Teste Prático #TeamKlickpages

Instruções:

Abaixo temos 5(cinco) testes e você deve escolher 1(um) deles para desenvolver. Siga as instruções abaixo para qualquer um dos testes:

- Em nenhum dos testes é permitido o uso de gems que não seja explicitamente liberadas, assim como bibliotecas ou códigos externos, frameworks, bibliotecas gráficas, geradores e etc, excetuando bibliotecas para execução de testes unitários como o rspec ou qualquer outro de sua preferência;
- O teste deve estar funcionando e deve ser enviado compactado para podermos baixar, executar os testes unitários e utilizá-lo segundo a proposta da questão;
- Já que queremos que todos que realizem os testes tenham igualdade no processo, pedimos para que não compartilhe este teste ou publique a sua solução nos seus repositórios;
- Os testes unitários também serão avaliados;
- Utilize ruby, nodejs, python, java, php ou javascript para solucionar as questões.
- Deixe instruções para que possamos rodar as aplicações;
- Quanto maior o nível do teste e maior a qualidade do código e dos testes, melhor será a avaliação.
- **Caso você esteja fazendo seu teste para front-end, escolha entre os testes 1, 2, 4 e 5**
- **Se você estiver fazendo o teste para back-end, não faça o teste número 5**

==========

1. **(Difícil)** Um aplicativo visa encontrar a melhor rota para o seu cliente a partir de qualquer localidade, levando em conta quais ruas estão interditadas. Os clientes, com gostos diferenciados, podem escolher pegar caminhos personalizados.
   1.1. O cliente pode informar sua origem e seu destino, mais quantas paradas achar necessário durante o caminho;
   1.2. O cliente poderá definir se deseja o trajeto mais curto ou o mais rápido;
   1.3. A rota deve considerar os desvios dos locais que estiverem interditados;

   Ao inicializar o programa ele deverá receber um arquivo com a configuração do mapa seguindo o seguinte padrão, com um comando por linha:

   R19-(x,y);(x,y):30
   O **R19** identifica a rua, o primeiro par ordenado identifica onde a rua começa, o segundo onde ela termina, o último número indica a velocidade máxima da via;
   I-(x,y)
   Identifica onde se encontra um ponto de interdição, por onde uma rota não poderá passar;

Observações:

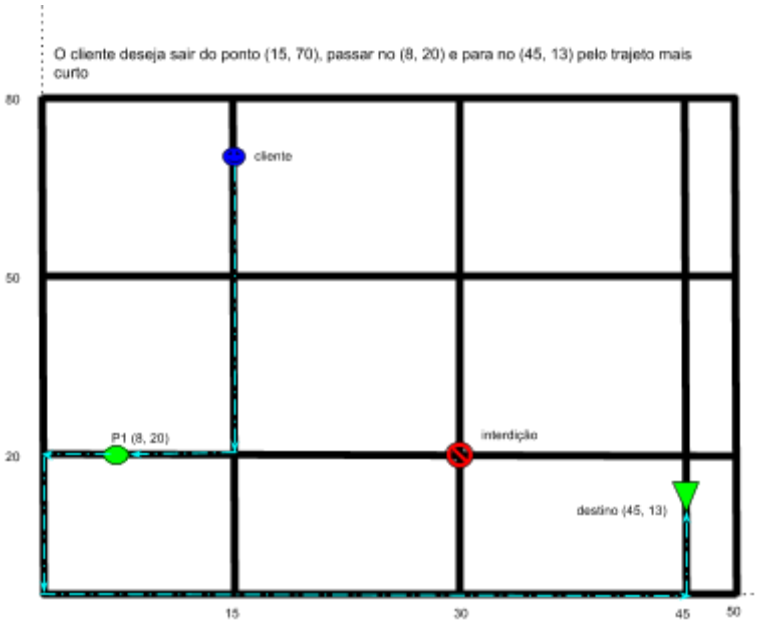
- as ruas só podem estar na horizontal ou na vertical;
- Tome como base o sistema cartesiano e que o eixo das ordenadas no sentido crescente vai para o norte e no sentido decrescente para o sul. O eixo das abscissas no sentido crescente vai para o leste e no decrescente para o oeste;

1. Se pede que o programa receba como parâmetro a origem, o destino, se deseja o caminho mais rápido ou o mais curto e retorne o trajeto no seguinte padrão:
   N-R5, O-R8, L-R19, …

   Onde a primeira letra indica a direção (norte, sul, leste e oeste) e a segunda a rua;
2. Não é obrigatória a construção de uma visualização gráfica (que pode ser feita preferencialmente no console), mas ela será um grande diferencial para nossa avaliação;

Exemplo de entrada para montagem do mapa:
Av. Axis - (0,0);(50, 0):110
Av. Awesome - (0, 20);(50, 20):60
Av. Love - (0, 50);(50, 50):80
Av. Slow - (0, 80);(50, 80):30
St. Oh - (0, 0);(0, 80):90
St. One - (15, 0);(15,80):40
St. Two - (30, 0);(30, 80):50
St. Three - (45, 0);(45, 80):40
St. Four - (50, 0);(50, 80):50

O que formaria algo como este mapa de exemplo, excetuando o trajeto que é um input do programa.:



