

Aula 1:

Problemas Ad hoc

Disciplina: Maratona de Programação 1

Profs. Edmilson Marmo e Luiz Olmes

edmarmo@unifei.edu.br, olmes@unifei.edu.br



Nas aulas anteriores...

▶ **O QUE JÁ ESTUDAMOS?**

- ▶ Apresentação da Disciplina

▶ **OBJETIVOS:**

- ▶ Introdução à Maratona
- ▶ Problemas ad hoc
- ▶ Shell:
 - ▶ Comandos básicos
 - ▶ Compilação
 - ▶ Correção
- ▶ Sistema BOCA
- ▶ Warmup

O que é a Maratona de Programação?

- ▶ A Maratona de Programação é um evento promovido pela Sociedade Brasileira de Computação desde 1996.
 - ▶ Site: maratona.sbc.org.br
- ▶ Surgiu a partir das competições regionais classificatórias para as etapas mundiais do concurso de programação International Collegiate Programming Contest (ICPC).
 - ▶ Site: icpc.global
- ▶ É parte da etapa regional sul-americana da competição.
- ▶ Os times, compostos por exatamente 3 integrantes, representam as instituições de ensino superior. Cada time tem um único computador.



Objetivo

Resolver o **maior número** de problemas computacionais...

...no **menor tempo** possível.

Objetivo

Resolver o **maior número** de problemas computacionais...

...no **menor tempo** possível.

como?



Como?

- ▶ Estudando os **tipos básicos** de problemas:
 - ▶ Ad-hoc.
 - ▶ Algoritmos e estruturas de dados.
 - ▶ Teoria dos grafos.
 - ▶ Matemática.
 - ▶ Geometria computacional.
 - ▶ Combinatória.
 - ▶ Strings.
 - ▶ Força bruta, Programação dinâmica.
- ▶ Explorando **aplicações** de algoritmos e estruturas de dados conhecidos.
- ▶ Aprendendo **novas técnicas** e algoritmos.
- ▶ Aprendendo com os nossos próprios **erros**.
- ▶ Aprendendo com **competidores** mais **experientes**.

Como?

- ▶ **Praticando** a solução de problemas:
 - ▶ Beecrowd: www.beecrowd.com.br
 - ▶ Codeforces: codeforces.com
 - ▶ ICPC Live Archive: icpcarchive.ecs.baylor.edu
 - ▶ UVA Online Judge: onlinejudge.org
 - ▶ E vários outros (SPOJ, POJ, etc.)...

Como?

- ▶ **Praticando** a solução de problemas:
 - ▶ Beecrowd: www.beecrowd.com.br
 - ▶ Codeforces: codeforces.com
 - ▶ ICPC Live Archive: icpcarchive.ecs.baylor.edu
 - ▶ UVA Online Judge: onlinejudge.org
 - ▶ E vários outros (SPOJ, POJ, etc.)...

- ▶ Praticando.
- ▶ Praticando.
- ▶ Praticando.
- ▶ Praticando **mais um pouco!**



Eu treinei 4 anos para
correr apenas 9
segundos.

Tem gente que não vê
resultados em dois
meses e já desiste.

Problemas ad hoc

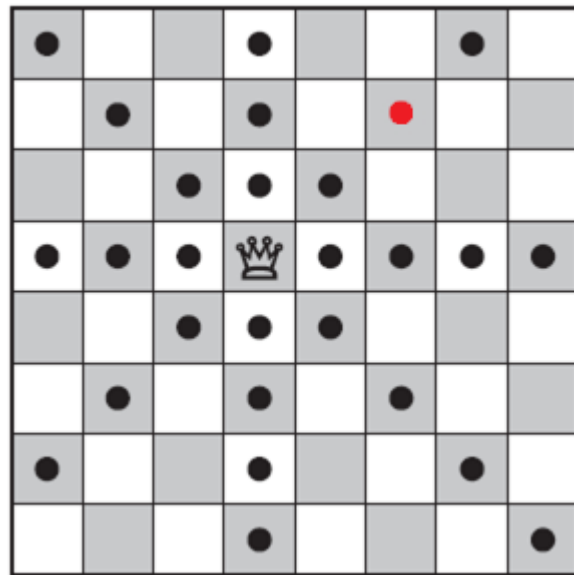
- ▶ **Ad hoc**: do latim, para este fim.
- ▶ Problemas ad hoc são aqueles que não podem ser classificados em alguma categoria de problemas (grafos, geometria computacional, etc.).
- ▶ São frequentes em competições de programação.
 - ▶ Aproximadamente 20% a 30% da prova são problemas ad hoc.
- ▶ É necessário fazer apenas o que o problema pede, seja de maneira direta ou através de alguma simulação.

