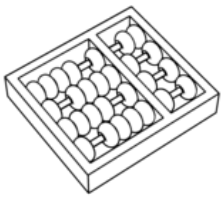

Instituto de Computação
Universidade Estadual de campinas

MC202 - Estruturas de Dados

Laboratório 7

Filas de Prioridade

Data de publicação: Sexta feira, 3 de junho de 2016
Prazo máximo de submissão: Domingo, 12 de junho de 2016 às 23h59m

Professor: Neucimar J. Leite <neucimar@ic.unciamp.br>

Monitores:
- Juan Hernández (PED) <juan.albarracin@students.ic.unicamp.br>
- Leonardo Yvens (PAD) <leoyvens@gmail.com>

Grupo do curso: https://groups.google.com/d/forum/mc202bc_2016s1
Sítio eletrônico da submissão do código: https://susy.ic.unicamp.br:9999/mc202bc

Enunciado

Determine a ordem em que várias solicitações de declaração de renda serão atendidas por um escritório de contabilistas, através de uma simulação do processo que utilize filas de prioridade.

Descrição do problema

Imagine o seguinte cenário fictício: unicamente durante o primeiro dia do ano, um escritório de contabilistas recebe solicitações para elaborar declarações de renda de pessoas físicas. As *n* solicitações coletadas nesse dia serão atendidas ao longo do ano segundo a sua prioridade, a qual depende dos seguintes três fatores:

1. **O mês em que a declaração deve ser apresentada:** Por exemplo, declarações que devam ser apresentadas no mês de março serão processadas, sem exceção, antes de declarações que devam ser apresentadas em meses seguintes.

2. **Pagamento de taxa de prioridade:** O cliente tem a opção de pagar uma taxa de prioridade, a qual faz com que a sua declaração seja elaborada antes que as demais declarações a serem apresentadas no mesmo mês.
3. **Quantia da declaração:** Cada declaração tem associada uma quantia a ser apresentada. Declarações com quantias maiores costumam ser mais complexas e por tanto mais demoradas do que declarações com quantias menores. Como estratégia para reduzir o tempo médio de espera dos clientes, o escritório atende primeiro as declarações menores.

Em resumo, o que o escritório faz é o seguinte: separa as solicitações por mês e, por cada grupo formado, separa as solicitações com a taxa paga das que não têm taxa paga. As solicitações de cada um dos grupos resultantes são ordenadas de forma ascendente por quantia e são processadas nessa ordem.

Cada mês, o escritório pode atender solicitações cuja soma de quantias deem um total de até *k*. Porém, caso a soma das quantias para o presente mês seja maior que *k*, o escritório, de algum jeito, deve atender tudo o que corresponde ao mês antes de o mesmo acabar. Se não for o caso, o escritório pode atender solicitações dos meses seguintes, sem ultrapassar um total de *k*. Para simplificar, suponha que se uma declaração a ser elaborada tiver uma quantia que somada às quantias das declarações já feitas no mês, supera *k*, a elaboração da mesma será adiada para o início do mês seguinte.

Por fim, o escritório permite que no primeiro dia de cada mês os clientes cujas declarações não tenham sido elaboradas, ainda, possam pagar a taxa de prioridade, caso não tenham pago no início do ano, modificando a ordem de atendimento planejada.

Especificação de entrada e saída

- A primeira linha contém os valores *n* e *k* separados por espaço.
- As seguintes *n* linhas contêm 4 valores (1 String e 3 inteiros) separados por espaço, representando a seguinte informação:

  ```
  <nome do cliente> <mês de entrega> <flag de pagamento da taxa> <quantia>
  ```

  em que o mês de entrega é representado por inteiros entre 1 e 12, o cliente é uma cadeia de uma palavra só com caracteres a-z em minúsculo e o flag de pagamento da taxa é 1 quando a taxa foi paga e 0 quando não.
- As linhas seguintes estão separadas em 11 grupos, representando cada mês do ano (a partir de fevereiro). Cada grupo de linhas corresponde ao descrito abaixo:
  - A primeira linha do grupo contêm um inteiro *c* representando o número de clientes que decidiram pagar a taxa no mês correspondente.
  - As *c* seguintes linhas contêm apenas o nome do cliente que pagou a taxa.

A saída esperada consiste de *n* linhas, cada uma contendo o nome do cliente, na ordem em que sua declaração foi elaborada.

