Atividade Avaliativa I

Estruturas de Dados Básicas I Instituto Metrópole Digital

2020.1

Introdução

Você foi contratado para acelerar o processo de correção de provas de um concurso. Você deve usar o conhecimento adquirido até o momento para realizar as atividades solicitadas da melhor forma possível.

1 Descrição

A empresa **FooBar** o contratou para que você possa auxiliar na correção de provas. As provas, geralmente, são corrigidas manualmente, mas devido ao grande volume de participantes, a tarefa pode demorar muito tempo.

Sua tarefa é fazer um programa que os auxilie na extração de informações dos resultados das provas:

1. Quem foram os X melhores candidatos?
2. Quem foram os Y piores candidatos?
3. Quais questões tiveram mais acertos?
4. Quais questões tiveram menos acertos?
5. Quais questões foram deixadas em branco mais vezes?

Para isto, lhe foi fornecido um arquivo, com valores separados por espaço, que contem o nome do participante e suas respostas da prova, que possuía **10** questões:

```
Fulano   A C D E A C B D X D
Beltrano D E A C B D X D C Y
...
Sicrano  B D X D C O A C D B
```

O **gabarito** da prova também foi fornecido: `A B C D E E D C B A`

Observações:

- O nome da pessoa será **apenas** o primeiro nome;
- Todas as pessoas possuem 10 respostas exatamente;
- Os valores válidos para uma respostas são: A, B, C, D, E. Qualquer outro valor é considerado uma resposta em branco;
- Cada resposta é representada com apenas um caractere;
- As respostas estão na ordem das questões: *resposta da questão 1*, *reposta da questão 2*, ..., *resposta da questão 10*.

A empresa prometeu te pagar um valor extra caso o programa consiga ser utilizado em concursos futuros[1].

2 Detalhes de Implementação

Para fazer um programa que resolva o problema apresentado na seção anterior, você vai precisar utilizar os seguintes conceitos:

- Argumentos de linha de comando
- Leitura de arquivo
- Ordenação
- Alocação Dinâmica

2.1 Interface do Programa

A interface do programa deve ser a seguinte[2]:

- `./prog respostas.txt best X`
  Lista os X melhores candidatos. Em caso de pontuação igual, exibir em **qualquer ordem**.
- `./prog respostas.txt worst Y`
  Lista os Y piores candidatos. Em caso de pontuação igual, exibir em **qualquer ordem**.
- `./prog respostas.txt best-questions X`
  Lista as X questões que tiveram mais acertos. Em caso de pontuação igual, exibir em **ordem crescente**.
- `./prog respostas.txt worst-questions Y`
  Lista as X questões que tiver menos acertos. Em caso de pontuação igual, exibir em **ordem crescente**.

[1] Ver Seção "Avaliação".

[2] Todas as chamadas seguem o padrão `<nome do programa> <nome do arquivo> <acao> <quantidade>`.

- `./prog respostas.txt blank-questions Z`
  Lista as Z questões que foram mais deixadas em branco. Em caso de pontuação igual, exibir em **ordem crescente**.
- Todos os outros comandos devem ser considerados inválidos.

2.2 Dicas de implementação

- Você pode criar um `struct` para armazenar o candidato e suas respostas:

```
struct Candidato
{
    std::string nome;
    char[10] respostas;
};
```

  Outros atributos podem ser necessários;
- Você vai precisar saber a quantidade de candidatos antes de criar um *array* para ordená-lo;
- Tente modularizar seu código de forma que a interface do programa não esteja atrelada ao funcionamento do mesmo.

3 Avaliação

O trabalho possui uma pontuação máxima de 100 pontos e deve ser feito **individualmente**. Esses pontos estão distribuídos da seguinte forma:

- Implementação do `best` — até **20 pontos**
- Implementação do `worst` — até **20 pontos**
- Implementação do `best-questions` — até **20 pontos**
- Implementação do `worst-questions` — até **20 pontos**
- Implementação do `blank-questions` — até **20 pontos**

Também serão contabilizados pontos extras e descontos:

- **Pontos Extras**[3]:
  - Receber um arquivo com o gabarito como parâmetro de entrada — até **10 pontos**
  - Receber a quantidade de questões como parâmetro de entrada — até **5 pontos**

[3] Caso faça alguma das atividades que valem pontos extras, coloque uma arquivo no *zip* informado o que fez.

  - Criar uma biblioteca com o código separado do funcionamento do programa e utilizar esta biblioteca na implementação — **15 pontos**

- **Descontos**:
  - Utilizar o *selection* ou o *insertion sort* — até **-10% dos pontos**
  - Utilizar a biblioteca padrão do C++ (`std::vector`, `std::sort`, ...) ou bibliotecas de terceiros — **-100% dos pontos**
  - Utilizar outra linguagem que não seja o C++ — **-100% dos pontos**
  - Não organizar o projeto com o **CMake** — **-100% dos pontos**

4 Colaboração e Plágio

Ajudar o colega de classe é legal, mas copiar seu trabalho, não. Por isso, cópias de trabalhos **não** serão toleradas, resultando em nota **zero** para **todos** os envolvidos.

Entrega

- Os arquivos devem ser entregues em um arquivo **zip** com o nome do arquivo sendo o número da sua matrícula. Exemplo: `2017105183.zip`
- **NÃO** serão aceitos envios por e-mail. O arquivo deve ser enviado **apenas** pelo *SIGAA* através da opção *Tarefas* até a data divulgada no sistema.