Trabalho Prático 3 - Consultas ao Sistema Logístico

1 Descrição do problema

Os Armazéns Hanoi ficaram muito satisfeitos com o simulador de sistema logístico que você implementou e decidiram te contratar para implementar as consultas ao sistema logístico. Há basicamente duas consultas a serem implementadas:

1. **Histórico de um pacote:** Dado um identificador de pacote, apresentar um histórico de eventos associados a um pacote, detalhando, para cada evento, data-hora, tipo de evento e localização, entre outros atributos. Ou seja, a consulta retorna, um por linha, os eventos associados ao pacote até o momento da consulta.

2. **Histórico de pacotes por cliente:** Dado um cliente, apresente uma lista, um pacote por linha, na qual ele figure como remetente ou destinatário, identificando o seu papel (remetente ou destinatário), identificador de pacote, estado atual (por exemplo, armazenado no armazém X, em transporte entre os armazéns X e Y, entregue), data hora da última atualização de estado.

O sistema de consultas deve ser construído a partir de uma lista de eventos associadas aos pacotes e as consultas sendo realizadas.

- `<dh> CL <nome cliente>`: Histórico dos pacotes associados a `<nome cliente>`.
- `<dh> PC <identificador pacote>`: Histórico do pacote `<identificador pacote>`.
- `<dh> EV <identificador pacote> <tipo evento> <dados evento>`: Evento associado à tramitação de um pacote nos Armazéns Hanoi.

  No âmbito do sistema logístico, o tipo de evento pode ser um dos seguintes:

  1. `RG`: Pacote ainda não foi postado (apenas registrado no sistema)
  2. `AR`: Pacote chegou no armazém e foi armazenado na respectiva seção
  3. `RM`: Pacote foi removido da seção para tentativa de transporte
  4. `UR`: Pacote foi rearmazenado por não ser possível transportá-lo
  5. `TR`: Pacote sendo transportado entre armazéns
  6. `EN`: Pacote entregue

A Tabela 1 apresenta os atributos que devem constar de cada estado do sistema logístico. Para fins deste trabalho, você vai receber um arquivo único, contendo tanto eventos dos pacotes quanto consultas, ordenados por data hora. Um exemplo de arquivo de entrada é apresentado na Figura 1. A saída do sistema é, para cada uma das várias consultas, uma resposta no formato apresentado na Figura 2. A saída do sistema de consultas é, para cada consulta do arquivo de entrada, a resposta multi-linha para a mesma. Por exemplo, o arquivo de entrada da Figura 1 geraria como saída o arquivo apresentado na Figura 3.

2 Arquitetura

Nesta seção vamos detalhar a especificação do sistema a ser implementado neste trabalho prático.

```
0000001 EV RG 003 JOAO MARIA 000 003
0000001 EV AR 003 000 002
0000006 EV RG 000 ESTER JULIANA 001 000
0000006 EV RG 006 LUISA ANDRE 003 002
0000007 EV AR 006 003 002
0000009 EV AR 000 001 002
0000012 EV RG 005 BERNARDO LUISA 002 003
0000018 EV AR 005 002 003
0000040 EV RG 001 DAVI ANDRE 001 002
0000045 EV AR 001 001 002
0000070 EV RG 007 JOAO JOSE 000 003
0000078 EV AR 007 000 002
0000080 EV RG 002 PEDRO LUISA 000 003
0000080 EV RG 004 LUCIA RICARDO 003 002
0000092 EV AR 004 003 002
0000093 EV AR 002 000 002
0000102 EV RM 001 001 002
0000102 EV RM 002 000 002
0000102 EV RM 004 003 002
0000102 EV RM 005 002 003
0000102 EV TR 005 002 003
0000103 EV RM 000 001 002
0000103 EV RM 006 003 002
0000103 EV RM 007 000 002
0000103 EV TR 000 001 002
0000103 EV TR 001 001 002
0000103 EV TR 004 003 002
0000103 EV TR 006 003 002
0000104 EV RM 003 000 002
0000104 EV TR 003 000 002
0000104 EV TR 007 000 002
0000104 EV UR 002 000 002
0000122 EV EN 005 003
0000123 EV AR 000 002 000
0000123 EV EN 001 002
0000123 EV EN 004 002
0000123 EV EN 006 002
0000124 EV AR 003 002 003
0000124 EV AR 007 002 003
0000150 CL LUISA
0000200 PC 003
0000202 EV RM 000 002 000
0000202 EV RM 002 000 002
0000202 EV RM 007 002 003
0000202 EV TR 000 002 000
0000202 EV TR 002 000 002
0000203 EV RM 003 002 003
0000203 EV TR 003 002 003
0000203 EV TR 007 002 003
0000210 CL JOAO
0000222 EV AR 002 002 003
0000222 EV EN 000 000
0000223 EV EN 003 003
0000223 EV EN 007 003
0000300 PC 002
0000302 EV RM 002 002 003
0000302 EV TR 002 002 003
0000322 EV EN 002 003
```