==========

2. **(Difícil)** Em um teste de robótica, o robô BB8 tem a missão de mover discos de um lado para outro da composição de forma a reorgalizá-los na haste que for ordenado seguindo algumas regras.
   2.1. Há três hastes para onde o BB8 pode mover os discos de uma haste para a outra, sendo identificadas da letra **A** até a letra **C** da esquerda para direita;
   2.2. Há 4 discos de tamanho diferentes, sendo numerados de 1 a 4 do menor para o maior;
   2.3. Um disco maior não pode ficar em cima de um disco menor;
   2.4. Um disco deve ser movido de cada vez;

   Ao inicializar o programa deve receber como parâmetro a entrada onde cada linha informa a composição das hastes, sendo que antes do traço(ou a primeira coluna da linha) contém o identificador da haste, e cada número depois da haste representa os discos, sendo que quanto mais próximo da esquerda mais está perto da base da haste:

   A-4:3:2:1
   B-
   C-

   Esta posição pode ser visualizada assim:

O BB8 deverá ser capaz de:

1. Identificar se a configuração informada no arquivo de configuração é válido ou não, exibindo quais discos estão violando as regras;
2. Sendo a configuração válida, informar quais os passos para reordenar os discos na posição informada pelo operador, por exemplo: A-1, B-4-2, C-3 e respondendo com a sequência de comandos para alcançar o objetivo, como: 1-C, 2-A, 1-B, 2-C, 1-C, 4-A, .. e assim por diante;
3. Não é obrigatória a construção de uma visualização gráfica (que pode ser no terminal) para o teste, contudo ela pode ser útil e um diferencial para os avaliadores;

==========

3. **(Médio)** Construa uma aplicação CRUD (CREATE, READ, UPDATE, DELETE), orientado a objetos. Este CRUD deverá gerenciar Páginas Web que são composta pela model página, configuração da página e tag da página. Uma página poderá ter uma configuração e N tags.
    - A página terá os atributos nome e slug;
    - A configuração da página deverá conter o título, descrição, palavras chaves em uma tabela diferente relacionada a tabela pages;
    - A página poderá ter várias tags e a tag com atributo nome;
    - Deve ser possível pela interface adicionar, deletar, visualizar e editar uma página, onde na visualização devem ser listada as tags desta página e sua configuração;
    - À uma página pode ser adicionada N tags;
    - Em uma próxima feature usaremos a tag para marcar outra entidade;
    - Deverá ser possível adicionar, visualizar, editar e deletar uma tag;

    Para este teste você pode utilizar o database driver apenas para realizar a conexão com o banco, nenhum ORM deve ser utilizado.
    Para subir a aplicação web você poderá utilizar rackup (http://rack.github.io/)

    - Não utilize nenhuma gem que abstraia a arquitetura do teste.
    - Use um banco de dados SQL

==========

4. **(Fácil)** Caixa Eletrônico
   Implementar um caixa eletrônico com a funcionalidade de saque e "carregamento" de notas.

   O caixa eletrônico deve ter as seguintes funcionalidades:
   - Saque
     - Para o saque o usuário deve informar o valor e selecionar quais notas ele deseja que o caixa eletrônico entregue o valor a partir de combinações informadas pelo caixa eletrônico.
     - Exemplo: **R$ 1000,00** em **5** notas de **R$ 100,00** e **10** notas de **R$ 50,00**
   - “Recarga” de notas
     - Para recarregar o caixa eletrônico deve ser validado o valor da nota e se foi informado uma quantia maior que 0.
   - Exibir informações:
     - Notas estão disponíveis;
     - Limite para saque (caso não seja possível efetuar o saque com o limite padrão, calcular qual o limite possível com as notas disponíveis).

   Regras para saque:

   - Limite: R$ 2000,00;
   - Iniciar com: R$ 5000,00
     - 20 x R$ 100,00
     - 100 x R$ 50,00
     - 200 x R$ 10,00
     - 50 x R$ 20,00
   - Após o usuário informar a quantia para saque exibir TODAS as possibilidades para saque;
   - Caso não seja possível realizar o saque do valor solicitado informar os possíveis erros:
     - Limite menor do que o solicitado;
     - Impossibilidade de sacar o valor por conta da quantia de notas e valores;

   Regras para "recarga":

   - Receber uma lista com as notas para serem adicionadas;
   - Cada item da lista deve conter o valor da nota e a quantidade;
   - Validar cada item e caso algum item seja invalido rejeitar a recarga;
   - Aceitar apenas as notas disponíveis informadas nas regras para saque;

   Regras para informações:

- As informações devem exibidas apos ações de saque e recarga serem executadas;
- O usuário pode visualizar as informações a qualquer momento;

==========

5. **(Fácil)** Crie uma calculadora com as funcionalidades de adicionar, subtrair, multiplicar, dividir, porcentagem, calcular raiz quadrada, exponenciação e limpar “visor”.
    - A calculadora deve ser acessível por meio de um navegador;
    - Não é permitido o uso de bibliotecas ou frameworks (ex: jQuery, Angular, React);
    - É permitido a utilização de bibliotecas ou frameworks para testes (ex: mocha, jasmine);