

## Trabalho do Grau A (Peso 30%)

### Cenário: Maratona de Programação

---

A *maratona de programação* é um evento já clássico onde grupos de alunos de diversas instituições participam para demonstrar e competir utilizando suas habilidades e conhecimentos de algoritmos e programação. Tipicamente, as equipes recebem uma lista de problemas e devem construir programas que permitam calcular a resposta para aquele tipo de problema para diferentes casos de teste (parâmetros de entrada).

De forma a permitir a correção automática dos programas produzidos pelas equipes, a organização da maratona estabelece algumas diretivas sobre como os programas deverão ser construídos. Terminado o desenvolvimento do programa, a equipe envia o código fonte para o sistema de correção, o qual compila e executa o programa para diferentes casos de teste.

Considere, por exemplo, que organização da maratona de programação definiu as seguintes diretrizes de desenvolvimento para as soluções (programas) desenvolvidos pelas equipes:

- Os programas deverão ser desenvolvidos em linguagem C, utilizando apenas chamadas a API padrão da linguagem (sem bibliotecas externas adicionais) e compilar sem erros de sintaxe, utilizando as opções padrão do compilador **gcc**.
- Deverá ser utilizado um único arquivo `.c` para a solução de cada problema. O nome do arquivo deverá seguir o padrão `solucao_n.c`, onde 'n' é o número correspondente ao problema que está sendo atendido por aquela solução.
- O programa desenvolvido receberá dados de entrada na entrada padrão de seu processo e e deverá enviar a resposta correspondente para a saída padrão do processo em formato texto, repetindo o formato de entrada e saída definido em cada questão.
- Para verificação do programa, o programa será executado múltiplas vezes com diferentes dados de entrada, sendo comparada a saída produzida com a saída esperada, as quais devem ser idênticas para que o programa seja considerado correto.
- Qualquer caractere adicional ou mensagem de erro ou depuração incluída na saída do programa invalidará a resposta, mesmo que o valor computado esteja correto.

### auto\_test

---

Você foi designado para desenvolver uma ferramenta, denominada **auto\_test**, que auxilie a correção automática dos programas enviados pelas equipes que estão participando da maratona de programação.

#### Entrada de Dados

Seu programa receberá os seguintes parâmetros de execução:

- Através da variável de ambiente `TESTCASES_DIR`, o caminho para um diretório onde estão os casos de teste. Caso não esteja definida, o programa deverá finalizar a execução exibindo mensagem de erro apropriada e finalizando a execução com status de falha; e
- Como parâmetro de linha de comando opcional, o caminho para um diretório onde estão os programas desenvolvidos pela equipe que está sendo avaliada. Caso seja omitido, o `auto_test` deverá assumir o diretório corrente de trabalho como valor para esse parâmetro.

O **diretório de casos de teste** está organizado de forma que existe um subdiretório correspondendo a cada um dos problemas propostos na maratona seguindo um padrão de nome **problema\_nn**, onde *nn* corresponde ao número do problema. Dentro de cada um desses subdiretórios existem pares de arquivos (**kk\_INPUT** e **kk\_OUTPUT**), correspondendo as entradas e saídas de um caso de teste daquele problema. Essa organização está ilustrada na figura 1.

```

casos_de_teste
├── problema_01
│   ├── 00_INPUT
│   ├── 00_OUTPUT
│   ├── 01_INPUT
│   ├── 01_OUTPUT
│   ├── 02_INPUT
│   └── 02_OUTPUT
├── problema_02
│   ├── 00_INPUT
│   ├── 00_OUTPUT
│   └── ...
├── ...
├── problema_03
│   ├── 00_INPUT
│   ├── 00_OUTPUT
│   └── ...
└── ...

```

**Figura 1: Exemplo de Diretório de Casos de Teste**

Já o **diretório de programas entregues** pela equipe contém um arquivo .c para cada uma das soluções submetidas, com o padrão de nome **problema\_nn.c** . É importante ressaltar que uma determinada equipe pode não entregar todas as soluções, como ilustrado na figura 2.

```

equipe1
├── problema_01 .c
├── problema_02 .c
├── problema_05.c
├── problema_07 .c
└── ...

```

**Figura 2: Exemplo de Diretório com Soluções Entregues por uma Equipe**

### Processamento a ser Realizado

Considerando que valores apropriados tenham sido fornecidos como parâmetros, o `auto_test` deverá então compilar, usando o **gcc**, cada um dos programas a serem avaliados e, a seguir, executá-los com os correspondentes casos de teste, observando a informação produzida como status de terminação e na saída padrão dos processos executados.

A partir da informação coletada dos programas executados, o `auto_test` deverá gerar dois tipos de saída:

- **Na saída padrão** a lista dos problemas (1 por linha) para os quais ou a equipe não enviou uma solução, a solução enviada não compilou ou, a execução da solução falhou para algum dos casos de teste;
- **No status de terminação de sua execução**, o `auto_test` deverá retornar 0 (zero) caso a equipe solucione todos os problemas propostos com sucesso, ou um valor maior que zero representando o número de problemas propostos que não foram solucionados.

### Orientações sobre desenvolvimento e entrega da atividade

---

- A atividade poderá ser desenvolvida em grupos de até 3 pessoas (definição em aula em 26/Mar).
- O programa deverá ser desenvolvido em linguagem shell script.
- O arquivo final contendo o programa solicitado, devidamente formatado e documentado, com comentários complementares ao código, assim como os sub-produtos solicitados deverão ser entregues através do moodle dentro do prazo especificado.

- Para ser útil, o comentário incluído deve esclarecer o propósito do comando, no contexto da lógica (algoritmo do script) e não ser uma simples tradução literal das palavras
- **Produtos e prazos de entregas:** vide informações publicadas no moodle.

**AVISO IMPORTANTE:**

É **permitido** e **benéfico** discutir e **trocar ideias** com os colegas ou mesmo utilizar os fóruns para resolver dúvidas. Porém, **em nenhuma circunstância cópias serão admitidas**, mesmo que parciais, sendo penalizadas com a nota zero a todos os envolvidos. Se emprestar diretamente a sua solução a um colega, tenha isso em mente.

***Bom Trabalho!***