Figura 1: Exemplo de arquivo de entrada

```
<consulta 1>
<n1 - numero de linhas da resposta da consulta 1>
<linha 1 da resposta a consulta 1>
...
<linha n1 da resposta a consulta 1>
<consulta 2>
...
```

Figura 2: Formato de arquivo de saída

```
0000150 CL LUISA
6
0000006 EV RG 006 LUISA ANDRE 003 002
0000012 EV RG 005 BERNARDO LUISA 002 003
0000080 EV RG 002 PEDRO LUISA 000 003
0000104 EV UR 002 000 002
0000122 EV EN 005 003
0000123 EV EN 006 002
0000200 PC 003
5
0000001 EV RG 003 JOAO MARIA 000 003
0000002 EV AR 003 000 002
0000104 EV RM 003 000 002
0000104 EV TR 003 000 002
0000124 EV AR 003 002 003
0000210 CL JOAO
4
0000001 EV RG 003 JOAO MARIA 000 003
0000070 EV RG 007 JOAO JOSE 000 003
0000203 EV TR 003 002 003
0000203 EV TR 007 002 003
0300 PC 002
7
0000080 EV RG 002 PEDRO LUISA 000 003
0000093 EV AR 002 000 002
0000102 EV RM 002 000 002
0000104 EV UR 002 000 002
0000202 EV RM 002 000 002
0000202 EV TR 002 000 002
0000222 EV AR 002 002 003
```

Figura 3: Exemplo de arquivo de saída

Atributos	RG	AR	RM	UR	TR	EN
Data Hora	1	1	1	1	1	1
Tipo evento	3	3	3	3	3	3
ID Pacote	4	4	4	4	4	4
Remetente	5					
Destinatário	6					
Armazém Origem	7				5	
Armazém Destino	8	5	5	5	6	5
Seção Destino		6	6	6		

Tabela 1: Atributos por evento no arquivo de entrada. Números indicam o campo na linha, numerado a partir de 1. Campo 2 é sempre "EV".

2.1 Entrada e Saída

O sistema a ser implementado recebe um arquivo de entrada, pela entrada padrão (`stdin`), no formato descrito. A partir dessa entrada, que deve ser lida uma única vez, ele gera as respostas para as consultas na saída padrão (`stdout`). Os eventos podem ser armazenados ordenados pela sua data hora. As consultas devem ser processadas e as respectivas respostas geradas à medida que são lidos. Vislumbramos pelo menos 3 índices para acelerar as consultas:

- **Clientes:** Permite acesso rápido a todos os eventos dos clientes.
- **Pacotes:** Permite acesso rápido a todos os eventos associados a um pacote.
- **Eventos:** Permite acesso rápido a um identificador de evento.

