DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
MATD04 – Estruturas de Dados
Prof. Roberto Freitas Parente
roberto.parente@ufba.br

Trabalho 02
Versão: 30 de agosto de 2019

1 Instruções:

1. O trabalho é para ser feita em até **4 (quatro) pessoas**. O nome dos integrantes devem estar no cabeçalho comentado no arquivo enviado.
2. Será aplicado detector de plágio! Uma vez detectado plágio, a **nota da disciplina** será **zerada**.
3. Data de entrega: **16 de setembro de 2018 (segunda-feira)** no sistema do moodle até as 23:50.
4. A entrega deve ser no **formato ZIP** com o seguinte conteúdo: Apenas 1 arquivo como código-fonte;
5. O código deve estar escrito em `Linguagem C` e as únicas bibliotecas permitidas são: `stdio.h`, `stdlib.h` e `string.h` caso contrário o trabalho será **invalidado**.
6. Para validação do código será utilizado um ambiente GNU/Linux com o compilador `GCC` (versão 7+) dado os seguintes parâmetros[1]:
   `gcc -Wall -Wextra -Werror -Wpedantic`
7. Para verificação de vazamento de memória e gerência dos ponteiros, será utilizado o software `Valgrind`[2].

2 Correção

Serão utilizados os seguintes critérios para correção:

2.1 Compilação

• Compilação com parâmetros requeridos.
(Não compilação com parâmetro corretos implica trabalho **invalidado**.).

2.2 Organização do código

• Funções separadas para cada uma das operações.
• Comunicação por retorno de função.

2.3 Entrada/Saída

• Cada entrada gera uma saída correta.

2.4 Gerência de memória

• Teste de alocação (verificar se a memória foi alocada).
• Remoção correta da memória alocada.
• Vazamentos de memória e controle de ponteiros.

[1] Leia mais sobre em: `https://gcc.gnu.org/onlinedocs/gcc-6.4.0/gcc/Warning-Options.html`.
[2] `http://valgrind.org`

3 Simulação de uma AGÊNCIA – PARTE II

Nosso objetivo final é desenvolver uma simulação de uma agência de banco simplificada. Tal simulação gerencia o saldo dos clientes envolvidos na simulação através de saques, transferências e depósitos realizados nos guichês da agência.

Para avançar na implementação da simulação precisamos implementar nossas estruturas de dados. Nesta etapa do desenvolvimento iremos implementar o relatório final e a fila de entrada dos clientes e integrar à implementação dos guichês de atendimento. Desta forma, deveremos ser capazes de executar simulação que serão definidas através de uma entrada, a simulação deve gerar um relatório parcial informando a situação de cada guichê e depois um relatório final da simulação geral.

3.1 A simulação

Inicialmente para cada simulação, informamos que a agência contém 3 (três) guichês e recebe N clientes durante um dia. Cada cliente atendido gera os seguintes dados: Código de pessoa física(CPF), operação, valor e CPF de terceiro envolvido na operação. Cada um dos dados tem a seguinte forma:

- CPF cliente: inteiro;
- Operação: caractere 'D' para depósito, 'S' para saque, 'T' para transferência;
- Valor: inteiro;
- CPF terceiros: inteiro;

A cada cliente recebido deve-se alocar um registro dinamicamente e enviá-lo a um guichê disponível utilizando a estratégia de round–robin iniciando do guichê 0 (zero), ou seja, se o cliente anterior foi alocado no guichê $k \mod 3$, então o presente cliente de ser alocado no guichê $k + 1 \mod 3$ ficando a seguinte ordem dos guichês selecionados: $0, 1, 2, 0, 1, 2, 0 \ldots, 2, 0, 1, 2, 0, 1, 2 \ldots$.

Cada cliente realiza apenas uma operação, mas não existe restrição da quantidade de vezes que um cliente é atendido. Cada guichê atende a um cliente por vez e armazena os documentos de cada cliente em uma pilha. Ao finalizar os atendimentos, deve ser impresso o relatório de cada guichê de acordo com a descrição na Seção 3.3. Após o relatório parcial, o sistema deve unir todos os dados referentes aos documentos, processar as transações e imprimir um resumo da ordem do dia. Tal relatório deve ser armazenado em uma lista ordenada pelo CPF.

Ao efetuar uma transação, temos os seguintes comportamentos possíveis:

- Depósito: Acrescentar o valor no saldo do cpf de terceiros (CPFt);
- Saque: Retirar o valor no saldo do cpf do cliente (CPF);
- Transferência: Retirar o valor do saldo do cpf do cliente (CPF) e acrescentar o valor no saldo do cpf de terceiros (CPFt).

Observe que a operação de saque não deve inserir o cpf de terceiros no relatório.

3.2 Entrada

A entrada é constituída por uma única simulação enviada pela entrada padrão. A primeira linha contém um inteiro N, onde $1 \leq N \leq 2^{32} - 1$, e N representa a quantidade de clientes envolvidos na simulação. As próximas N linhas deve conter os seguintes valores CPF, $CPFT$, O e V onde $1 \leq CPF \leq 2^{32} - 1$ representa o CPF do cliente, $1 \leq CPFT \leq 2^{32} - 1$ o CPF do terceiro envolvido, $O \in \{D, S, T\}$ o caractere referente a operação e $1 \leq V \leq 2^{30} - 1$ a quantidade de dinheiro envolvido na transação.

