Trabalho prático de CI1001
TP3
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 tp3, deverá estar no subdiretório tp3;
- Ao compilar, incluir pelo menos `-Wall`. 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 tp3;
  - Então seu tgz deve ser criado assim:
    - Estando no diretório tp3, faça:
    - `cd ..`
    - tar zcvf tp3.tgz tp3
  - Desta maneira, quando os professores abrirem o tgz (com o comando tar zxvf tp3.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 Objetivos

Este trabalho tem como objetivo modificar mais uma vez o Tipo Abstrato de Dados (TAD) para números racionais feito no TP1 e TP2 para exercitarmos alocação dinâmica.

O programa principal é mais parecido com o do TP2, mas desta vez é necessário cuidar dos processos de alocação dinâmica, basicamente, alocar e liberar espaços.

Nesta fase do aprendizado, a ferramenta `valgrind` ajuda a detectar vazamentos de memória (leaks), além de outros erros cometidos, como variáveis não inicializadas, etc.

Assim, são objetivos deste trabalho a prática dos seguintes conceitos:

- Alocação dinâmica de structs e de vetores;
- Manipulação de ponteiros;
- Uso da ferramenta `valgrind`.

3 O trabalho

Você deve reescrever a sua implementação do arquivo `racionais.c` conforme o novo arquivo `racionais.h` fornecido. A diferença básica está no fato das funções retornarem ponteiros para racionais (ponteiros para as structs) e não as structs propriamente ditas.

Você deve baixar o tp3.tgz anexo a este enunciado e abrí-lo para poder fazer o trabalho, pois irá precisar de todos os arquivos ali contidos:

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

racionais.c: um esqueleto de arquivo racionais.c;

makefile: sugestão de um makefile que você pode usar.
É sua responsabilidade fazer as adaptações necessárias neste arquivo sugerido.

tp3.c: um esqueleto de arquivo tp3.c.

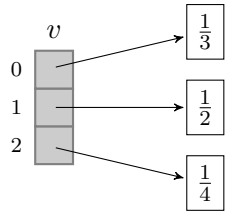
exemplos: um conjunto de entradas para fins de testes.

O arquivo `.h` não pode ser alterado. Na correção, os professores usarão os arquivos `.h` originais.

Você deve implementar um programa que manipule ponteiros para números racionais, que são números da forma $\frac{num}{den}$, onde num e den são números inteiros.

Inicialmente, você vai alocar dinamicamente um vetor de ponteiros para números racionais. Em seguida, você vai gerar vários ponteiros para números racionais gerados aleatoriamente e vai inserir estes ponteiros em ordem no vetor.

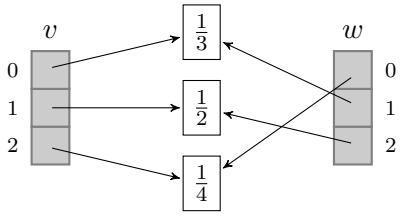
A título de exemplo, considere a figura abaixo. Pode-se ver um vetor v contendo três elementos (índices de 0 a 2). O exemplo mostra que foram lidos, nesta ordem, os números racionais $\frac{1}{3}$, $\frac{1}{2}$ e $\frac{1}{4}$, os quais foram inseridos respectivamente nas posições 0, 1 e 2 do vetor.



Agora seu programa deve manipular este vetor para eliminar os inválidos e em seguida ordená-lo em ordem crescente.

A ideia é que a *struct* pode ser grande e não queremos ficar trocando estas de lugar, só queremos movimentar ponteiros, que custa bem menos.

Considere na figura abaixo que o vetor w é o vetor v ordenado, a ilustração é para fins didáticos apenas. Na figura, o vetor w, quando percorrido do índice 0 até o índice 2, permite ver os racionais ordenados, isto é, as posições 0, 1 e 2 do vetor w apontam respectivamente para os racionais $\frac{1}{4}$, $\frac{1}{3}$ e $\frac{1}{2}$, isto é, estão ordenados.



- 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.

4 Seu programa

No arquivo `racionais.h` foi definida uma interface para o tipo abstrato de dados *racional* que usa a mesma *struct* para os números racionais usada no TP1. Você deve implementar o arquivo `racionais.c` conforme especificado no `racionais.h` fornecido. A sua função `main` deve incluir o *header* `racionais.h` e deve implementar corretamente em *C* o seguinte pseudo-código:

```
leia um n tal que 0 < n < 100
crie um vetor de n posicoes contendo ponteiros para numeros racionais
    - Os racionais deverao ser inicializados com valores lidos do teclado
    - Este vetor tambem deve ser alocado dinamicamente
imprima os racionais apontados pelos elementos do vetor
elimine deste vetor os racionais invalidos
imprima o vetor resultante
ordene este vetor
imprima o vetor ordenado
calcule e imprima a soma de todos os racionais apontados pelo vetor
libere toda a memória alocada
    - todos os racionais
    - o vetor
    - o espaço utilizado para fazer o cálculo da soma
ao final, mude de linha

retorne 0
```

Imprima os elementos do vetor em uma única linha usando um único espaço em branco para separar os elementos. Ao final do vetor mude de linha.

5 Exemplo de entrada e saída

A seguir, 4 exemplos de entrada e saída.

```
Entrada e saida 1:
6
1 1
2 1
1 0
4 1
1 0
1 0
1 2 INVALIDO 4 INVALIDO INVALIDO
1 2 4
1 2 4

Entrada e saida 2:
6
1 1
2 1
1 0
4 0
1 0
1 0
1 2 INVALIDO INVALIDO INVALIDO INVALIDO
1 2
1 2
a soma de todos os elementos eh: 3

Entrada e saida 3:
6
1 0
2 0
1 0
4 0
1 0
1 0
INVALIDO INVALIDO INVALIDO INVALIDO INVALIDO INVALIDO


a soma de todos os elementos eh: 0
```

```
Entrada e saida 4:
6
1 1
2 1
1 1
4 1
1 1
1 1
1 2 1 4 1 1
1 2 1 4 1 1
1 1 1 1 2 4
a soma de todos os elementos eh: 10
```

6 O que entregar

Entregue um único arquivo `tp3.tgz` que contenha por sua vez os seguintes arquivos:

- `racionais.h`: o mesmo arquivo fornecido, não o modifique;
- `racionais.c`: sua implementação do `racionais.h`;
- `tp3.c`: contém a função `main` que usa os racionais;
- `makefile`

Atenção: Não modifique em nenhuma hipótese o arquivo `racionais.h`. Na correção, os professores usarão o arquivo originalmente fornecido.

Bom trabalho!