Trabalho 1 - Gestão de um Hipermercado (Parte 1)

Desenvolva uma aplicação em C++ para gestão de um hipermercado. O sistema deve conter informação sobre produtos, funcionários, clientes e vendas.

Uma venda inclui diversos produtos e respetivas quantidades. O hipermercado efetua vendas a retalho (consumidor final) e vendas por grosso (revendedor). Numa venda a retalho, os produtos são pagos pelo seu valor de venda normal. Numa venda por grosso, o valor dos produtos sofre um decréscimo (percentagem) de acordo com a quantidade vendida.

Uma venda está associada a um funcionário.

Os clientes podem ser individuais ou empresas. Para os clientes empresa deve ser conhecido o capital social da empresa. As vendas por grosso só podem ser efetuadas a clientes empresa.

Os funcionários podem ser ou não supervisores. Um supervisor é responsável por um ou mais funcionários. Um supervisor não pode ser responsável por outro supervisor. Deve equilibrar o número de funcionários sob a responsabilidade de um supervisor.

Os produtos devem incluir, pelo menos, informação relativa a preço de venda e stock.

Alguns requisitos obrigatórios são (pode e deve incluir outros que considere relevantes):

- Manutenção de clientes
- Manutenção de funcionários
- Manutenção de produtos
- Manutenção de vendas
- Associar funcionários a supervisores
- Associar clientes a vendas
- Associar funcionários a vendas
- … outras associações que julgue necessárias

A aplicação deve permitir registar e gerir toda a informação relativa às vendas efetuadas no hipermercado e quando foram efetuadas.

Sugere-se como entidades a implementar: Cliente, Funcionario, Produto, Venda entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de clientes
- Manutenção do conjunto de funcionários
- Manutenção do conjunto de vendas
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - clientes, individuais e/ou empresas
  - funcionários, normais e/ou supervisores
  - produtos
  - vendas efetuadas, por cliente, por funcionário, por data, …
  - …

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 2 - Gestão de uma Escola de Condução (Parte 1)

Desenvolva uma aplicação em C++ para gestão de uma escola de condução. O sistema deve conter informação sobre viaturas, instrutores, alunos e aulas de condução.

A escola de condução fornece aulas de condução de veículos ligeiros, veículos pesados e motociclos. Os veículos são identificados pela matrícula, ano de fabrico, marca, data da última inspeção, data da próxima inspeção e outros atributos dependentes do tipo de veículo. A data da próxima inspeção de um veículo é dependente do ano de fabrico e do tipo de veículo e deve ser calculada pelo programa quando o veículo realiza uma inspeção.

Os instrutores são qualificados para lecionar aulas de um ou mais tipos de veículos. Procure manter uma distribuição equilibrada de alunos a instrutores.

As aulas são requeridas por um aluno para determinada data (dia/hora). A duração da aula pode ser de 1 ou 2h. Considere, para este trabalho, um período temporal de uma semana na marcação das aulas. Um aluno não pode exceder o limite de 6h de aula por semana nem 2h por dia. A aula deve ser lecionada com o veículo usual deste aluno; se o veículo estiver indisponível (oficina, outra aula,…), deve ser usado um outro veículo do mesmo tipo.

O sistema deve conter e gerir informação sobre as aulas a lecionar e quando. Deverá ainda incluir informação sobre viaturas, instrutores, alunos e aulas.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de veículos
- Manutenção de instrutores e alunos
- Manutenção de aulas
- Associar aulas a alunos
- Associar aulas a instrutores
- Associar aulas a veículos
- … outras associações que julgue necessárias

Sugere-se como entidades a implementar: Veiculo, Instrutor, Aluno, Aula, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de veículos
- Manutenção do conjunto de instrutores
- Manutenção do conjunto de alunos
- Manutenção do conjunto de aulas para um período (1 semana)
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
    - alunos (e respetivo horário)
    - instrutores (e respetivo horário)
    - veículos
    - aulas (por instrutor, por aluno, por veículo, por data)
    - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 3 - Gestão de turmas numa Escola (Parte 1)

Desenvolva uma aplicação em C++ para gestão de turmas numa escola secundária.

Considere a existência de *n* turmas. As turmas possuem uma identificação, um ano escolar e um horário semanal. Na semana, existem 4aulas de 1h para cada disciplina em dias diferentes.