Problemas ad hoc

- ▶ Normalmente, é o tipo mais simples de problema.
 - ▶ Não requer nenhuma técnica específica.
- ▶ Em alguns casos, possuem enunciados longos, para distrair o leitor do objetivo principal.
 - ▶ As vezes possuem casos limite perigosos, em que é necessário estar atento.
 - ▶ Podem, ainda, ser bem difíceis, sendo necessário uma ideia genial para resolvê-lo.
- ▶ Principais tipos:
 - ▶ Simulação de jogos de cartas, jogos de tabuleiros, palíndromos, anagramas, problemas com cálculo de tempo, teclado de celular, etc.

Problemas ad hoc – Exemplo: Dama

- ▶ O jogo de xadrez possui várias peças com movimentos curiosos: uma delas é a dama, que pode se mover qualquer quantidade de casas na mesma linha, na mesma coluna, ou em uma das duas diagonais, conforme exemplifica a figura.
- ▶ Dada a posição de uma dama em um tabuleiro de xadrez vazio (ou seja, um tabuleiro 8×8 , com 64 casas), de quantos movimentos, no mínimo, ela precisa para chegar em outra casa do tabuleiro?
- ▶ Fonte: Maratona de Programação da SBC 2008.



Problemas ad hoc – Exemplo: Dama

- ▶ **Entrada:** A entrada contém vários casos de teste. A primeira e única linha de cada caso de teste contém quatro inteiros x_1, y_1, x_2 e y_2 ($1 \leq x_1, y_1, x_2, y_2 \leq 8$). A dama começa na casa de coordenadas (x_1, y_1) , e a casa de destino é a casa de coordenadas (x_2, y_2) . No tabuleiro, as colunas são numeradas da esquerda para a direita de 1 a 8 e as linhas de cima para baixo, também de 1 a 8. O final da entrada é indicado por uma linha contendo quatro zeros.
- ▶ **Saída:** Para cada caso de teste da entrada seu programa deve imprimir uma única linha na saída, contendo um número inteiro, indicando o menor número de movimentos necessários para a dama chegar em sua casa de destino.

Formato de Entrada / Saída – Exemplo: Dama

- ▶ Casos de teste únicos: basta ler cada variável, usando `scanf` ou `cin`.

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
4 4 6 2	1

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
3 5 3 5	0

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
5 5 4 3	2

Formato de Entrada / Saída – Exemplo: Dama

- Casos de teste únicos: basta ler cada variável, usando `scanf` ou `cin`.

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
4 4 6 2	1

```
1. ...
2.
3. int x1, y1, x2, y2;
4.
5. scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
6.
7. // ou, em C++:
8.
9. cin >> x1 >> y1 >> x2 >> y2;
```

Formato de Entrada / Saída – Exemplo: Dama

- ▶ **Finalizando em zeros**: basta ler cada variável usando `scanf` ou `cin`, criar um laço de processamento e, a última instrução do laço é a repetição da leitura.

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
4 4 6 2 3 5 3 5 5 5 4 3 0 0 0 0	1 0 2

Formato de Entrada / Saída – Exemplo: Dama

- ▶ **Finalizando em zeros**: basta ler cada variável usando `scanf` ou `cin`, criar um laço de processamento e, a **última** instrução do laço é a repetição da leitura.

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
4 4 6 2 3 5 3 5 5 5 4 3 0 0 0 0	1 0 2

```
1. int x1, y1, x2, y2;  
2.  
3. scanf("%d %d %d %d", &x1, &y1, &x2, &y2);  
4.  
5. while(x1 || y1 || x2 || y2)  
6. {  
7.     ...  
8.     scanf("%d %d %d %d", &x1, &y1, &x2, &y2);  
9. }
```

Formato de Entrada / Saída – Exemplo: Dama

- ▶ **Indicação da quantidade de casos de teste:** basta ler a quantidade **N** de casos de testes e, em um laço que conta **N** vezes, ler as variáveis da entrada.

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
3 4 4 6 2 3 5 3 5 5 5 4 3	1 0 2

Formato de Entrada / Saída – Exemplo: Dama

- ▶ **Indicação da quantidade de casos de teste:** basta ler a quantidade **N** de casos de testes e, em um laço que conta **N** vezes, ler as variáveis da entrada.

