Módulo III: Projeto Computacional

Estruturas de Dados - Turma B

1. Introdução

No Brasil cerca de 65% do transporte de carga é feito por caminhões. O custo de combustível dos caminhões é cerca de 70% maior que o custo para trens de carga. Devido a isso o governo Brasileiro iniciou o projeto *Futuro nas Ferrovias* para reduzir os custos, bem como melhorar o transporte de carga entre as regiões litorâneas e centrais do Brasil.

Com isso em mente, o Centro de Informática (CPD) da UnB solicitou o desenvolvimento de um *software* para o controle das cargas e vagões a serem distribuídos entre os trens em uma estação. O grupo de desenvolvimento deve possuir conhecimento de **fila** e **pilha**. Por isso, os grupos de desenvolvimento da turma de Estruturas de Dados deverão desenvolver o *software* seguindo os requisitos descritos nas seções a seguir.

2. Requisitos

Em uma estação chegam em **fila** vários trens de carga. Cada trem carrega um número máximo de vagões. Cada vagão pode carregar uma carga limitada e as cargas são caixas de diferentes pesos, como apresentado na Figura 1.

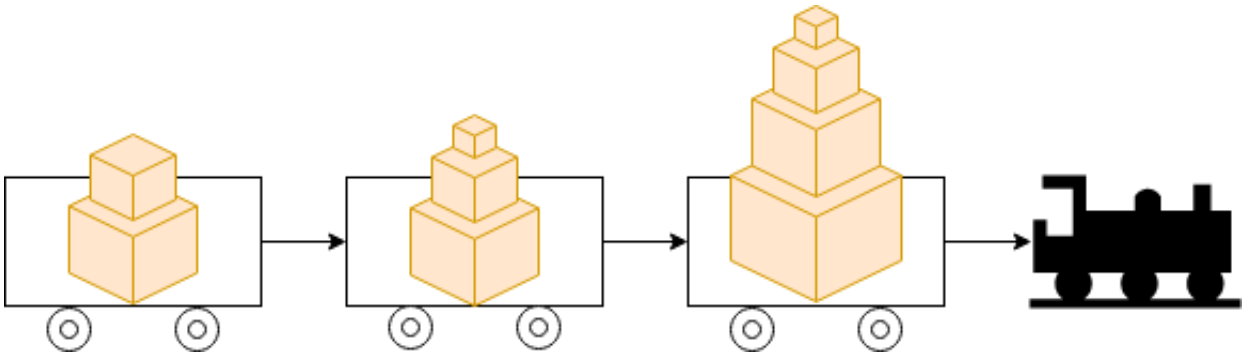

Figure 1: Exemplo de um trem de carga transportando um número máximo de vagões.

Salienta-se que somente um vagão é abastecido por vez e a estação fornece as caixas em uma ordem predeterminada (não necessariamente ordenada). As caixas devem ser colocadas de tal forma que as mais pesadas fiquem mais em baixo, formando uma **pilha** crescente de peso. Como informado anteriormente, os vagões suportam uma carga limitada e todos vagões devem ser abastecidos de tal forma que comportem a maior carga possível. A última exigência é que, ao final do abastecimento, os vagões mais pesados fiquem mais próximos do trem formando uma **lista** crescente de carga do último até o primeiro vagão (mais próximo do trem).

Enquanto o trem está recebendo a carga nenhum vagão pode ser movido. Após o abastecimento total os vagões podem ser reordenados para se adequarem aos requisitos.

3. Entrada e Saída

3.1. Entrada

A entrada é uma sequência de números descritos da seguinte maneira:

1. Quantos vagões os trens suportam.
2. Qual a carga máxima por vagão em kg.
3. Uma lista que representa a ordem e o peso de cada carga que será fornecida pela estação para os vagões

3.2. Saída

Uma das exigência para a saída do programa são os *logs*. Cada ação executada pelo sistema deve ser apresentada em tela.

3.2.1. Exemplo

Entrada:

- 2 quantidade de vagões por trem
- 1200 peso máximo por cada vagão
- 300 600 200 300 100 750 lista das cargas a serem despachadas da estação. vagões

Saída:

```
> [logger] Trem 1 entrou na estação.
>
> [Trem 1] Alocando vagão 1.
>
> [Trem 1][Vagão 1] carga de 300kg carregada.
>
> [Trem 1][Vagão 1] carga de 600kg mais pesada que a inferior desempilhando 300kg.
>
> [Trem 1][Vagão 1] carga de 600kg carregada.
>
> [Trem 1][Vagão 1] carga de 300kg carregada.
>
> [Trem 1][Vagão 1] carga de 200kg carregada.
>
> [Trem 1] Carga máxima no vagão 1 será excedida. Alocando vagão 2.
>
> [Trem 1][Vagão 2] carga 300kg carregado.
>
> [Trem 1][Vagão 2] carga 100kg carregado.
>
> [Trem 1][Vagão 2] carga 750kg mais pesada que a inferior desempilhando 100kg.
>
> [Trem 1][Vagão 2] carga 750kg mais pesada que a inferior desempilhando 300kg.
>
> [Trem 1][Vagão 2] carga de 750kg carregada.
>
> [Trem 1][Vagão 2] carga de 300kg carregada.
>
> [Trem 1][Vagão 2] carga de 100kg carregada.
>
> [Trem 1] Despache finalizado.
>
> [Trem 1] Inciando ordenação dos vagões.
>
> [Trem 1] Resultado final: TREM_1 <- 2[750, 300, 100]1150 <- 1[600, 300, 200]1100.
```