Um aluno pertence a uma única turma. Os alunos são distribuídos pelas turmas de forma equilibrada.

Os professores lecionam uma única disciplina, a uma ou mais turmas de vários anos. Um professor pode ser ou não Diretor de Turma. Um Diretor de Turma é responsável por uma ou mais turmas e deve possuir um horário de atendimento semanal de 3h/turma para os estudantes das turmas pelas quais é responsável.

O sistema deve conter e gerir informação sobre as turmas e respetivos horários. Deverá ainda incluir informação sobre professores, diretores de turma e alunos.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de alunos
- Manutenção de professores e diretores de turma
- Manutenção de turmas
- Manutenção de disciplinas
- Associar turmas a alunos
- Associar aulas a professores
- Associar turmas a diretores de turma
- … outras associações que julgue necessárias

Sugere-se como entidades a implementar: Aluno, Professor, Turma, Disciplina, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de alunos
- Manutenção do conjunto de professores e diretores de turma
- Manutenção do conjunto de turmas
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - alunos (e respetivo horário)
  - professores e/ou diretores de turma (e respetivo horário)
  - disciplinas (e respetivo horário) (alunos inscritos,…)
  - turmas (e respetivo horário) , (todas, de determinado professor/aluno/ano)
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 4 – Gestão de uma Galeria d'Arte (Parte 1)

Desenvolva uma aplicação em C++ para a gestão de uma galeria d'arte. O sistema deve conter informação sobre artistas, produção artística, acervo da galeria, exposições, compra e venda de peças d'arte. Também poderá incluir outra informação que considere pertinente.

Uma galeria d'arte possui um acervo constituído por peças d'arte que podem ser propriedade da galeria ou de clientes. As peças do acervo podem estar guardadas no armazém da galeria, em exposição, para apreciação do público, ou à venda. Cada peça tem um autor, um ano de execução, e poderá ser de diferentes tipos, como pinturas, esculturas, fotografia e artesanato. Os autores das obras de arte poderão utilizar o armazém para guardar as suas peças, realizar uma exposição, com data de início e data de fim, ou colocá-las à venda, que no caso dará lugar a uma comissão para a galeria. O espaço de exposições da galeria poderá suportar várias exposições ao mesmo tempo, a depender da fração alugada por cada expositor (artista ou cliente), por exemplo, 100% do espaço, 50% do espaço, etc. A galeria também poderá realizar exposições das peças do seu cervo, de um mesmo autor ou de autores diferentes. Caso as peças pertençam a clientes ou artistas, estes deverão autorizar a exposição das suas peças nas exposições da galeria. As exposições também poderão ser temáticas ou por tipo de obra (esculturas ou pinturas, por exemplo).

Alguns requisitos obrigatórios são (pode e deve incluir outros que considere relevantes):

- Manutenção de clientes
- Manutenção de artistas
- Manutenção de peças de arte
- Manutenção de exposições
- Manutenção de compras e vendas de obras d'arte
- Associar artistas às suas obras
- Associar clientes às suas coleções
- Associar peças às exposições
- … outras associações que julgue necessárias

A aplicação deve permitir registar e gerir toda a informação relativa às vendas efetuadas na galeria e quando foram efetuadas. Deverá ser possível fazer-se o percurso de compra e venda de uma obra de arte à medida que diferentes compradores a vão adquirindo ao longo do tempo.

Sugere-se como entidades a implementar: Cliente, Artistas, Peças/obras, Exposições, Compra/Venda, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de clientes, artistas, etc.
- Manutenção do conjunto das peças no acervo (em exposição, no armazém, ou à venda)
- Manutenção do conjunto das transações de compra e venda
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - Clientes e artistas
  - Exposições
  - Acervo
  - …

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 5 – Empresa de Entrega ao Domicílio de Restaurantes (Parte 1)

Desenvolva uma aplicação em C++ para gestão de uma empresa que presta serviços de entrega ao domicílio a restaurantes. O sistema deve conter informação sobre os restaurantes associados e respetivos produtos, assim como seus clientes, que poderão ser particulares ou empresas.

