Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Data: 14 de Março de 2019
Professora: Elloá B. Guedes
Disciplina: Fundamentos Teóricos da Computação
Monitor: David Yonekura

Projeto Prático I
Validando o Histórico Acadêmico

1 Conhecendo a plataforma Run.Codes

A disciplina de *Fundamentos Teóricos da Computação* possui projetos práticos em sua avaliação, os quais possuem **caráter individual** e **obrigatório** e que serão executados por intermédio da plataforma *Run.Codes*.

O primeiro passo a ser realizado é o cadastro na plataforma. Acesse o site run.codes e, utilizando o seu **nome completo** e **e-mail institucional**. Após esta etapa, procure a disciplina **Fundamentos Teóricos da Computação** (ESTECP006) e cadastre-se na sua turma utilizando o código **3PD8**. Todos os alunos devem obrigatoriamente se cadastrarem até o dia **20/03/2019**. O não cadastro, o cadastro em turma incorreta ou a não realização dos exercícios nos prazos estabelecidos culminará em nota zero.

Para quem não conhece, o run.codes é uma plataforma automatizada para testes de entrada e saída. Cada aluno submete o seu código escrito na linguagem Python e este código é submetido a um conjunto de testes, previamente escritos e cadastrados pela professora e monitor. A nota do aluno é individual e obtida de maneira automática, correspondendo ao percentual de acertos nos testes. Por exemplo, se há 15 casos de testes cadastrados e o aluno acertou 12, a nota obtida é 8.

A partir do momento em que o exercício inicia até o momento do seu encerramento, o aluno pode submeter o código para a plataforma quantas vezes quiser, sem que isso afete a nota final. Por exemplo, se um aluno submeteu 10 vezes e acertou 12/15 casos de teste, a nota será a mesma de um outro aluno que acertou 12/15 mas que submeteu 300 vezes.

Uma outra regra a ser considerada na plataforma é que a última versão do código é sempre a que será considerada. Imagine que o aluno João Última Hora está enlouquecidamente programando o exercício e já tem 14/15 casos corretos mas, nos segundos finais do prazo limite submete alterações em seu código e cai para 8/15 acertos. Se o sistema encerrar, a versão 8/15 será a considerada para avaliação. Se João Última Hora tivesse aproveitado melhor o tempo e começado a resolver o exercício desde o momento em queficou disponível na plataforma, não teria passado esse sufoco e ficado com uma nota final tão ruim. Todos podemos aprender com o drama de João Última Hora e evitar tais problemas.

Algumas dicas finais para você ter um bom desempenho nos problemas práticos são:

1. Considere que seu programa recebe uma entrada de cada vez;

2. Efetue testes em seu programa antes de submetê-lo ao run.codes. É uma forma simples de conhecer como seu programa se comporta e uma oportunidade de acertar mais testes logo de primeira;

3. Aproveite o tempo;

4. Há boatos de que você não deve deixar seu computador saber quando você está com pressa!

2 Apresentação do Problema

Neste problema, iremos ajudar a validar o histórico acadêmico dos alunos da Escola Superior de Tecnologia com a ajuda de expressões regulares. Os históricos são fornecidos em arquivos-texto e é sua tarefa aplicar os conceitos da disciplina para examinar as partes íntegras deste histórico e, com elas, obter o cálculo correto do coeficiente de rendimento escolar, quando for possível.

Um arquivo correspondente ao histórico escolar de um aluno é composto pelos seguintes elementos:

