UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br/

Professor: Rodrigo Fernandes de Mello (mello@icmc.usp.br)
Alunos PAE: Lucas Pagliosa (lucas.pagliosa@usp.br)
Ricardo Fuzeto (ricardofuz@usp.br)
Yule Vaz (yule.vaz@gmail.com)
Monitores: Victor Forbes (victor.forbes@usp.br)

Trabalho 5: Relatório de Fluxo de Caixa para Terminal de Auto-Atendimento

1 Prazos e Especificações

O trabalho descrito a seguir é individual e não será tolerado qualquer tipo de plágio ou cópia em partes ou totalidade do código. Caso seja detectada alguma irregularidade, os envolvidos serão chamados para conversar com o professor responsável pela disciplina.

A entrega deverá ser feita única e exclusivamente por meio do sistema run.codes no endereço eletrônico `https://run.codes`. Sejam responsáveis com o prazo final para entrega: o run.codes disponibiliza o prazo máximo de submissão dos trabalhos, está programado para não aceitar submissões após este prazo e não serão aceitas entregas fora do sistema.

Leia a descrição do trabalho com atenção e várias vezes, anotando os pontos principais e as possíveis formas de resolver o problema. Comece a trabalhar o quanto antes para que você não fique com dúvidas e que consiga entregar o trabalho a tempo.

2 Descrição do Problema

"Caixa automático" é o nome popularmente dado ao terminal de auto-atendimento, localizado tipicamente no *hall* de entrada de bancos e em quiosques de atendimento bancário. A principal função desses terminais é tornar o atendimento aos clientes mais fácil e rápido, oferecendo acesso a diversas opções, permitindo-os gerir a própria conta. Entre estas opções, as mais comuns são saque, depósito, pagamentos e consultas. Para isso, o cliente deve possuir o cartão da respectiva conta, a senha e outras informações de acesso que o banco possa adotar.

A característica mais importante no funcionamento de um terminal de auto-atendimento é a **operação por transações**: quando uma operação bancária é executada por um cliente, todos os efeitos são registrados e executados **somente se** todas as etapas puderem ser executadas com sucesso. Quando uma etapa de uma operação não é executada corretamente, toda a operação é abortada e ninguém deve ser prejudicado, seja cliente ou o próprio banco.

Neste trabalho, você deve implementar um sistema que calcule o fluxo de dinheiro de um conjunto de terminais de auto-atendimento, ao final de um dia de funcionamento. Cada terminal é capaz de atender a um conjunto de bancos , portanto nenhum terminal está associado a um único banco. O relatório final, portanto, deve indicar a variação de dinheiro disponível em cada um dos terminais, distinguindo o fluxo pertencente a cada banco atendido no dia.

O conjunto utilizado será de **4 terminais diferentes**, sendo todos universais: atendem a maioria significativa dos bancos que operam na região. Cada operação bancária é relacionada a um terminal específico, assim como a um banco específico. Cada terminal terá um código de identificação próprio, assim como cada banco será identificado por um código próprio.

A cada operação realizada, o sistema deve registrar: tal operação, o terminal utilizado, o banco responsável pela operação, e variações de saldo (caso haja). Além disso, ao final do dia, o sistema deve permitir que o registro de operações seja **auditado**, exibindo o fluxo de dinheiro (entrada e saída) de cada terminal, para cada banco atendido. Ainda durante a etapa de auditoria o sistema deve fazer consultas imediatas ao registro de operações, exibindo os dados de uma ou mais operações do tipo escolhido, que serão fornecidos como parâmetro de entrada. Para que a etapa de auditoria do sistema não seja desnecessariamente demorada, o registro de operações deve ser implementado utilizando endereçamento e acesso via *hashing*. Ao final desta fase, o sistema também deve exibir o total arrecadado pelos terminais, através das taxas cobradas sobre cada operação realizada.

Além das especificações da auditoria a ser realizada, há uma restrição muito importante: todos os relatórios de auditoria só devem ser gerados **após a execução de todas** as operações bancárias. Por isso, é proibido armazenar e/ou auditar qualquer informação antes que a última operação bancária seja processada e finalizada.