A empresa possui um diretório onde apresenta os seus restaurantes associados. Um cliente, ao realizar uma compra de uma ou mais refeições, receberá o seu pedido em casa, pela empresa de entrega. A diferença entre um cliente particular e uma empresa, basicamente, está na possibilidade da empresa realizar pedidos periódicos, com uma data de início e uma data de fim para refeições como pequeno-almoço, almoço ou jantar. Cada refeição será composta por entrada, pranto principal, segundo prato principal, sobremesa, e bebidas. As refeições poderão ter produtos de diferentes restaurantes, que serão coletados antes, para a realização de uma só entrega por pedido. Ao se associar, um restaurante concorda com a empresa de entrega sobre o valor da comissão a pagar por seus serviços, que poderá ser fixo ou relativo ao valor da encomenda. Restaurantes e clientes deverão ter informações como nome, morada, NIF/NIC, etc. Os restaurantes também deverão ter os seus menus e preços respetivos dos seus pratos e bebidas. Os pedidos dos clientes só podem ser realizados a restaurantes que situam-se na mesma localidade.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de restaurantes
- Manutenção de clientes
- Manutenção de pedidos
- Associar restaurantes a menus
- Associar clientes a pedidos
- … outras associações que julgue necessárias

Sugere-se como entidades a implementar: Restaurante, Cliente, Menu, Prato, Bebida, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de restaurantes
- Manutenção do conjunto de clientes
- Manutenção do conjunto de menus, por restaurantes
- Manutenção do conjunto de pedidos, por clientes
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - restaurantes (e respetivas especialidades/cozinhas)
  - clientes (e respetivas preferências e histórico de pedidos)
  - sugestões do Chefe e pratos do dia
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 6 - Gestão de Condomínios (Parte 1)

Desenvolva uma aplicação em C++ para gestão de condomínios. Os condomínios podem ser residenciais, de escritórios e lojas, ou mistos.

A empresa mantém uma carteira de condomínios e a sua gestão consiste em receber os valores das parcelas dos proprietários, que podem ser diferentes, por condomínio, por tipo de unidade (apartamento, escritório ou loja), pela área do imóvel, e pelo andar onde se encontra. Também utiliza as receitas de cada condomínio para pagar mensalmente as despesas relativas à utilização dos espaços comuns, como água e energia. O objetivo é manter sempre as receitas positivas, relativamente aos compromissos de despesas mensais. Os valores poderão ser pagos por inteiro, ao fim de cada ano, ou parcelados por mês ou trimestre, a depender do contrato de cada proprietário de cada condomínio. A empresa também é responsável pelos serviços técnicos de manutenção, sempre que um condomínio os requisitar, mantendo assim um número de técnicos de eletricidade, alvenaria, pintura, etc. Cada um desses serviços também terão inspeção periódica, a depender do serviço (p. ex. pintura anual, eletricidade semestral, etc.). As realizações das inspeções também têm um custo associado, que deve ser incluído nas despesas do condomínio. Ao fim de cada ano, é gerado um relatório financeiro, para cada condomínio, com receitas, despesas, e valor em caixa. A aplicação deverá simular a evolução temporal por incremento de um mês ou múltiplos meses.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de condomínios
- Manutenção de moradores e proprietários de cada condomínio
- Manutenção de receitas, custos, e transações
- Manutenção dos relatórios financeiros, ao longo dos anos
- Associar proprietários a unidades do condomínio (apartamentos, lojas, escritórios)
- Associar contratos de pagamento das taxas do condomínio por proprietário
- Associar relatórios financeiros a condomínios
- Associar serviços técnicos de manutenção e inspeções periódicas a condomínios
- … outras associações que julgue necessárias

Sugere-se como entidades a implementar: Condomínio, Proprietário (morador ou não), Propriedade (apartamento, escritório, loja), entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de condomínios
- Manutenção do conjunto de proprietários
- Manutenção do conjunto de unidades por condomínio (apartamento, escritório, lojas)
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - condomínio
  - proprietários
  - serviços de manutenção realizados e agendados/previstos
  - histórico dos pagamentos realizados pelos proprietários
  - histórico das despesas realizadas para cada condomínio
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 7 – Empresa de Traduções (Parte 1)

Desenvolva uma aplicação em C++ para gestão de uma empresa de traduções. O sistema deve conter informação sobre textos a traduzir, tradutores e carteira de encomendas.