Teremos a seguinte estrutura geral da entrada:

Entrada:
N
$CPF_1\ CPFT_1\ O_1\ V_1$
$CPF_2\ CPFT_2\ O_2\ V_2$
$\vdots$
$CPF_N\ CPFT_N\ O_N\ V_N$

3.3 Saída

A saída é constituída pelo relatório parcial seguido do relatório final que deve ser enviados pela saída padrão (teclado) seguindo a formatação descrita nessa seção.

Relatório Parcial

Antes do relatório parcial deve ser impressa a mensagem "`-:| RELATÓRIO PARCIAL |:-`" e seguido de um inteiro M referente a quantidade de guichês envolvidos na simulação. Na linha seguinte deve ser impresso o texto "`Guiche` k`:` Q_k" onde k é o número do guichê e um inteiro Q_k referente a quantidade de operações empilhadas no guichê k. Nas próximas P_k linhas deve constar os dados de cada operação na formatação "`[CPFc,CPFt,O,V]`" onde "CPFc" é referente ao CPF do cliente atendido, "CPFt" o cpf do terceiro envolvido na operação, "O" o caractere referente a operação e "V" o valor envolvido na operação. A ordem e impressão dos dados armazenados em cada guichê deve ser a mesma ordem da remoção da pilha.

Relatório Final

Antes do relatório final deve constar uma linha em branca e na seguinte impressa a mensagem "`-:| RELATÓRIO FINAL |:-`". Na linha seguinte um inteiro K referente a quantidade de CPFs envolvidos na simulação e o relatório deve seguir a formatação "`-:[ CPF`$_\ell$`:` $B_\ell\ C_\ell\ C_\ell$", onde "CPF$_\ell$" um inteiro que indica o ℓ-ésimo CPF, B_ℓ um inteiro referente a quantidade de operações que envolvem o CPF$_\ell$ e C_ℓ o saldo final das movimentos referente ao CPF$_\ell$.

Observações:
`i) No relatório final deve constar TODOS os CPFs envolvidos na simulação e não apenas os CPFs dos clientes (ignorando cpf de terceiros em saques).`
`ii) Todos os clientes iniciam a simulação com saldo igual a zero.`

Saída final

Como descrito acima, seguem os seguintes parâmetros para formatação da saída:

- M: Quantidade total de guichês.
- P_k: Quantidade de documentos armazenados no guichê k.
- $CPFc_j^i$: CPF do cliente referente a j-ésima operação desempilhada do guichê i.
- $CPFt_j^i$: CPF de terceiros referente a j-ésima operação desempilhada do guichê i.
- O_j^i: Operação do j-ésima operação desempilhada do guichê i.
- V_j^i: Valor envolvido na j-ésima operação desempilhada do guichê i.
- K: Quantidade total de CPF's distintos envolvidos (cliente e terceiros).
- CPF$_\ell$: ℓ-ésimo CPF envolvido na simulação (ordenado).
- B$_\ell$: Quantidade operações que envolvendo CPF$_\ell$
- C$_\ell$: Saldo final referente ao CPF$_\ell$.

Ademais, teremos a seguinte estrutura:

Saída:

-:| RELATÓRIO PARCIAL |:-
M
Guiche 1: Q_1
[CPFc$_1^1$,CPFt$_1^1$,O$_1^1$,V$_1^1$]
[CPFc$_2^1$,CPFt$_2^1$,O$_2^1$,V$_2^1$]
⋮
[CPFc$_{Q_1}^1$,CPFt$_{Q_1}^1$,O$_{Q_1}^1$,V$_{Q_1}^1$]
⋮
Guiche M: Q_M
[CPFc$_1^M$,CPFt$_1^M$,O$_1^M$,B$_1^M$]
[CPFc$_2^M$,CPFt$_2^M$,O$_2^M$,B$_2^M$]
⋮
[CPFc$_{Q_M}^M$,CPFt$_{Q_M}^M$,O$_{Q_M}^M$,B$_{Q_M}^M$]

-:| RELATÓRIO FINAL |:-
K
-:[CPF$_1$: B_1 C_1
-:[CPF$_2$: B_2 C_2
⋮
-:[CPF$_K$: B_K C_k

4 Restrições e informações adicionais

Estruturas de dados & programação

1. Cada guichê deve ser implementado como uma pilha implementada utilizando **alocação encadeada**.
2. A lista que armazena o relatório final (ordenado por cpf) deve ser implementada utilizando **alocação encadeada sem cabeça**.
3. A fila de clientes deve ser implementada como uma fila utilizando **alocação sequêncial circular com tamanho estático**.

Funções

É importante que as funções façam suas operações independentes. Ex: não passar por parâmetro o guichê destino. Para um código bem organizado é importante que o mesmo não contenha variáveis globais e que as funções retornem valores que indiquem o seu comportamento. É bastante comum definir padrões para o retorno das funções de acordo com os possíveis comportantes da mesma. Códigos bem documentados receberão um agrado.

Por fim, espera-se que o código contenha as seguintes funções (não necessariamente com mesmo nome)

1. Lista: criar, destruir, remover, inserir e buscar;
2. Fila: criar, destruir, remover, inserir e buscar;
3. Pilha: criar, destruir, remover, inserir e buscar;
4. Função para "criar" um nó para o cliente (evitando mallocs dentro do main);
5. Função de enviar cliente para atendimento (guichê);
6. Função para imprimir relatório parcial;
7. Função para gerar/imprimir relatório final;

`Lembre-se: Tudo que é alocado, deve ser desalocado!`

☻ **Boa diverSão!!!!** ☺