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
3 4 4 6 2 3 5 3 5 5 5 4 3	1 0 2

```
1. int n, x1, y1, x2, y2;  
2.  
3. scanf("%d", &n);  
4.  
5. while(n--)  
6. {  
7.     scanf("%d %d %d %d", &x1, &y1, &x2, &y2);  
8.     ...  
9. }
```

Formato de Entrada / Saída – Exemplo: Dama

- ▶ **Entrada finalizando com fim de arquivo (EOF):** não se sabe quantos casos de teste o problema possui. Neste caso, testa-se o `scanf` ou `cin` em um laço.

<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
4 4 6 2 3 5 3 5 5 5 4 3	1 0 2

Formato de Entrada / Saída – Exemplo: Dama

- ▶ **Entrada finalizando com fim de arquivo (EOF):** não se sabe quantos casos de teste o problema possui. Neste caso, testa-se o `scanf` ou `cin` em um laço.

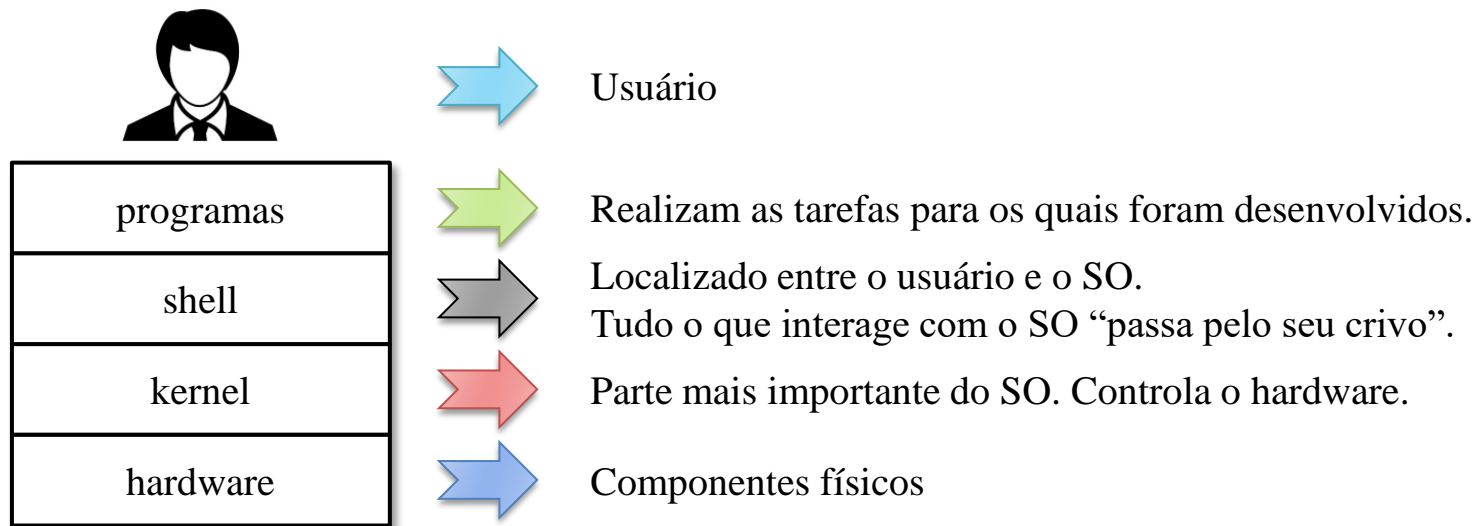
<i>Exemplo de entrada</i>	<i>Exemplo de saída</i>
4 4 6 2 3 5 3 5 5 5 4 3	1 0 2

```
1. int x1, y1, x2, y2;
2.
3. while( scanf("%d %d %d %d", &x1, &y1, &x2, &y2) != EOF ) // em C
4. {
5.     ...
6. }
7.
8. // ou, em C++:
9. while( cin >> x1 >> y1 >> x2 >> y2 ) { ... }
```

Shell

- ▶ Embora não seja uma regra oficial, o computador disponibilizado aos times para a competição costuma vir com o sistema **Maratona-Linux** instalado.
 - ▶ Ou alguma outra distribuição, quase sempre Linux.
 - ▶ Muito raro, mas não impossível, que Windows seja usado.
- ▶ Trata-se de uma modificação do Ubuntu contendo diversos aplicativos, compiladores para as linguagens aceitas e plugins de segurança previamente configurados.
- ▶ Neste caso, conhecer o básico do Linux e do Shell é fundamental para os competidores.