Um texto tem um identificador, o número das palavras do seu conteúdo, a língua em que está escrito e uma função para calcular a complexidade de tradução. Um texto pode ser técnico, caracterizado pelo domínio de especialidade, pode ser literário, caracterizado por ter um título e autor, e pode ser noticioso, caracterizado pelo assunto da notícia e pelo tipo de jornal onde foi publicado (diário ou semanário). Os textos literários poder ser poesia, caracterizados pelo número de quadras, ou prosa, caracterizados pelo número de páginas. A complexidade de um texto esta relacionada com o tipo de texto e o seu tamanho (número de palavras, número de quadras etc).

Um tradutor é caracterizado por ter um nome, os anos de experiência, ter uma lista de línguas que domina, saber calcular o custo de uma tradução e por estimar o tempo que demora uma tradução. Um tradutor pode ser tradutor de textos técnicos, caracterizados pela lista de domínios de especialização, pode ser tradutor de textos literários, caracterizado por ser de prosa ou poesia, ou ser tradutor de notícias, caracterizado pelo tipo de jornal. O custo de uma tradução depende dos anos de experiência do tradutor, da complexidade do texto e do número de palavras. O cálculo do tempo de tradução depende da língua original e da língua para a qual vai ser traduzido e da experiência do tradutor.

Uma encomenda deve ter um texto, a indicação da língua para a qual vai ser traduzido e a duração máxima admitida para a tradução.

O sistema deve conter e gerir informação sobre encomendas e tradutores. Deverá ainda incluir informação sobre textos, tradutores e encomendas.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de tradutores
- Manutenção de encomendas
- Manutenção de textos (incluídos numa encomenda)
- Verificar se uma encomenda pode ou não ser satisfeita
- Calcular o custo de traduzir uma encomenda
- Calcular o tempo para satisfazer a lista de encomendas
- … outras associações que julgue necessárias

Sugere-se como entidades a implementar: Text, Tradutor, Encomenda, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando estruturas de dados lineares, deve implementar:

- Manutenção do conjunto de Tradutores
- Manutenção do conjunto de Encomendas
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - tradutores (todos, que dominam determinada língua, etc)
  - encomendas (todas, com prazos inferiores um dado limite, etc)
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 8 – Gestão de um Bloco Operatório (Parte 1)

Desenvolva uma aplicação em C++ para gestão de um Bloco Operatório.

Esta aplicação dispõe de informação sobre: i) pacientes; ii) profissionais de saúde; iii) requisitos para cada cirurgia; iv) salas do bloco operatório.

Um paciente tem um nome e o tipo da cirurgia que vai realizar.

Os profissionais de saúde são médicos e enfermeiros. Os médicos podem ser anestesistas ou cirurgiões. Todos os profissionais de saúde têm um nome e um tipo de cirurgia que podem realizar. Os enfermeiros têm um custo por hora e uma especialidade. Todos os médicos pertencem a um serviço/departamento no hospital. Os anestesistas têm um custo fixo por hora. Os cirurgiões têm um custo fixo por cada cirurgia e têm uma especialidade.

Os requisitos para a realização de uma cirurgia incluem: o tipo de cirurgia (que determina os cirurgiões e enfermeiros que podem participar); o número de cirurgiões; o número de anestesistas; o número de enfermeiros; e uma lista de equipamento necessário para realização da cirurgia. Uma cirurgia só pode ser realizada se TODOS os requisitos forem satisfeitos.

Uma sala do bloco operatório tem: número de identificação; lista de equipamento disponível; e o custo da sala por hora.

Para realizar uma cirurgia deve formar-se uma equipa. A informação associada a cada equipa inclui: tipo da cirurgia; a lista de profissionais de saúde que participam na cirurgia; o tempo estimado para a sua realização; e o custo total com pessoal.

Um Bloco Operatório tem disponíveis as seguintes listas: anestesistas, cirurgiões; enfermeiros; pacientes; de salas; e uma lista com os requisitos de cada cirurgia.

