V BXComp

5º Campeonato de Programação para Calouros do Curso de Sistemas de Informação – 2015

Introdução ao BXComp para calouros

Olá Competidor!

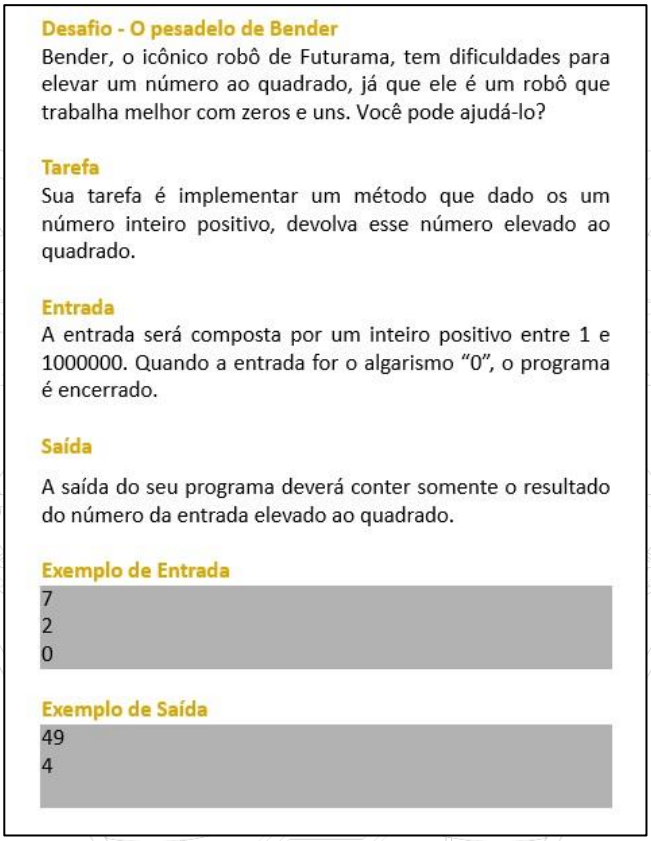
Você está prestes a participar do BXComp, o campeonato de programação para calouros de Sistemas de Informação organizado pelo PET-SI. O BXComp tem como objetivo, além de promover a integração dos alunos, criar um ambiente divertido e desafiador, composto por atividades de resolução de problemas usando programação, para estimular e preparar os alunos do curso para atividades relacionadas como olimpíadas, competições e maratonas de programação.

Cada etapa será composta por vários desafios. Em cada desafio é apresentada uma situação-problema e uma tarefa: resolver esse problema utilizando os seus conhecimentos em programação. Todos os desafios possuem uma pontuação e ao final do campeonato (ou seja, ao final da última etapa), a equipe vencedora será aquela que obtiver o maior número de pontos. Para mais detalhes, consulte o regulamento do BXComp 2015 (http://each.uspnet.usp.br/petsi/bxcomp2015/?page_id=12).

Com o intuito de introduzir a dinâmica adotada no campeonato, elaboramos este pequeno manual no qual explicaremos como os desafios são estruturados e corrigidos. Além disso, também daremos algumas dicas que podem ser muito úteis ao longo do BXComp.

1. *ESTRUTURA DO DESAFIO*

Veja um exemplo de um desafio do campeonato:



Desafio - O pesadelo de Bender

Bender, o icônico robô de Futurama, tem dificuldades para elevar um número ao quadrado, já que ele é um robô que trabalha melhor com zeros e uns. Você pode ajudá-lo?

Tarefa

Sua tarefa é implementar um método que dado os um número inteiro positivo, devolva esse número elevado ao quadrado.

Entrada

A entrada será composta por um inteiro positivo entre 1 e 1000000. Quando a entrada for o algarismo "0", o programa é encerrado.

Saída

A saída do seu programa deverá conter somente o resultado do número da entrada elevado ao quadrado.

Exemplo de Entrada

```
7
2
0
```

Exemplo de Saída

```
49
4
```

Todos os desafios utilizam essa mesma estrutura padrão apresentada no desafio "O pesadelo de Bender":

Introdução: contém um texto de situação-problema explicando a motivação para criar uma resolução ao desafio;

Tarefa: nesta parte é explicada qual a tarefa que a sua equipe deve implementar para resolver o problema do desafio;

Entrada e Saída: nestes itens são explicados qual será o formato da entrada que o seu programa precisa aceitar e como deve ser a saída após a resolução do problema;

Exemplos de Entrada e Saída: em todos os desafios, há exemplos de algumas entradas válidas e o respectivo resultado (saída) esperado.

2. *COMO O DESAFIO É TESTADO*

Para afirmar que uma resolução de um desafio está correta, uma entrada válida deve resultar em uma saída correspondente também válida. Aqui se utiliza a mesma lógica de correções de EPs, por exemplo. A grande diferença é que se há uma saída inválida, ou seja, a resolução do desafio não "funciona" para alguma determinada entrada, o desafio é considerado incorreto.

No BXComp, assim como em diversos campeonatos de programação, as resoluções propostas pelos competidores de cada desafio são testados por um corretor automático, o "Autojudge". No caso do nosso campeonato, ele avalia as resoluções propostas por meio do sistema BOCA. Para mais informações sobre este sistema, não deixe de ler o “Manual sobre o BOCA”, que elaboramos para explicar como se dá a submissão e correção dos desafios propostos.