É uma decisão de projeto se o sistema processa os estados dos eventos à medida que lê a entrada, para facilitar a execução das consultas. Note que o índice de Clientes pode apontar para os seus Pacotes ou diretamente para os Eventos. Isso também é uma decisão de projeto. Recomenda-se a utilização de árvores balanceadas ou de mecanismos que maximizem o balanceamento, de forma que não haja risco do índice degradar em termos de desempenho, por exemplo pela inserção de valores de atributos ordenados.

2.2 Estruturas de Dados

Como mencionado, o cerne do trabalho é o projeto e a implementação das estruturas de dados, incluindo os índices, que vão permitir acesso efetivo e eficiente aos dados para a consulta. A discussão a seguir é ilustrada pela Figura 4.
Uma premissa importante é que os eventos não podem ser replicados em múltiplas estruturas de dados. Eles devem ser organizados numa estrutura única, por exemplo um vetor, organizado na ordem cronológica dos eventos.
Vamos considerar as consultas ao último estado dos pacotes de um dado cliente (`CL`). Para satisfazer essas consultas, precisamos de duas informações. A primeira informação é o conjunto de pacotes associado ao cliente. E depois temos que identificar o último pacote de cada cliente. Isso pode ser implementado a partir de um índice de clientes. O índice de clientes pode ser implementado por uma árvore, cuja chave é o nome do cliente e, para cada cliente, temos um segundo índice que lista os pacotes associados ao cliente. Esse segundo índice, organizado pelo número de pacote, possui dois apontadores “Início"e

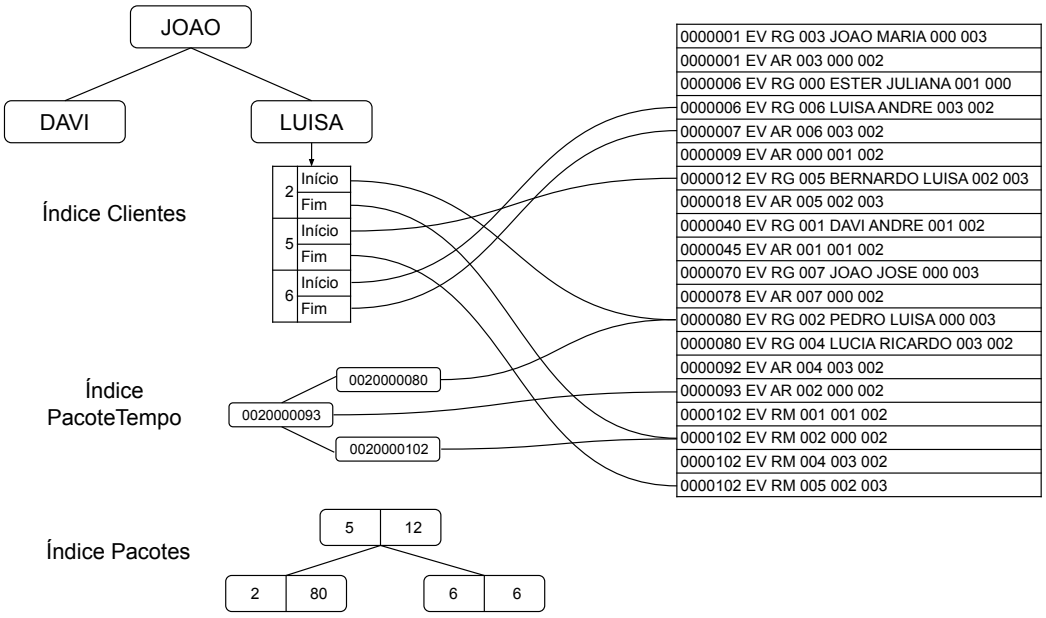

Figura 4: Exemplo de Índices e Base de Eventos

"Fim", que apontam respectivamente para o primeiro e último evento registrado até o momento. Assim, por exemplo, considerando a consulta `0000102 CL LUISA`, queremos identificar o estado dos pacotes nos quais a cliente LUISA apareça como remetente ou destinatário, que é o último estado do pacote até o tempo 102. Há 3 pacotes associados à referida cliente e o índice de pacotes aponta para os eventos de registro de cada pacote (Início) e o último evento associado a cada pacote. Note que, embora esse índice torne a resposta à consulta trivial, ele tem que ser atualizado a cada novo evento associado ao pacote.