O sistema deve conter e gerir informação sobre pacientes, profissionais de saúde, salas, os requisitos de cirurgias e equipas de cirurgia.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de profissionais de saúde
- Manutenção de pacientes
- Manutenção de salas
- Manutenção de requisitos de cirurgias
- Calcular a composição de uma equipa de cirurgia para um dado paciente
- Verificar se uma sala pode efetuar determinada cirurgia
- Calcular o custo associado a uma cirurgia (pessoal + sala)
- Fazer um escalonamento das cirurgias por sala mostrando a sequência de cirurgias a realizar
- … outras funcionalidades que julgue necessárias

Sugere-se como entidades a implementar: ProfissionalSaude, Paciente, Equipa, ReqCirurgia, Sala, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando estruturas ***de dados lineares***, deve implementar:

- Manutenção do conjunto de profissionais de saúde
- Manutenção do conjunto de pacientes
- Manutenção do conjunto de equipas
- Manutenção do conjunto de requisitos de cirurgias
- Manutenção do conjunto de salas
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - Profissionais de saúde (todos, médicos, enfermeiros)
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 9 – Proteção Civil (Parte 1)

Desenvolva uma aplicação em C++ para gestão de uma estrutura do tipo Proteção Civil. O sistema deve conter informação sobre tipos de acidentes, meios de socorro e sobre ocorrências de acidentes.

Um acidente tem sempre associado um nome, o local de ocorrência e um tempo máximo em que deve ser prestado o socorro e que é uma função do tipo de acidente. Inclui ainda uma função para calcular o número de socorristas necessários. Os acidentes podem ser incêndios, assaltos ou acidente de viação. Um incêndio é caracterizado pelo número de bombeiros necessários, calculado com base no tipo de incêndio, e pode ser de dois tipos: florestais ou domésticos. Os incêndios florestais são caracterizados pela área em chamas, enquanto os domésticos são caracterizados por serem em apartamento ou em moradia. Os bombeiros sabem calcular o tipo de equipamento e quantidade para cada tipo de fogo. Os assaltos são caracterizados pelo facto de haver ou não feridos e ter sido em cada particular ou numa loja. Os acidentes de viação são caracterizados pelo número de feridos graves, número de veículos envolvidos, serem em estrada nacional ou auto-estrada e se estão ou não envolvidas substâncias químicas perigosas.

Um posto de socorro é caracterizado pelo número de socorristas, pelo número de veículos e o local onde está. Dispõe ainda de uma função para calcular a distância ao local do acidente e estimar o tempo de chegada ao local do acidente. Os tipos de postos de socorro são os bombeiros, a polícia ou o INEM. Os bombeiros são caracterizados pelo número de ambulâncias, número de autotanques e ter ou não dispositivos de combate a substâncias químicas perigosas. A polícia é caracterizada pelo veículo que utiliza: moto ou carro. O INEM é caracterizado pelo tipo de veículo: ambulância, carro ou moto. Sabe-se que uma Ambulância demora mais tempo e leva 2 socorristas, um carro é mais rápido que a ambulância e leva 2 socorristas e a moto é a mais rápida mas leva só 1 socorrista.

Uma lista de ocorrências contém a descrição de acidentes que ainda não foram atendidos.

Escolha, com razoabilidade, todas as relações entre acidentes e socorristas (ex: atribuição de meios de socorro a acidentes, etc). Utilize uma tabela para calcular as distâncias entre acidente e posto de socorro.

O sistema deve conter e gerir informação sobre tipos de acidentes, meios de socorro e a lista de ocorrência de acidentes. Deverá ainda incluir informação sobre.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de postos de socorro
- Manutenção de acidentes
- Manutenção da lista de ocorrência de acidentes
- Dado um acidente, calcular um posto de socorro adequado para assistir ao acidente
- Dada uma lista de ocorrências, calcular a lista de postos de socorro capazes de satisfazerem todos os pedidos da lista
- Calcular o tempo para satisfazer cada ocorrência da lista de ocorrências
- … outras associações que julgue necessárias

Sugere-se como entidades a implementar: Acidentes, PostoDeSocorro, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando estruturas ***de dados lineares***, deve implementar:

- Manutenção do conjunto de Acidentes
- Manutenção do conjunto de PostoDeSocorro
- Listagens várias (quer totais, quer parciais com critérios a definir pelo utilizador) de, por exemplo:
  - postos de socorro (todos os bombeiros, todos com ambulâncias, etc)
  - tipos de acidente (com tempos de socorro máximo inferiores um dado limite, etc)
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático