de controle sobre a disponibilidade de voos e assentos. Para resolver esses problemas, a Voo Seguro contratou uma equipe de desenvolvedores de *software* (vocês) para desenvolver um sistema de gerenciamento de voos.

O Sistema

Deseja-se cadastrar passageiros, tripulação e voos da companhia aérea. As informações a serem cadastradas são:

- **PASSAGEIRO**: código, nome, endereço, telefone, fidelidade (sim/não), pontos fidelidade.
- **TRIPULAÇÃO**: código, nome, telefone, cargo (piloto, copiloto, comissário).
- **VOO**: código do voo, data, hora, origem, destino, código do avião, código do piloto, código do copiloto, código do comissário, status (ativo/inativo), tarifa.
- **ASSENTO**: número do assento, código do voo, status (ocupado/livre).
- **RESERVA**: código do voo, número do assento, código do passageiro.

Considere as seguintes **restrições**: *** Não se esqueça de sempre validar essas restrições*

Para cadastrar uma reserva, é necessário que o passageiro e o voo estejam previamente cadastrados. Reservas podem ser feitas apenas para voos com status ativo e com assentos disponíveis. A tripulação de cada voo deve conter ao menos um piloto e um copiloto para que o voo seja considerado ativo. Cada assento deve ser ocupado por apenas um passageiro em um determinado voo, e um voo não deve ter o mesmo assento reservado para mais de um passageiro.

Funcionalidades a Implementar:

1. **Cadastro de Passageiro**:
   - Deve garantir que não haja dois passageiros com o mesmo código.
   - Opcionalmente, pode-se gerar o código automaticamente.
2. **Cadastro de Tripulação**:
   - Cada membro da tripulação deve ter um cargo específico.
   - Deve garantir que não haja dois membros da tripulação com o mesmo código.
   - Opcionalmente, pode-se gerar o código automaticamente.
3. **Cadastro de Voo**:
   - Deve ser possível cadastrar informações sobre data, hora, origem, destino, tarifa, tripulação e o avião.
   - Deve verificar a presença de ao menos um piloto e um copiloto para que o voo seja marcado como ativo.
4. **Cadastro de Assento**:
   - Deve ser possível cadastrar os assentos de cada voo.
5. **Reserva**:
   - Deve garantir que o assento esteja disponível antes de reservar.
   - Reservas duplicadas para o mesmo assento no mesmo voo devem ser evitadas.
6. **Baixa em Reserva**:
   - Deve liberar o assento e atualizar o status para livre.
   - Calcular o valor total a ser pago, se necessário, de acordo com a tarifa do voo.
7. **Pesquisa**:
   - Deve ser possível buscar passageiros e membros da tripulação pelo nome ou código.

- Deve ser possível listar todos os voos de um determinado passageiro.

8. **Programa de Fidelidade**:
   - Cada voo concede 10 pontos de fidelidade ao passageiro.
   - Um passageiro pode acumular pontos ao longo de múltiplos voos.

Para implementar este programa pode ser necessário criar mais funções do que as que estão descritas.
Finalmente, implemente uma função **main()** que teste o sistema acima. A função **main()** deve exibir um *menu* na tela com as opções de cadastrar passageiro, tripulação, voo, assento, reserva, baixa em reservas, pesquisa e consulta ao programa de fidelidade. Esse *menu* deve ficar em *loop* até o usuário selecionar a opção SAIR. Além disso, todas as informações deverão ser **persistidas/armazenadas em arquivos binários** (em estruturas heterogêneas). Portanto, deverá ser implementada leitura e escrita em arquivos. Utilize **bibliotecas** para organizar os módulos (funções e procedimentos) do *software*.

****Não é obrigatória a implementação do programa usando **Orientação a Objetos**, mas caso o grupo queira usar, é permitida.*

Metodologia

Este é um trabalho interdisciplinar em que você deve planejar, analisar, projetar, implementar e testar uma solução de *software* para o problema apresentado utilizando o Scrum para gerenciar seu progresso.

Inicialmente organize o *backlog* do produto com as funções básicas do sistema. Cada um dos módulos (procedimentos ou funções) será de responsabilidade de um membro do grupo e será desenvolvido em *sprints* de 3 a 4 dias. Seguem algumas sugestões de atividades a serem realizadas nas *sprints*:

- Definir a assinatura dos módulos (procedimentos e funções) e documentar seus parâmetros. Reflita sobre os parâmetros de entrada e saída do módulo e comunique aos seus colegas de projeto.

- Documentar o módulo (procedimento ou função) indicando seu propósito, parâmetros de entrada e saída. O nome do módulo deve ser escolhido sob o ponto de vista de quem o usa ou de quem vai chamar o módulo e deve refletir o que ele faz.

- Implementar o caso de sucesso do módulo (procedimento ou função).

- Selecionar casos de teste para verificar o funcionamento de cada módulo. Um caso de teste deve conter os valores de entrada para o módulo e a saída esperada.

- Executar os casos de teste planejados para o módulo.

- Criar um relatório de execução de testes que contenha os casos de teste, a saída retornada durante sua execução e uma indicação se o módulo passou ou não no teste. Isso é feito comparando-se a saída esperada, documentada no caso de teste, com a saída retornada durante a execução do módulo (esperado x real).

- Implementar os casos especiais, exceções que possam existir no módulo (procedimento ou função). Em seguida, executar os casos de teste anteriores para garantir que as mudanças não quebraram o código anterior que já funcionava. Pense também em novos casos de teste necessários para a nova versão do módulo (procedimento ou função).

O que Deve ser Entregue

1. A evolução do *backlog* do produto a cada *sprint*. Indique quais tarefas encontravam-se inicialmente no *backlog* do produto, e em qual *sprint* cada tarefa foi alocada, juntamente com seu responsável.

2. A documentação das funcionalidades do *software*.

3. O planejamento dos casos de teste (entradas, procedimento de teste e saídas esperadas) e o relatório de execução dos testes.

4. O **código**, em C, dos módulos (funções e procedimentos) em bibliotecas e do programa principal, juntamente com o projeto completo do *software*.

5. Arquivos contendo dados já incluídos para teste das funcionalidades.

6. Apresentação gravada em vídeo (*pitch*) mostrando todas as funcionalidades do sistema.