Entrada	**Saída**
7 400 miguel 2 1 310 sophia 2 1 250 davi 8 0 200 alice 6 0 120 arthur 2 0 100 julia 5 0 270 pedro 8 1 260 1 arthur 0 0 1 davi 0 0 0 0 0 0 0	sophia arthur miguel julia alice pedro davi
10 500 isabella 7 0 220 gabriel 7 1 470 manuela 2 1 330 bernardo 8 0 170 laura 2 0 400 lucas 8 1 210 luiza 8 1 100 matheus 8 1 410 valentina 7 1 190 rafael 2 1 280 0 0 0 0 0 1 Isabella 0 0 0	rafael manuela laura valentina gabriel isabella luiza lucas matheus bernardo

0 0	

Bônus

Como pode ser visto, o seu programa deve procurar um cliente na fila de prioridade. Dado que a estrutura da fila não considera o nome do cliente, você teria de percorrer a fila em ordem até encontrá-lo. Seria interessante que você construísse um índice baseado em árvores binárias de busca para optimizar este processo. Uma implementação **totalmente correta** deste índice somará 2.0 á nota do presente laboratório, e caso a mesma exceda 10, o resto será somado à nota do laboratório 6, correspondente a árvores binárias de busca.

Estrutura da submissão

O código fonte deve ser submetido no sistema SuSy para ser executado e testado. O sistema receberá três arquivos:

Nome	**Função**
main.c	Programa principal que lê os dados de entrada, faz as chamadas às funções e escreve a saída.
pqueue.h	Contém unicamente a declaração do TAD e de funções para operações em filas de prioridade. Não será fornecida uma estrutura de arquivo, portanto você tem a liberdade de fazer a sua implementação para demonstrar seu domínio sobre o tema. Espera-se, no mínimo, uma função de criação, uma função de destruição e funções para adicionar, remover e mudar prioridade dos elementos da fila.
pqueue.c	Implementação das funções declaradas em **pqueue.h**.
request.h	Cada solicitação deve ser representada com sua própria estrutura contendo o nome do solicitante, mês de apresentação, estado da taxa de pagamento e quantia. Neste arquivo deve estar a declaração da estrutura e as funções para operações em nesta estrutura.
request.c	Implementação das funções declaradas em **request.h**.
bstree.h	Só se quiser optar pelo bônus. Caso contrário, submeta um arquivo vazio. Veja as especificações no enunciado do laboratório 6 para mais detalhes.
bstree.c	Só se quiser optar pelo bônus. Caso contrário, submeta um arquivo vazio.

Observações

- Para evitar confusões, os testes no SuSy serão controlados de forma tal que o nome de cada cliente seja único e as quantias sejam sempre diferentes.

- Se por acaso a entrada ao sistema indica que um cliente cuja declaração já foi elaborada pagou a taxa de prioridade, este evento deve ser ignorado pelo programa.

- É mandatório liberar memória dinamicamente alocada.

- Arquivos de teste serão fornecidos para a validação do programa. A avaliação no sistema será feita com esses e outros testes privados.

- O limite de submissões no SuSy é **15**. Recomenda-se executar de forma local seu programa, usando tanto os testes abertos quanto os exemplos aqui fornecidos. **Submeta seu programa somente após haver testado localmente e de maneira bem sucedida todos os casos.**

- A clareza do código e a sua documentação (por meio de comentários) serão avaliadas. O esforço dos monitores para entender o seu código deve ser mínimo. Com a documentação, você demonstra que compreendeu de maneira efetiva os conteúdos do curso referentes ao tema filas de prioridade.

- Dúvidas podem ser esclarecidas nas aulas de laboratório ou no grupo do curso indicado no cabeçalho deste documento.

Critério de avaliação

$$nota = 5\frac{n_c}{c} + QD + AC$$

$$0 \leq QD \leq 1$$
$$0 \leq AC \leq 4$$

onde,

n	:	Número total de testes no SuSy
n_c	:	Número de testes corretos
QD	:	Qualidade do código

AC : Indicador de quanto o estudante tem demostrado dominar o tema de árvores binárias

A qualidade do código (*QD*) dependerá tanto da legibilidade quanto da documentação. Deve ser fácil para os monitores entender o que foi feito. Já o *AC* é uma nota que depende de quanto o estudante conseguiu convencer o revisor de que sabe implementar corretamente os TADs e operar com a estrutura em questão.

Considerações finais

- Embora haja várias maneiras de resolver os problemas indicados nos laboratórios, o estudante deve optar pela maneira que melhor exercite os conceitos trabalhados em sala de aula. **Não atender a esta advertência invalidará a parte da nota referente aos testes do SuSy**.

- Casos de plágio acarretam **média final zero na disciplina** para todos os envolvidos, sem exceção. O SuSy pode detectar casos de plágio no código, portanto evite compartilhar seu código com outros colegas, mesmo que seja apenas pequenos trechos.