Caso ao final do primeiro trem ainda exista cargas restantes, o *software* deve alocar mais trens a medida do necessário e efetuar o mesmo procedimento.

Na última linha é necessário apresentar o *Resultado final* do trem. Esse deve ser montado da seguinte maneira:

```
TREM_T <- n[P]K <- n[P]K <- ...
```

```
- T : <Número de alocação do Trem>
- n : <Número de alocação do Vagão>
- P : <Pilha de cargas por vagão(mais a esquerda é o inferior da pilha)>
- K : <Peso total do vagão>
```

***Observação**: O número dos vagões alocados correspondem a ordem que foram alocados, ou seja, se um vagão foi alocado como vagão 2 ele deve permanecer assim no Resultado Final, mesmo após a ordenação.*

4. Material a entregar

1. Relatório com no mínimo 4 páginas, contendo:
   - **Capa**: Deve conter possuir as seguintes informações: (i) nome do software, (ii) nome dos membros do Grupo, e (iii) data de entrega.
   - **Introdução**: Descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
   - **Implementação**: Descrição detalhada da estrutura de dados utilizada com diagramas ilustrativos, o funcionamento das principais funções e procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
   - **Membros**: Descrição das atividades desenvolvidas por cada membro do grupo.
   - **Conclusão**: Comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
2. Código fonte com os seguintes arquivos:
   - **tads.h**: Cabeçalho de arquivo com todas as TADs utilizadas.
   - **tads.c**: Implementação do cabeçalho de arquivo anterior.
   - **main.c**: Programa que contém o "main" para utilizar as TADs.

O relatório e o código fonte devem ser submetidos compactados (.zip) no Moodle no dia 22 de Novembro de 2019 às 19h00min (isto é, antes de iniciar a aula). O trabalho será avaliado no laboratório (LINF3). Trabalhos entregues depois do prazo, valerão menos 1 ponto por dia de atraso.

5. Critérios de Avaliação

Trabalhos entregues depois do prazo, valerão menos 1 ponto por dia de atraso. O trabalho será pontuado de acordo com a implementação e os critérios da Tabela 5. Os pontos apresentados na Tabela 5 não são independentes entre si. Em outras palavras, alguns itens são pré-requisitos para obtenção da pontuação dos outros. Por exemplo, o item "Resultados" só é válida se também o trabalho estiver compilando e executando. Código com falta de legibilidade e modularização pode perder ponto conforme informado na Tabela 5. Erros gerais de funcionamento, lógica ou outros serão descontados como um todo.

Table 1: Critérios de avaliação

Item	Quesitos	Pontos
Relatório	Documento PDF contendo todas as informações sobre o trabalho	+2
Código e execução	O projeto compilou e executou corretamente	+2
Resultados	Saídas corretas de acordo com o solicitado no trabalho	+3
Conceitos de ED	TAD e Lista e Pilha implementados adequadamente	+3
Legibilidade e Modularização	Pode perder ponto caso não faça: -Uso de comentários -Identação do código -Uso de funções inadequadas (duplicado/redundante/não atingível) -Uso de tipos de dados definidos pelo usuário e respectivos arquivos (.c e .h)	-3
Vazamento de Memória	Perde pontos caso esqueça de liberar a memória ou acesso indevido de memória	-2
Formatação	Perde pontos caso a saída do programa não respeite a formatação sugerida neste documento	-2
Atraso	Perde 1 ponto para cada dia de atraso da entrega	-1
Plágio	Caso seja constatado plágio, ZERO no projeto	0

6. Ferramentas

A implementação do trabalho será na linguagem C. Pode-se utilizar qualquer IDE/compilador para o desenvolvimento contanto que execute sem problemas no IDE do Linux ou Windows ou Mac.

7. Informações Importantes

O trabalho deverá ser elaborado em grupo de três pessoas. Cada grupo deverá desenvolver o trabalho e cada membro do grupo deverá conhecer e dominar todos os trechos de código gerados. Os grupos deverão desenvolver o projeto de maneira independente para não haver cópia ou compartilhamento de código. O projeto irá passar por um verificador automático de plágio. Os projetos detectados como plágio receberão nota zero, independente do grupo. Dessa forma, fica a cargo do grupo proteger o projeto contra cópias ilegais.