Um segundo exemplo de índice registra os pacotes e os seus tempos iniciais, e vamos denominá-lo "Índice Pacotes". Cada nó desse Índice Pacotes especifica um pacote (que é a chave) e o tempo quando ele foi registrado (note que o registro contem informações importantes como remetente, destinatário, armazém de origem e armazém de destino. O evento de registro é fundamental para tornar o caminhamento nos eventos associados ao pacote mais eficiente.

Um terceiro exemplo de índice, complementar ao segundo, é "Índice PacoteTempo", que possui uma chave composta identificando o pacote e o tempo de evento e congrega todos os pacotes em um único índice, em contraponto a um índice por pacote. O uso desse índice pode ser ilustrado se tivermos a consulta `0000102 PC 002`, que solicita o histórico do pacote 002 até o momento 102, inclusive. Nesse caso, acessamos inicialmente o índice de pacotes, onde obtemos a informação que o registro do pacote 2 foi no tempo 80. Com essa informação podemos construir a chave a ser pesquisada no índice PacoteTempo que no caso vai ser `0020000080`. Realizamos então uma pesquisa por essa chave e, a partir da localização do nó associado, iniciamos um caminhamento inorder pelos eventos do pacote 2 (que vão aparece de forma contígua no caminhamento) até que o pacote indicado no índice seja outro. Com isso, geramos todos os eventos a serem reportados como resposta à consulta.

Os índices apresentados são apenas ilustrativos e é parte do trabalho o projeto, a implementação e a avaliação das estruturas de dados utilizadas.

2.3 Processamento de Consultas

Cada tipo de consulta tem finalidade diferente e portanto dever ser processada de forma diferente. Entretanto, sob a perspectiva de processamento, elas são simples pelo fato de compreenderem apenas um critério de consulta, que é o identificador de pacote ou o identificador de cliente. A utilização de índices deve tornar esse processo mais eficiente. O processamento de consultas gera uma lista contendo as informações que devem constar da resposta.

2.4 Gerador de respostas

A geração de respostas apenas formata o resultado do processamento de consultas. Isso pode ser importante porque desacopla o processamento de consultas da geração das saída, que depende do formato, o que pode mudar com mais frequência.

3 Análise Experimental

Considerando a natureza das consultas a serem implementadas, sugerimos que a análise experimental avalie o impacto de variações nas seguintes dimensões:

Número de clientes

Número de pacotes

Número de eventos, que é variado pela topologia dos armazéns e contenção por transporte.

Em termos de experimentação, sugerimos que seja desmembrado o tempo de leitura do arquivo de entrada do tempo de processamento do mesmo. Ou seja, o arquivo deve ser lido para um armazenamento em memória auxiliar inicialmente e depois é processado linha a linha. Desta forma, as medições não serão dominadas pelos custos de leitura da memória secundária.

4 Pontos Extra

Os pontos extra tem por suportar eficiente outras consultas, o que provavelmente vai demanda um ajuste nos índices para que a consulta seja executada eficientemente. Alguns exemplos não exaustivos de consultas são:

- Dado um intervalo de tempo, retorne os movimentos de pacotes de um armazém.
- Identifique as rotas mais congestionadas entre dois armazéns

Esteja à vontade para propor consultas que não seja resolvidas eficientemente pelos índices originais.

5 Como será feita a entrega

5.1 Submissão

A entrega do TP3 compreende duas submissões:

VPL TP3: Submissão do código a ser submetido até **07/07/25, 7:59**. **Não serão aceitas submissões em atraso.** Detalhes sobre a submissão do código são apresentados na Seção 5.3.