1. **Cabeçalho**. Composto pelas informações que identificam o aluno. Cada uma das informações a seguir encontra-se disposta em uma linha do arquivo.
   a) **Nome**. Iniciado por `Nome:` seguido por um espaço em branco e depois pelo nome completo do estudante. Este nome pode conter apenas caracteres afabéticos não-acentuados e o espaço em branco. O nome contém, no máximo, 50 caracteres;
   b) **CPF**. Iniciado por `CPF:` seguido por um espaço em branco e depois pelo CPF do aluno. Este dado segue o padrão de CPFs amplamente conhecido. Exemplo: 123.456.789-01. Além de checar se a entrada está no formato de um CPF, deve-se adicionalmente aplicar o algoritmo para checar a validade do CPF fornecido, disponível em http://www.macoratti.net/alg_cpf.htm. Um CPF válido é aquele que passa no padrão e também no teste de validade;
   c) **Matrícula**. Iniciado por `Matricula:` seguido por um espaço em branco e depois pela matrícula do aluno, composta por dez dígitos.
   d) **Separador**. Separa o cabeçalho do conteúdo do histórico. O separador possui uma sequência de 20 hífens contíguos.

2. **Lista de disciplinas**. Segue-se uma lista de disciplinas até que o arquivo termine. Uma disciplina é válida se contempla todos os itens a seguir, na ordem especificada e obedecendo as regras apresentadas. Os itens a seguir devem ser separados por um espaço em branco.
   a) **Semestre letivo**. Um ano composto de quatro dígitos, separado por uma barra e o semestre repetido duas vezes, separado por um hífen. O semestre só pode ser `1` ou `2`. São exemplos de semestres válidos: `2018/1-1`, `1999/2-2`.
   b) **Código da disciplina**. Composto por uma combinação de letras maiúsculas e dígitos, em que as três primeiras letras são EST, as três letras seguintes podem ser ECP, BSI, LIC, BAS, e três dígitos. São exemplos de códigos válidos: `ESTECP009`, `ESTBSI999`,`ESTLIC122`;

c) **Nome da disciplina**. Aparece entre aspas duplas e é composto por letras maiúsculas e minúsculas sem caracteres especiais e com eventuais espaços em branco. Se houver dígitos, estes devem aparecer ao final das letras. São exemplos de nomes válidos: `"Introducao a Engenharia"`, `"Programacao 2"`, `"Fundamentos Teoricos da Computacao"`. O nome da turma deve ter pelo menos um caractere;

d) **Nome da turma**. Composto por letras e dígitos da seguinte forma: iniciado por 3 letras maíusculas, que podem ser ECP, BSI, LIC ou ENG. Se a turma é regular no período, é seguido de dois dígitos. Se a turma é fora de período, seguem-se as letras TFP. Em ambos os casos segue-se um subtraço (*underline*), a letra T e mais dois dígitos. Exemplos de nomes válidos de turma são: `ECP01_T01`, `LIC99_T12`, `BSITFP_T45`;

e) **Créditos**: Um número real positivo e sem sinal, com um dígito inteiro e dois dígitos decimais As casas inteiras e decimais são separadas por vírgulas. São exemplos de créditos válidos: `4,00`, `6,00`, `9,00`;

f) **Nota**: Um número real positivo e sem sinal com 2 dígitos decimais e que utiliza um ponto como separador. Deve estar no intervalo $[0.00, 10.00]$;

g) **Status**. Caracterizado pelas palavras `APROVADO` ou `REPROVADO`, em referência ao resultado do desempenho do aluno na disciplina.

Uma disciplina é considerada válida se todos os seus elementos são válidos e se o status possui acordância com a nota, isto é, se a nota for maior ou igual a 6.00, o status correspondente é de `APROVADO`. Em caso contrário, é reprovado. Assume-se que a primeira disciplina encontra-se na linha 1, a segunda na linha 2 e, assim, sucessivamente. Deve-se tomar nota das disciplinas inválidas. Não havendo linhas inválidas, essa informação não deve ser impressa.

Para todas as disciplinas válidas, se houver, deve ser calculado o coeficiente do rendimento escolar (CRE) do aluno e exibido como saída ao final. Este CRE é dado pelo somatório de cada nota vezes o crédito correspondente, dividido ao final pelo somatório de todos os créditos, como mostrado a seguir:

$$\text{CRE} = \frac{\sum_i \text{nota}_i \cdot \text{credito}_i}{\sum_i \text{credito}_i}, \tag{1}$$

em que o somatório contempla apenas as disciplinas válidas.

