Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Data: 12 de Março de 2018
Professora: Elloá B. Guedes
Disciplina: Fundamentos Teóricos da Computação

Projeto Prático I
Validando Notas Fiscais

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 **KHU5**. Todos os alunos devem obrigatoriamente se cadastrarem até o dia **18/03/2018**. 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 monitora. 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

Recentemente as prefeituras têm usado dados de notas fiscais de compra de produtos para sorteio de vantagens entre adeptos dos programas. Porém, na tentativa de obter mais cupons de sorteio, alguns usuários mal intencionados podem tentar submeter notas falsas, o que deve ser detectado pelo sistema. O seu objetivo neste projeto é construir um verificador de notas fiscais baseado na utilização de expressões regulares.

Uma nota fiscal é composta de algumas informações, a citar:

1. **CPF do Comprador**. 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;
2. **CPF do Vendedor ou CNPJ da Empresa**. No caso de venda entre pessoas físicas, deve ser apresentado o CPF do vendedor, o qual possui a mesma estrutura e obedece às mesmas regras do CPF do comprador. Se a venda tiver sido realizada por uma empresa, deve-se receber o Cadastro Nacional de Pessoa Jurídica (CNPJ). Este número é composto por 14 dígitos, sendo da forma `XX.XXX.XXX/YYYY-ZZ` e há uma regra de formação que deve ser verificada para assegurar sua integridade, que pode ser vista em `http://www.geradorcnpj.com/algoritmo_do_cnpj.htm`.
3. **Estampa de tempo da transação**. A transação é acompanhada de uma estampa de tempo com a data e hora da sua realização. Esta estampa é da forma `AAAA.MM.DD HH:MM:SS`.
4. **Preços**. Os preços dos itens que compõem a compra serão exibidos com duas casas decimais e separados por vírgula. O conjunto de todos os preços aparecem entre colchetes. As notas fiscais válidas contemplam, no mínimo, um preço;

5. **Código da transação**. O código de transação é produzido pelo vendedor, de modo a assegurar compras realmente realizadas. O código da transação é uma sequência de 9 números, seguida por uma sequência de 5 letras minúsculas ou números (não pode haver repetições neste trecho), 3 números pares (cada dígito é um número par) e, opcionalmente, um trio de dígitos em binário (compostos por 0 ou 1). As sequências que compõem um código de transação são separadas por hífens.

Os elementos que compõem uma nota fiscal são apresentados em uma única linha conforme a ordem especificada, separados por espaços em branco. Uma transação é válida quando todos os seus elementos são caracterizados de maneira adequada, na ordem especificada, como apresentado anteriormente. De maneira resumida, a entrada do seu problema é uma string contendo uma nota fiscal e a saída é a palavra "True" quando a nota fiscal é válida e "False" em caso contrário.

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.

3 Exemplos de Entradas e Saídas

Entrada	**Saída**
159.200.335-41 56.468.354/0001-10 2018.03.12 23:45:12 [3.15,99.65] 123456789-a32bz-460	True
159.200.335-42 58.468.354/0001-10 2018.13.12 23:45:12 [1.234] 123456789-a32baz-450-101	False

4 Observações Importantes

- Lembre-se, a entrada de dados é feita via `input` e a saída via `print`;
- 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;
- 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**. 14/03/2018 às 8h (horário do servidor)
- **Encerramento**. 21/03/2018 às 23h55min (horário do servidor)

6 Links Úteis

- https://developers.google.com/edu/python/regular-expressions
- http://goo.gl/SWwe4R
- https://www.debuggex.com/cheatsheet/regex/python
- http://www.miraclesalad.com/webtools/md5.php