Campeonato Brasileiro

Programação de Computadores I

ABI/TAI

Prof. Daniel Saad Nogueira Nunes

1 Introdução

O Campeonato Brasileiro, desde 2003, segue uma estrutura de pontos corridos, isto é, a fase do mata-mata foi retirada da competição. Neste formato, atualmente, jogam 20 equipes, que se enfrentam em dois turnos, totalizando 38 jogos por equipe. Ao final de todos os jogos, a classificação é definida de acordo com os seguintes critérios, nesta ordem:

1. Número de pontos;
2. Número de vitórias;
3. Saldo de gols;
4. Gols a favor;

Em outras palavras, em caso de empate no número de pontos, utiliza-se o número de vitórias para desempate. Em caso de empate no número de pontos e vitórias, utiliza-se o critério de saldo de gols, e assim em diante.

Apesar de existirem critérios adicionais, como confronto direto e número de cartões vermelhos e amarelos, eles não serão considerados para este projeto.

O objetivo é calcular a tabela de classificação dos times após todas rodadas do campeonato. As informações das rodadas estarão registradas em um arquivo texto de entrada e a tabela de classificação deverá ser escrita em um arquivo texto de saída.

2 Especificação

Os caminhos dos arquivos de entrada e saída devem ser capturados, nesta ordem, através da linha de comando durante a invocação do programa. O arquivo de entrada contém a descrição das rodadas do Brasileirão, enquanto o arquivo de saída conterá a tabela final de classificação após todas as rodadas.

2.1 Entrada

A primeira linha do arquivo de entrada contém um inteiro n, que corresponde ao número de times que estão disputando o campeonato brasileiro.

As próximas n linhas descrevem, cada, o nome de uma equipe.

As $2 \cdot (n-1)$ rodadas são descritas a seguir. Cada rodada possui $\frac{n}{2}$ jogos no formato: `time_1 gols_1 x gols_2 time_2`, que indica que o `time_1` jogou contra o `time_2` e que o resultado do jogo foi `gols_1` a `gols_2` para o `time_1`. O início e fim de cada rodada estão marcados pelo separador `---`.

Restrições

- n é sempre par;
- $2 \leq n \leq 20$
- Cada nome de time possui apenas letras minúsculas e hífen e estão limitados a 20 caracteres.

2.2 Saída

O arquivo de saída deverá imprimir a tabela de classificação das equipes, da melhor colocada para a pior, após todas as rodadas levando em consideração os critérios de desempate. Cada linha deverá conter as seguintes informações, separadas por um espaço:

1. Posição final (inteiro de 1 a n).
2. Nome do time.
3. Total de pontos.
4. Número de jogos.
5. Número de vitórias.
6. Saldo de gols.
7. Gols a favor.
8. Gols contra.

2.3 Exemplos

Arquivo de Entrada

```
4
juventude
cuiaba
flamengo
internacional
---
juventude 2 x 3 internacional
cuiaba 5 x 5 flamengo
---
juventude 0 x 5 flamengo
internacional 0 x 1 cuiaba
---
juventude 4 x 0 cuiaba
flamengo 5 x 0 internacional
---
internacional 1 x 0 juventude
flamengo 4 x 5 cuiaba
---
flamengo 5 x 4 juventude
cuiaba 2 x 1 internacional
---
cuiaba 3 x 5 juventude
internacional 1 x 5 flamengo
---
```

Arquivo de Saída

```
1 flamengo 13 6 4 14 29 15
2 cuiaba 10 6 3 -3 16 19
3 juventude 6 6 2 -2 15 17
4 internacional 6 6 2 -9 6 15
```

2.4 Modularização

O sistema deverá ser dividido em módulos, cada um para cumprir uma tarefa. Estes módulos podem ser organizados internamente através de várias funções e eles correspondem aos seguintes:

- Módulo de tratamento de erros: trata erros associados aos argumentos transmitidos via linha de comando ou referentes à manipulação de arquivos.
- Módulo de leitura: efetua a leitura dos dados do arquivo de entrada.
- Módulo de saída: efetua a impressão das respostas para cada arquivo de entrada no arquivo de saída..
- Módulo de processamento: realiza o processamento da entrada para composição da saída.
- Módulo de ordenação: ordena os times de acordo com os critérios.
- Módulo principal: contém a função `main` e as chamadas das funções exportadas pelos outros módulos. O ideal é que este módulo possua uma quantidade muito pequena de código, já que ele vai utilizar funções que estão presentes nos outros módulos.

Os módulos devem ser organizados em arquivos separados, com seus respectivos arquivos de cabeçalho e implementação.

2.5 Construção do sistema

Um `Makefile` deverá ser produzido para a compilação dos códigos-fontes no executável e deverá ser distribuído junto ao código.

2.6 Documentação

O código deve ser bem documentado, com presença de comentários explicando os trechos mais complexos do código. Além disso, um arquivo README deve ser providenciado com a devida identificação do autor descrevendo o projeto e instruindo como o código deve ser compilado através da ferramenta `make`.

3 Critérios de correção

Deve ser utilizada a linguagem de programação `C` para a implementação do caça-palavras.

Para validação da correção do algoritmo, testes automatizados serão realizados, então é **crucial** que a saída esteja conforme o especificado.

Serão descontados pontos dos códigos que não possuírem indentação.

3.1 Ambiente de Correção

Para a correção dos projetos, será utilizada uma máquina de 64-bits com sistema operacional `GNU/LINUX` e compilador `GCC >= 10.2.0`, logo é imprescindível que o sistema seja capaz de ser compilado e executado nesta configuração.

4 Considerações

- o `GDB`, `Valgrind` e ferramentas gráficas associadas podem ajudar na depuração do código.
- Este trabalho deve ser feito **individualmente**.
- Não serão avaliados trabalhos que não compilem ou sem a presença de um `Makefile`.
- Como a correção é automatizada, deverá ser impresso apenas o que a especificação pede. Atentem-se para a formatação da saída.
- A incidência de plágio será avaliada automaticamente com nota 0 para os envolvidos. Medidas disciplinares também serão tomadas.
- O trabalho deve ser entregue dentro de uma pasta zipada com a devida identificação do aluno no prazo combinado pelo ambiente virtual de aprendizagem da disciplina.