As saídas a serem produzidas pelo programa são as seguintes:

1. `CABECALHO VALIDO` ou `CABECALHO INVALIDO`. Se o cabeçalho for inválido, o programa deve ser encerrado.

2. Se o cabeçalho for válido e houver disciplinas inválidas, listar todas as linhas em que há disciplinas inválidas. As linhas são sempre denotadas com dois dígitos inteiros. Além disso, exibir o coeficiente do aluno.

Para resolver o problema em questão, você deve utilizar a linguagem de programação Python 3 e obrigatoriamente fazer uso de expressões regulares. Soluções que não fizerem uso de expressões regulares serão anuladas. Os exemplos a seguir auxiliam a ilustrar entradas e saídas para o problema considerado.

3 Exemplos de Entradas e Saídas

Entrada
`Andre_Markov.txt`
Conteúdo do Andre_Markov.txt

`Nome: Andre Markov`
`CPF: 111.444.777-35`
`Matricula: 1715040792`
`--------------------`
`2017/1-1 ESTBAS002 ‘‘Calculo 1’’ ENG01_T01 6,00 7.00 APROVADO`
`2017/2-2 ESTBAS012 ‘‘Probabilidade e Estatistica’’ ENG02_T05 4,00 10.00 APROVADO`
`2018/1-1 ESTCMP029 ‘‘Matematica Discreta’’ ECP03_T01 4,00 8.00 REPROVADO`
`2018/2-2 ESTBAS049 ‘‘Calculo numerico’’ ECP04_T01 4,00 9.00 APROVADO`

Saída

`CABECALHO VALIDO`
`LINHAS INVALIDAS:`
`LINHA 03`
`CRE: 8.43`

Entrada
`Jorge_Cantor.txt`
Conteúdo do Jorge_Cantor.txt.txt

`Nome: Jorge Cantor`
`CPF: 111-222-333.44`
`Matricula: 1234567890`
`--------------------`
`2017/1-1 ESTBAS001 ‘‘Algebra Linear 1’’ ENG01_T01 4,00 8.00 APROVADO`
`2017/2-2 ESTBAS010 ‘‘Desenho Basico’’ ENG02_T05 4,00 3.00 REPROVADO`
`2018/1-1 ESTCMP029 ‘‘Matematica Discreta’’ ECP03_T01 4,00 10.00 APROVADO`
`2018/2-2 ESTCMP030 ‘‘Teoria da Computacao’’ ECP04_T01 4,00 10.00 APROVADO`
`2019/1-1 ESTCMP037 ‘‘Simulacao e analise de desempenho’’ ECP05_T01 4,00 7.00 APROVADO`

Saída
`CABECALHO INVALIDO`

4 Observações Importantes

- Lembre-se, a entrada de dados é feita via `input` e a saída via `print`;
- O arquivo a ser aberto encontra-se no diretório local;
- Atenha-se exatamente ao padrão de entrada e saída fornecidos nos exemplos. Qualquer mensagem adicional na entrada ou na saída de dados pode culminar em incorretude;
- Cuidado ao copiar caracteres do PDF! Eles podem estar com codificação incorreta. Atente-se ao enunciado;

- A cada execução do programa será fornecida apenas uma entrada, cujo resultado deve ser exibido ao final do processamento;
- Na construção do seu programa você deve usar apenas os conceitos aprendidos em sala de aula. Respostas que utilizem bibliotecas prontas não serão consideradas;
- Em caso de plágio, todos os envolvidos receberão nota zero!
- Na execução do seu programa no *run.codes*, existem casos de testes que vão além dos exemplos mostrados a seguir. Esses casos de teste não serão revelados. Pense em exemplos de entradas e saídas que podem acontecer e melhore o seu código para capturá-las.

5 Prazos Importantes

- **Início**. 14/03/2019 às 8h (horário do servidor)
- **Encerramento**. 22/03/2019 às 23h55min (horário do servidor)

6 Links Úteis

- https://developers.google.com/edu/python/regular-expressions
- https://www.debuggex.com/cheatsheet/regex/python