tp5- Trabalho prático de CI1001
Tipos Abstratos de Dados e listas ligadas
TAD's Fila e LEF
Departamento de Informática/UFPR

1 Sobre a entrega do trabalho

São requisitos para atribuição de notas a este trabalho:

- Uso de um arquivo makefile para facilitar a compilação. Os professores rodarão "make" e deverão obter o arquivo executável funcional com a sua solução. Este executável, cujo nome deverá ser tp5, deverá estar no subdiretório tp5;
- Ao compilar, incluir pelo menos `-Wall -Werror -Wextra`. Se não compilar, o trabalho vale zero. Haverá desconto por cada *warning*;
- Arquivo de entrega:
  - Deve estar no formato tar comprimido (.tgz);
  - O tgz deve ser criado considerando-se que existe um diretório com o nome do trabalho. Por exemplo, este trabalho é o tp5;
  - Então seu tgz deve ser criado assim:
    - Estando no diretório tp5, faça:
    - `cd ..`
    - tar zcvf tp5.tgz tp5
  - Desta maneira, quando os professores abrirem o tgz (com o comando tar zxvf tp5.tgz) terão garantidamente o diretório correto da entrega para poderem fazer a correção semi-automática.
  - O que colocar no tgz? Todos os arquivos que são necessários para a compilação, por isso se você usa arquivos além dos especificados, coloque-os também. Mas minimamente deve conter todos os arquivos `.c`, `.h` e o makefile;
  - Os professores testarão seus programas em uma máquina do departamento de informática (por exemplo, cpu1), por isso, antes de entregar seu trabalho faça um teste em máquinas do dinf para garantir que tudo funcione bem.

2 O trabalho

Você deve baixar o tp5.tgz anexo a este enunciado e abrí-lo para poder fazer o trabalho, pois irá precisar de todos os arquivos ali contidos:

fila.h: arquivo (read only) de *header* com todos os protótipos das funções para manipular a fila;

lef.h: arquivo (read only) de *header* com todos os protótipos das funções para manipular a LEF;

tp5.c: arquivo (read only) para testar sua implementação do `fila.c`.

makefile: sugestão de um makefile que você pode usar (ou adaptar, se quiser).

testa.sh: um script shell para testar o seu programa;

saida_esperada.txt: saída esperada do ./tp5).

Os arquivos `fila.h`, `lef.h` e `tp5.c` não podem ser alterados. Na correção, os professores usarão os arquivos originais.

- Use boas práticas de programação, como indentação, bons nomes para variáveis, comentários no código, bibliotecas, defines... Um trabalho que não tenha sido implementado com boas práticas vale zero.
- Quaisquer dúvidas com relação a este enunciado devem ser solucionadas via email para `prog1prof@inf.ufpr.br` pois assim todos os professores receberão os questionamentos. Na dúvida, não tome decisões sobre a especificação, pergunte!
- Dúvidas podem e devem ser resolvidas durante as aulas.

3 O problema

Este trabalho consiste na implementação dos programas `fila.c` e `lef.c` com base nos arquivos `fila.h` e `lef.h` fornecidos.

Sua implementação da fila deve usar uma lista ligada com nodo cabeça que implementa a política FIFO (First In First Out), isto é, uma política que implementa um Tipo Abstrato de Dados `Fila`.

A sigla `LEF` vem de *lista de eventos futuros* e é, na prática, uma das formas de se implementar uma fila de prioridades.

Sua implementação deve usar uma lista ligada com nodo cabeça que implementa a seguinte política: os elementos são inseridos em ordem crescente pelo campo `tempo`. A remoção é sempre no início.

O desafio é aprender a usar alocação dinâmica e manter a lista sempre ligada, sem perder ponteiros. Use o `valgrind`, lembre-se que ele é seu amigo e vai ajudar você a deixar seu programa livre dos erros mais graves.

Para testar sua implementação, use o arquivo `tp5.c` fornecido. Procure entender a motivação de cada teste feito e a sua importância.

4 Programa de teste

Disponibilizamos um script shell que visa testar seu programa. Neste script fazemos uso pipes combinado com o comando "`diff`", o qual faz a comparação da entrada com a saída.

O uso do script é:

- `./testa 1`: teste inicial, não usa o valgrind, serve para você ver se a lógica do seu programa está correta. Se a saída do script for vazia é porque seu programa está correto.
- `./testa 2`: após você ter sucesso no teste 1, use este teste para que o valgrind aponte demais erros, não apenas os vazamentos de memória que seu programa tem como também outros erros que ele encontra. Aqui a saída não será vazia.

Observação importante: pode ser que você não consiga concluir o teste 1 por causa de problemas mais graves, como segmentation fault ou outras coisas ruins. Neste caso você talvez queira rodar o teste 2 até encontrar o problema.

Se o seu programa não apresentar vazamentos de memória, você deverá receber a mensagem:

`All heap blocks were freed -- no leaks are possible`.

Caso contrário, seu programa tem vazamentos e as mensagens de erro deverão indicar a causa.

Outro tipo de mensagens de erro que podem ocorrer são do tipo:

`Invalid read of size 8` ou outras coisas estranhas. Significa que seu programa tem erros mais graves, embora talvez possa ter passado eventualmente pelo teste 1.

Os professores podem ajudar a entender as mensagens do valgrind!

5 O que entregar

Entregue um único arquivo `tp5.tgz` que contenha por sua vez os arquivos necessários para a compilação do executável `./tp5`.
Atenção: Não modifique em nenhuma hipótese os arquivos `fila.h` e `lef.h`. Na correção, os professores usarão os arquivos originalmente fornecidos.

6 Recomendação

Faça o programa aos poucos, uma sugestão é ir comentando trechos do código do programa `tp4.c` e a medida que estes trechos vão funcionado, descomente mais um trecho e assim por diante.

Ao desenvolver, faça uso de `printfs` para ajudar a depurar o código.

Uma dica importante: quando ocorre `segmentation fault`, nem sempre o `printf` funciona. Isso ocorre pois as impressões não ocorrem imediatamente, elas vão para um buffer que pode não ser esvaziado quando ocorre o `segmentation fault`.

Neste caso, faça uso da função `fprintf`, que faz com que a saída seja impressa em um arquivo. A letra "`f`" antes de `printf` vem de *file* e significa imprimir em arquivo. A recomendação é imprimir na saída padrão de erros, a `stderr`. Assim, você pode substituir um

```
printf ("blabla", lista de variaveis);
```

por:

```
fprintf (stderr, "blabla", lista de variaveis);
```

Bom trabalho!