Shell: o ambiente Linux



Shell: comandos básicos

- ▶ `ls`: lista o conteúdo do diretório (pasta) atual.
- ▶ `pwd`: mostra qual é o diretório atual.
- ▶ `cd`: permite trocar de diretório
 - ▶ Exemplo: `cd /var/www`
 - ▶ `cd ..` sobre um nível na árvore de diretórios.
- ▶ `mv`, `cp`: comandos para mover/copiar arquivos e diretórios.
 - ▶ Sintaxe: `mv origem destino`, `cp origem destino`
- ▶ `rm <nome>`: remove arquivos e diretórios (`rmdir`)

Shell: compilação de programas (C/C++)

- ▶ Embora a correção dos programas submetidos, hoje, seja feita de forma automatizada, por meio de scripts, os juízes (pessoas) irão verificar o resultado do “autojudge” antes de enviar o julgamento ao time.
- ▶ Em caso de dúvidas sobre o resultado, os programas serão compilados e testados manualmente, usando um Shell, da seguinte forma:
- ▶ Compilação (onde o time enviou um arquivo chamado `problemaA.c`):
 - ▶ `gcc -o saida problemaA.c`

Shell: compilação de programas (C/C++)

- ▶ Embora a correção dos programas submetidos, hoje, seja feita de forma automatizada, por meio de scripts, os juízes (pessoas) irão verificar o resultado do “autojudge” antes de enviar o julgamento ao time.
- ▶ Em caso de dúvidas sobre o resultado, os programas serão compilados e testados manualmente, usando um Shell, da seguinte forma:
- ▶ Compilação (onde o time enviou um arquivo chamado problemaA.c):

▶ `gcc -o saida problemaA.c`

compilador C nome do arquivo código fonte
 executável gerado

Shell: compilação de programas (C/C++)

- ▶ Embora a correção dos programas submetidos, hoje, seja feita de forma automatizada, por meio de scripts, os juízes (pessoas) irão verificar o resultado do “autojudge” antes de enviar o julgamento ao time.
- ▶ Em caso de dúvidas sobre o resultado, os programas serão compilados e testados manualmente, usando um Shell, da seguinte forma:
- ▶ Compilação (onde o time enviou um arquivo chamado problemaA.c):

▶ `gcc -o saida problemaA.c`

compilador C nome do arquivo executável gerado código fonte

Para submissões em C++:
Arquivo fonte terá extensão **.cpp**
Ao invés do **gcc**, invoca-se o **g++**

Shell: correção do programa

- ▶ Após compilado (e não apresentar erros), o programa pode ser executado. Retomando o exemplo anterior:
 - ▶ `gcc -o saida problemaA.c` (compila)
 - ▶ `./saida` (executa)
- ▶ Invocando simplesmente o nome do arquivo executável gerado (`saida`), o programa se abrirá para que os dados sejam digitados e as respostas obtidas.
- ▶ Logicamente, os juízes não irão fazer isso, pois as baterias de testes podem ser imensas (dezenas de milhares de linhas).
- ▶ Os juízes têm à disposição o gabarito (arquivos de texto) de **entrada** e **saída esperada** para os problemas.
 - ▶ Em nosso exemplo, eles se chamarão `problemaA.in`, `problemaA.sol`

Shell: correção do programa

- ▶ Os juízes irão executar o programa da seguinte forma:
 - ▶ `./saida < problemaA.in > problemaA.out`

Shell: correção do programa

- ▶ Os juízes irão executar o programa da seguinte forma:

▶ `./saida` `< problemaA.in` `> problemaA.out`

`<`: redireciona a entrada
para o arquivo informado

`>`: redireciona a saída
para o arquivo informado


Shell: correção do programa

- ▶ Os juízes irão executar o programa da seguinte forma:
 - ▶ `./saida < problemaA.in > problemaA.out`
- ▶ E, a seguir, irão verificar a diferença entre os dois arquivos
 - ▶ `problemaA.out`: saída gerada pelo código do time
 - ▶ `problemaA.sol`: gabarito dos juízes com as saídas esperadas
- ▶ da seguinte forma:

Shell: correção do programa

- ▶ Os juízes irão executar o programa da seguinte forma:
 - ▶ `./saida < problemaA.in > problemaA.out`
- ▶ E, a seguir, irão verificar a diferença entre os dois arquivos
 - ▶ `problemaA.out`: saída gerada pelo código do time
 - ▶ `problemaA.sol`: gabarito dos juízes com as saídas esperadas
- ▶ da seguinte forma:
 - ▶ `diff problemaA.out problemaA.sol`
- ▶ Se o resultado do `diff` for nulo (isto é, o comando `diff` não imprimir nada), a solução do time está correta. Senão, a solução apresentou diferenças em relação ao gabarito dos juízes.