Para este trabalho, considere apenas as seguintes operações bancárias:

- **Saque**: o cliente retira parte do valor disponível na conta, para transportar em dinheiro. Neste caso, o saldo da conta **diminui**;
- **Depósito**: o cliente insere uma quantia de fundos (em dinheiro ou cheque) na conta de destino. Nesta operação, o cliente pode inserir fundos na própria conta ou na de terceiros, **aumentando** o saldo disponível;
- **Consulta**: o cliente pode gerar um relatório das finanças da própria conta, podendo consultar apenas o saldo disponível (saldo) ou um relatório de movimentação mais detalhado (extrato). Neste caso, o saldo da conta **não é alterado**;
- **Transferência**: esta operação é similar ao depósito, no entanto o valor a ser depositado na conta de destino é retirado diretamente da conta do cliente. Ao mesmo tempo em que o saldo do **cliente diminui**, o saldo do **destinatário aumenta**.
- **Taxa operacional**: para cada operação realizada, o cliente terá o valor de 3 reais descontado de sua conta, que não é transferido para outra conta.

3 Arquivos de Entrada e Saída

Os dados do arquivo de entrada definem o terminal utilizado, o banco do cliente e a operação realizada. Caso a operação seja de saque ou depósito, o próximo dado de entrada é o valor associado à operação realizada. Se a transferência for entre contas de bancos diferentes, o código de identificação do banco de destino aparece antes do valor da transação. Para transferências entre contas do mesmo banco, a entrada terá o mesmo formato de um depósito. Quando houver a requisição de consultas para auditoria, o arquivo de entrada terá, ao final, o terminal consultado, o tipo de operação e o número de operações a serem consultadas.

Para descrever o arquivo de entrada, utilizaremos as seguintes variáveis:

- `int` terminal t_i
- `int` banco b_i
- `char` operação o_i
- `float` valor v
- `int` banco de destino b_j (casos de transferência entre bancos distintos)
- `int` número de operações n (casos de auditoria)

Cada operação possui o seguinte formato específico (notar os valores instanciados de o_i a cada operação):

- **Saque**: t_i b_i o_i=S v
- **Depósito**: t_i b_i o_i=D v
- **Consulta**: t_i b_i o_i=C
- **Transferência entre contas do mesmo banco**: t_i b_i o_i=T v
- **Transferência entre contas de bancos diferentes**: t_i b_i o_i=T b_j v
- **Auditoria de terminal**: t_i o_i n

A saída do sistema deve exibir um relatório completo, terminal a terminal, do fluxo de dinheiro alcançado durante o período de operação determinado pelos dados de entrada. O relatório deve apresentar, distintamente para cada terminal: lista de bancos atendidos, com os valores de entrada e saída de dinheiro vivo e via transferência de cada um; e o valor lucrado pela companhia responsável pelos terminais, adquirido através das taxas cobradas. Ao final dos relatórios particulares de cada terminal, o sistema deve exibir um relatório no mesmo formato, mas compilando as informações de todos os terminais. Para as operações de transferência entre contas do mesmo banco, não deve haver alteração nos valores finais do relatório.

Caso hajam requisições de auditoria no arquivo de entrada, o sistema deve exibir: a auditoria requisitada (exibindo os dados de entrada já formatados), e uma lista padronizada com a formatação dos dados das operações solicitadas.

Exemplo: suponha os seguintes dados de entrada:

```
1 1 S 100.00
3 1 S 50.00
1 33 C
1 33 S 120.00
2 341 S 80.00
3 33 T 43.57
2 341 C
2 341 S 25.00
1 1 S 70.00
3 237 C
3 237 T 1 132.97
4 1 S 75.00
1 1 S 30.00
1 33 S 50.00
4 1 C
4 1 S 50.00
3 1 S 100.00
1 341 C
1 341 T 241.47
1 341 S 70.00
2 1 S 50.00
2 33 S 80.00
```