Relatório TP3: Arquivo PDF contendo a documentação do TP, assim como a avaliação experimental, conforme instruções, a ser submetido até **07/07/25, 7:59**. **Não serão aceitas submissões em atraso.** Detalhes sobre a submissão de relatório são apresentados na Seção 5.2.

5.2 Documentação

A documentação do trabalho deve ser entregue em formato **PDF** e também **DEVE** seguir o modelo de relatório que será postado no `minha.ufmg`. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

1. **Capa:** Título, nome, e matrícula.
2. **Introdução:** Contém a apresentação do contexto, problema, e qual solução será empregada.
3. **Método:** Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
4. **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
5. **Estratégias de Robustez:** Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
6. **Análise Experimental:** Apresenta os experimentos realizados em termos de desempenho computacional[1], assim como as análises dos resultados.
7. **Conclusões:** A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
8. **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
9. Número máximo de páginas incluindo a capa: 10

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

A documentação deverá ser entregue como uma atividade separada designada para tal no minha.ufmg. A entrega deve ser um arquivo `.pdf`, nomeado `nome_sobrenome_matricula.pdf`, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais.

[1] Para este trabalho não é necessário analisar a localidade de referência.

5.3 Código

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado. Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile:

```
– TP
    |– src
    |– bin
    |– obj
    |– include
    Makefile
```

A pasta **TP** é a raiz do projeto; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (headers) do projeto, com extensão *.h, por fim as pastas **bin** e **obj** devem estar vazias. O Makefile deve estar na raiz do projeto. A execução do Makefile deve gerar os códigos objeto *.o no diretório **obj** e o executável do TP no diretório **bin**. O arquivo executável **DEVE** se chamar **tp3.out** e deve estar localizado na pasta **bin**. O código será compilado com o comando:

```
make all
```

O seu código será avaliado através de uma **VPL** que será disponibilizada no moodle. Você também terá à disposição uma VPL de testes para verificar se a formatação da sua saída está de acordo com a requisitada. A VPL de testes não vale pontos e não conta como trabalho entregue. Um pdf com instruções de como enviar seu trabalho para que ele seja compilado corretamente estará disponível no Moodle.

6 Avaliação

- Corretude na execução dos casos de teste - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Conteúdo segundo modelo proposto na seção **Documentação**, com as seções detalhadas corretamente - (20% da nota total)
- Definição e implementação das estruturas de dados e funções - (10% da nota total)
- Apresentação da análise de complexidade das implementações - (10% da nota total)
- Análise experimental - (25% da nota total)
- Aderência completa às instruções de entrega - (5% da nota total)

Se o programa submetido **não compilar**[2] ou se compilar mas não passar em **pelo menos um caso de teste**, seu trabalho não será avaliado e sua nota será **0**. **Trabalhos não poderão ser entregues com atraso.**

[2]Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

7 Considerações finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia atentamente o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é crime. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na seção de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos.**

8 FAQ (*Frequently asked Questions*)

1. Posso utilizar qualquer versão do C++? NÃO, o corretor da VPL utiliza C++11.
2. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM, porém lembre-se que a correção é feita sob o sistema Linux, então certifique-se que seu trabalho está funcional em Linux.
3. Posso utilizar alguma estrutura de dados do C++ do tipo Queue, Stack, Vector, List, etc? NÃO.
4. Posso utilizar smart pointers? NÃO.
5. Posso utilizar o tipo String? SIM.
6. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO.
7. Posso utilizar alguma biblioteca para tratar exceções? SIM.
8. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
9. As análises e apresentação dos resultados são importantes na documentação? SIM.
10. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO.
11. Posso fazer o trabalho em dupla ou em grupo? NÃO.
12. Posso trocar informações com os colegas sobre os fundamentos teóricos do trabalho? SIM.
13. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.