Julgamento da solução

- ▶ O código é enviado aos juízes e pode receber as seguintes respostas:
- ▶ *Accepted (YES)*: todos os casos de teste submetidos ao seu programa produziram respostas idênticas ao gabarito dos juízes. Balão! 
- ▶ *Wrong answer*: há um ou mais casos de teste (geralmente vários) que não obtiveram a resposta esperada.
 - ▶ Não são dados detalhes sobre o que está errado.
- ▶ *Time Limit Exceeded*: em um ou mais casos de teste, o programa executou por mais tempo que o máximo permitido.
 - ▶ Este erro não significa que as respostas produzidas estão corretas.

Julgamento da solução

- ▶ *Compilation error*: seu programa está com erro de sintaxe.
 - ▶ **NUNCA FAÇA ISSO!!!!**
 - ▶ Verifique variáveis, protótipos de funções, arquivos de cabeçalho, etc.
- ▶ *Runtime error*: o programa “dá crash” durante a execução.
 - ▶ Causas comuns: divisão por zero, violação de limites de vetor, violação de memória, referências inválidas de ponteiros, uso de mais memória que o permitido, etc.
- ▶ *Presentation error*: seu programa produz a resposta correta, porém ela está mal formatada.
 - ▶ Causas comuns: espaços a esquerda/direita, pula linha a mais/menos, etc. Embora a resposta possa estar correta, a solução não é aceita.

Sistema BOCA

- ▶ O Boca é um sistema de administração de competições de programação. Escrito em PHP, usa o banco de dados PostgreSQL.
- ▶ É o sistema empregado em competições oficiais na América Latina.
- ▶ Gerencia diversos aspectos da competição: times, juízes, problemas, placar, etc.
- ▶ As principais características do sistema são:
 - ▶ Portabilidade (Apache + PHP + PostgreSQL).
 - ▶ Controle de concorrência.
 - ▶ Gerenciamento de competições distribuídas em locais distintos.
 - ▶ Interface web de fácil utilização.

Tela de Login

Nesta tela, os times informam o nome de usuário e a senha.

BOCA Login

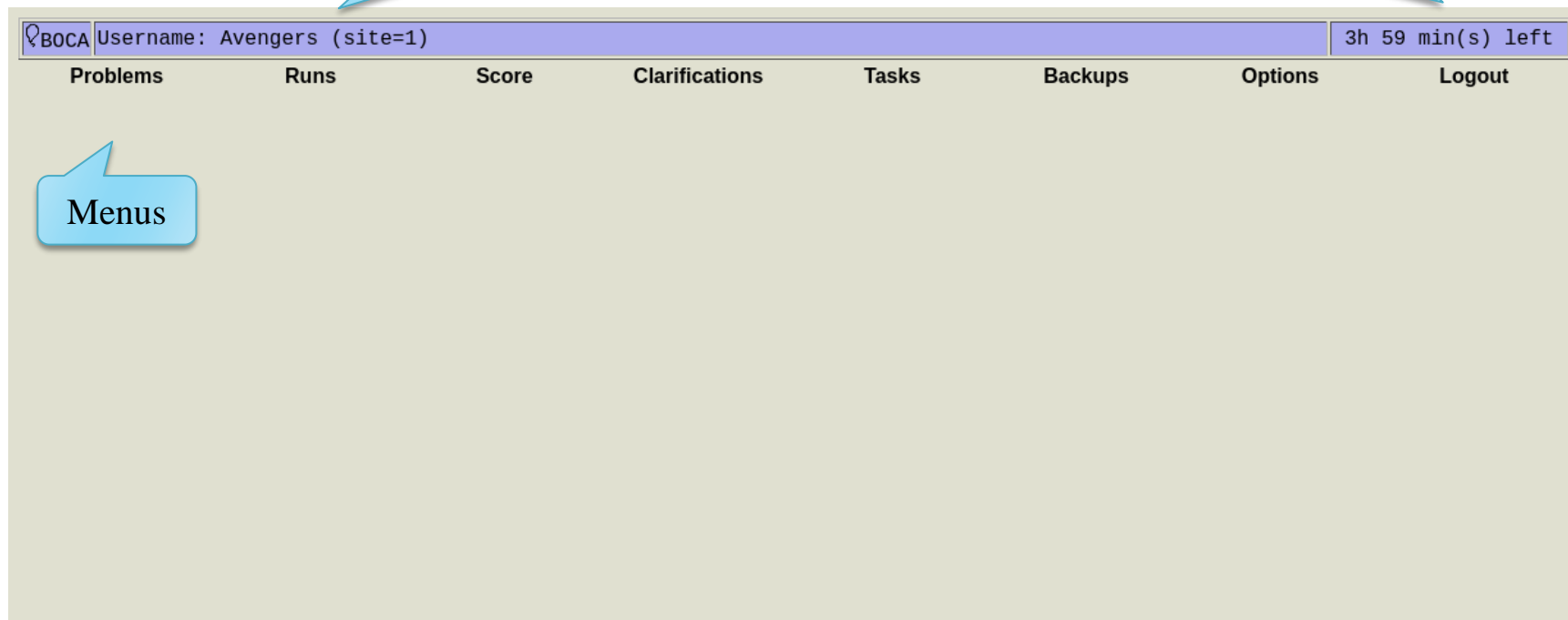
Name	<input type="text"/>	Login
Password	<input type="password"/>	