O relatório de saída do programa deve ser o seguinte:

```
===TERMINAL 1===
Banco 1: Moeda +0.00 -200.00 Transferencia +0.00 -0.00
Banco 33: Moeda +0.00 -170.00 Transferencia +0.00 -0.00
Banco 341: Moeda +0.00 -70.00 Transferencia +0.00 -0.00
Lucro obtido: 27.00
```

```
===TERMINAL 2===
Banco 1: Moeda +0.00 -50.00 Transferencia +0.00 -0.00
Banco 33: Moeda +0.00 -80.00 Transferencia +0.00 -0.00
Banco 341: Moeda +0.00 -105.00 Transferencia +0.00 -0.00
Lucro obtido: 15.00
===TERMINAL 3===
Banco 1: Moeda +0.00 -150.00 Transferencia +132.97 -0.00
Banco 237: Moeda +0.00 -0.00 Transferencia +0.00 -132.97
Lucro obtido: 15.00
===TERMINAL 4===
Banco 1: Moeda +0.00 -125.00 Transferencia +0.00 -0.00
Lucro obtido: 9.00
===TOTAL===
Banco 1: Moeda +0.00 -525.00 Transferencia +132.97 -0.00
Banco 33: Moeda +0.00 -250.00 Transferencia +0.00 -0.00
Banco 237: Moeda +0.00 -0.00 Transferencia +0.00 -132.97
Banco 341: Moeda +0.00 -175.00 Transferencia +0.00 -0.00
Lucro obtido: 66.00
```

Ainda no mesmo exemplo, se considerarmos que a entrada possui, ao final, os seguintes pedidos de auditoria:

```
1 S 2
4 T 1
1 T 2
```

A saída deve ser acrescida do seguinte relatório:

```
===AUDITORIA===
===SAQUE TERMINAL 1===
Mostrando primeiros 2 resultados
1- Banco 1 100.00
2- Banco 33 120.00
===TRANSFERENCIA TERMINAL 4===
Mostrando primeiros 1 resultados
Sem resultados
===TRANSFERENCIA TERMINAL 1===
Mostrando primeiros 2 resultados
1- Banco origem 341 Banco destino 341 241.47
```

4 Dicas

- Comece de baixo para cima: em geral, um sistema computacional é construído utilizando "partes" que são necessárias a outras, gerando uma dependência entre diferentes "pedaços" do sistema. Comece pelos pedaços mais simples e que não dependem de nenhuma outra parte, pois possivelmente você precisará destes logo em seguida;

- Pense na sequência das operações do sistema, e planeje o fluxo de execução para que esta ordem seja respeitada. Com o sequenciamento correto das operações, é mais fácil controlar a execução do seu sistema;

- No caso das partes onde será usado o *hashing*, lembre-se das funções de conversão dos índices de acesso e dos tratamentos de colisão. Além disso, neste projeto existem alguns casos em que não é necessário utilizar uma função matemática para calcular os índices de acesso. Mas onde isso acontece, e qual é a modelagem do sistema que permite o uso dessa propriedade?

5 Desafios

- **Funções de *hashing* distintas**: quando tratamos de sistemas bancários, segurança sempre é um dos fatores mais importantes. Para dificultar o acesso aos registros das operações, uma estratégia é tornar a organização dos dados mais "confusa". Para isso, neste desafio você deve utilizar funções de *hashing* distintas para cada terminal. Desta forma, o acesso aos dados de cada terminal é feito de forma diferenciada, o que irá potencialmente mudar a posição relativa dos dados nos vetores de armazenamento.

- **Operações com erro**: as operações de entrada podem conter um identificador especial, indicando que a operação não foi bem sucedida. Nestes casos, a linha seguinte à declaração da operação deve conter o *token* "ERRO", indicando que a operação deve ser tratada como falha. Ao encontrar tal *token*, o sistema deve armazenar a operação num espaço à parte, onde as informações das operações mal sucedidas são armazenadas. O mecanismo de armazenamento destas operações deve ser semelhante ao das operações bem sucedidas. No relatório de auditoria, deve constar, para cada terminal e no resumo geral, a quantidade de operações que falharam, com o volume de cada tipo de operação. No caso de várias operações com erro, a ordem de exibição das quantidades, por tipo de operação, é: saque, depósito, consulta e transferência.

6 Observações importantes

- Programe as impressões na tela EXATAMENTE como exemplificado no decorrer deste documento. Tome cuidado com pulos de linha, tabs, espaços, etc.

- Coloque dentro do zip todos os arquivos de código (`*.h *.c`), o makefile e um arquivo texto com o nome e número USP.