O “Autojudge” é pré-configurado com uma série de casos de teste, compostos por entradas e respectivas saídas esperadas. Se a resolução do desafio produz as saídas esperadas para todos os testes de entradas, a resolução é considerada correta.

Entenda que são testados apenas entradas e saídas. A forma como a resolução é implementada é de responsabilidade de cada equipe. Inclusive, é comum os desafios possuírem diversas implementações diferentes que resultam em uma mesma solução.

3. *ESTRUTURA DA RESOLUÇÃO*

Uma resolução válida para o desafio "O pesadelo de Bender" é essa:

```java
import java.util.Scanner;

public class DesafioBender {

    public static void main( String[] args ) {

        Scanner s = new Scanner( System.in );
        int entrada = s.nextInt();
        while ( entrada != 0 ) {
            System.out.println( quadrado(entrada) );
            entrada = s.nextInt();
        }
        System.out.println();
    }

    static int quadrado( int i ) {
        return i*i;
    }
}
```

Note que para ler alguma entrada, foi utilizada a classe Scanner do Java. Essa classe fica a sua disposição após realizar a importação no início do código. É de EXTREMA importância que haja uma familiaridade com o básico dessa classe, principalmente com os métodos "next" (hasNext(), nextInt(), nextLine(), etc). É comum que nas primeiras etapas esse processo de leitura das entradas já esteja pronto no seu código, porém, atente-se que a partir de um certo ponto do campeonato isso não será mais disponibilizado pela organização e deverá ser implementada pelos próprios competidores.

Estar familiarizado com os métodos de manipulação de Strings, como o “*charAt*”, “*split*”, entre outros, também será de grande valia no decorrer do campeonato.

Repare que nesta resolução do desafio “O pesadelo de Bender”, o programa sempre funcionará até o usuário digitar o algarismo "0", conforme previsto no enunciado. Então, para cada entrada válida, o método "quadrado(int i)" é chamado, retorna a resposta esperada, que é exibida por meio de um *print,* e o programa aguarda uma nova entrada.

4. *RESTRIÇÕES*

Em alguns enunciados de desafios haverá mais um tópico chamado "Restrições". Ele geralmente indica alguns casos de testes que não serão avaliados, ou seja, no momento da elaboração da resolução, sua equipe não deve se preocupar em tratar esses casos especiais.

Em todo o caso, mesmo quando não há esse item, em nenhum teste realizado haverá uma entrada inválida que não foi especificado no enunciado. Por exemplo, se no item "Entrada" do desafio está especificado que a entrada será um inteiro, a organização não elaborará um teste que contenha na entrada uma String ou um número do tipo ponto flutuante. Desse modo, não há necessidade da resolução da equipe tratar todos os casos de entrada: limitem-se a tratar as especificações que o enunciado do desafio está pedindo.

5. *POSSÍVEIS ERROS NA ELABORAÇÃO DA RESOLUÇÃO*

Pode ocorrer de a lógica de resolução elaborada pela equipe esteja correta, mas esta resolução não é aceita. Atente-se em casos muito simples que podem invalidar a sua resolução:

- **Inclusão de *packages*:** É comum em IDEs, ao se criar um novo projeto, automaticamente se criar um *package* com o mesmo nome do projeto e organizar todos os códigos feitos na sessão nesse *package*. Porém, os testes não serão realizados na sua máquina. As equipes submetem a resolução (o arquivo .java) no sistema BOCA e os testes serão realizados "fora" de qualquer pacote, portanto atente-se em excluir essa linha ao submeter o desafio;
- **Saídas inválidas:** Em alguns desafios, serão solicitados que a saída seja alguma frase, e não apenas um número. A linguagem Java é *case sensitive*, ou seja, letras maiúsculas consideradas diferentes de minúsculas. Portanto, atente-se em exibir as saídas solicitadas exatamente como estão no enunciado do desafio, para não ocorrer o caso da resolução estar correta, mas a resposta estar apresentada de um modo errado com relação ao que foi especificado e que será testado. Resoluções de desafios em que a saída deveria ser a palavra "GANHOU" e a equipe desenvolveu um programa certo, mas cuja saída exibe "Ganhou", apesar das saídas serem semanticamente iguais, são considerados inválidos. Cuidados também devem ser tomados para não se utilizar o caractere "espaço" ou linhas em branco onde não deveria.
- **Diferença entre nome do arquivo e nome da classe:** Atente-se que se o nome do arquivo a ser submetido, por exemplo, for "Teste.java" e o nome da classe no código for "teste" (com a primeira letra minúscula), a resolução será considerada errada. Aqui também ocorre um erro pelo mesmo motivo do item anterior: a linguagem Java diferencia minúsculas de maiúsculas.

Dúvidas? Entre em contato com o PET-SI, grupo organizador do BXComp através do e-mail pet-si-each@usp.br ou pelo formulário de contato disponibilizado no site do BXComp 2015 (http://each.uspnet.usp.br/petsi/bxcomp2015/?page_id=452).