Trabalho Prático - Particionamento do Espaço de Entrada

Instruções:

- Este trabalho deve ser feito em grupos de três pessoas;
- As soluções devem ser implementadas em Java e os casos de teste devem ser implementados utilizando o framework JUnit;
- Os grupos devem submeter através do SIGAA:
    - *Documento com os projetos de teste*: para listar as características, os blocos, os requisitos de teste e os casos de teste (incluindo os resultados esperados). Utilize os modelos apresentados em sala de aula (e o exemplo da Maratona no sigaa) durante os exercícios;
    - *Documento com o relatório da execução dos testes*: este documento deve conter, para cada critério de teste, a quantidade total de casos de teste criados, a quantidade de casos de teste que passaram, a quantidade de casos de teste que falharam e a identificação dos casos de teste que falharam (liste os dados de entrada, os resultados esperados e os obtidos);
    - *Implementação da solução e dos casos de teste*: o projeto Java com o código das soluções e os casos de teste implementados.
- O grupo deve submeter o trabalho até o dia 01/11/2013, 6a feira.

Problemas:

Problema 1

O prefeito da sua cidade lhe contratou para implementar um sistema de passes de ônibus eletrônico. Esse sistema deve suportar três tipos de passageiros: 1) passageiros do tipo *inteira*, que pagam o valor completo da passagem; 2) passageiros do tipo *estudante*, que pagam metade do valor da passagem; e 3) passageiros do tipo *livre* que não pagam pela passagem (um cartão do tipo *livre* deve possuir o saldo sempre igual a zero). Para facilitar o cálculo do saldo, o seu valor é baseado em unidades de crédito. Assim sendo, uma passagem inteira custa 2 créditos, uma passagem de estudante custa 1 crédito e uma passagem livre custa 0 créditos. O cartão não deve permitir que o saldo de créditos seja negativo.

Implemente uma classe que represente o cartão eletrônico. Esse cartão deve possuir um tipo (que pode ser *inteira*, *estudante* ou *livre*), um saldo (tal que $0 \leq saldo \leq 100$) e métodos para adicionar, debitar e consultar créditos.

Problema 2

A professora de uma creche está planejando fazer um campeonato de aviões de papel com as crianças da turma. Para isso ela precisa de um programa que verifica se uma certa quantidade de folhas de papel será suficiente para realizar o campeonato.

O programa deve possuir um único método que recebe como entrada três parâmetros: 1) o número de competidores (tal que 1 ≤ *nCompetidores* ≤ 1000), a quantidade de folhas de papel (tal que 1 ≤ *nFolhas* ≤ 1000) e a quantidade de folhas que cada competidor vai receber (tal que 1 ≤ *nFolhasPorCompetidor* ≤ 1000). O método deve retornar como saída o valor *verdade* caso seja possível realizar o campeonato com os parâmetros dados, ou o valor *falsidade* caso contrário.

Problema 3

Um grande projeto mundial está em curso para mapear todo o material genético do ser humano: o Projeto Genoma Humano. As moléculas de DNA (moléculas que contêm material genético) podem ser representadas por cadeias de caracteres que usam um alfabeto de apenas 4 letras: 'A', 'C', 'T' e 'G'. Um exemplo de uma tal cadeia é:

TCATATGCAAATAGCTGCATACCGA

Escreva um programa que procura ocorrências de uma pequena cadeia de DNA (que vamos chamar de *p*) dentro de uma outra cadeia de DNA (que vamos chamar de *t*). Você deverá procurar dois tipos de ocorrência: a *direta* e a *complementar invertida*.

Uma ocorrência *direta* é quando a cadeia *p* aparece como subcadeia dentro de *t*. Por exemplo, se:

p = CATA
t = TCATATGCAAATAGCTGCATACCGA,

então *p* ocorre na forma direta na posição 2 e na posição 18 de *t*.

Uma ocorrência *complementar invertida* depende da seguinte correspondência entre as letras do DNA: ('A' e 'T') e ('G' e 'C'). "Complementar o DNA" significa trocar as letras de uma cadeia de DNA seguindo essa correspondência. Se complementarmos a cadeia CATA, vamos obter GTAT.

Mas além de complementar, é preciso também inverter, ou seja, de GTAT obter TATG. E é esta cadeia que deverá ser procurada, no caso da ocorrência complementar invertida. Assim, se *p* e *t* são as mesmas cadeias do exemplo anterior, então *p* ocorre na forma complementar invertida na posição 4 de *t*.
Escreva uma classe que possui dois métodos. Ambos os métodos recebem *p* e *t* (4 ≤

comprimento de *p* ≤ comprimento de *t* e 4 ≤ comprimento de *t* ≤ 20) como parâmetros, um método deve procurar por ocorrências diretas e o outro método deve procurar por ocorrências complementares invertidas de *p* em *t*. Cada método deve retornar uma lista de inteiros em ordem crescente, contendo a posição inicial de cada ocorrência de *p* (ou complementar invertida de p) encontradas na cadeia *t*.

Casos de teste:

Para cada um dos problemas anteriores, projete casos de teste segundo os critérios de particionamento do espaço de entrada aprendidos em sala de aula.

1) Liste um conjunto de características (pelo menos 3) para o espaço de entrada de cada método do programa;

2) Faça um particionamento criando blocos para cada uma das características definidas (pelo menos dois blocos por característica);

3) Após definir os blocos do particionamento, liste os requisitos de testes para estes blocos utilizando os seguintes critérios:

a) *All-Combinations*
b) *Each-choice*
c) *Pairwise*

4) Para cada um dos critérios, liste o conjunto de casos de teste necessários para cobrir seus requisitos.

5) Implemente os casos de teste listados no item anterior. Implemente os testes para cada critério em classes separadas. Após implementar os testes, execute-os e faça um relatório indicando quantos casos de teste passaram, quantos casos de teste falharam e indique quais casos de teste falharam (indique a entrada utilizada, o resultado esperado, e o resultado obtido)