Estas informações são fornecidas pela organização da competição.

Tela inicial

Nome do time

Tempo restante de prova







Menu: Problems

BOCA Username: Avengers (site=1) 3h 58 min(s) left

Problems Runs Score Clarifications Tasks Backups Options Logout

Information:

Name	Basename	Fullname	Descfile
A 	formiga	The formiga test problem	formiga.pdf
B 	multas	The Poble of multas	multas.pdf
C 	bits	The Problem of bits	bits.pdf
D 	abacaxi	The abacaxi problem	abacaxi.pdf

Apresenta os problemas da prova.
Atente-se para a letra de cada um: A, B, C...

Menu: Runs

BOCA Username: Avengers (site=1) 3h 57 min(s) left

Problems	Runs	Score	Clarifications	Tasks	Backups	Options	Logout
Run #	Time	Problem	Language	Answer	File		
470194301	6	A	C	YES	A.c		
471642102	29	D	C	NO - Time limit exceeded	D.c		
473195903	55	B	C	Not answered yet	B.c		

To submit a program, just fill in the following fields:

Problem: -- v

Language: -- v

Source code: Procurar... Nenhum arquivo selecionado.

Send Clear

Menu usado para submeter a solução de um problema aos juízes.

Menu: Runs

BOCA Username: Avengers (site=1) 3h 57 min(s) left

Problems	Runs	Score	Clarifications	Tasks	Backups	Options	Logout
Run #	Time	Problem	Language	Answer	File		
470194301	6	A	C	YES	A.c		
471642102	29	D	C	NO - Time limit exceeded	D.c		
473195903	55	B	C	Not answered yet	B.c		

To submit a program, just fill in the following fields:

Problem: -- v

Language: -- v

Source code: Procurar... Nenhum arquivo selecionado.

Send Clear

Escolha a letra que identifica o problema, a linguagem utilizada e o arquivo para upload.

Menu: Runs

BOCA Username: Avengers (site=1) 3h 57 min(s) left

Problems	Runs	Score	Clarifications	Tasks	Backups	Options	Logout
Run #	Time	Problem	Language	Answer	File		
470194301	6	A	C	YES	A.c		
471642102	29	D	C	NO - Time limit exceeded	D.c		
473195903	55	B	C	Not answered yet	B.c		

To submit a program, just fill in the following fields:

Problem: -- v

Language: -- v

Source code: Procurar... Nenhum arquivo selecionado.

Send Clear

Escolha a letra que identifica o problema, a linguagem utilizada e o arquivo para upload.

Cuidado para não selecionar opções erradas! Uma vez submetida, a solução será julgada de acordo com aquela configuração.

Menu: Runs

BOCA Username: Avengers (site=1) 3h 57 min(s) left

Problems	Runs	Score	Clarifications	Tasks	Backups	Options	Logout
----------	------	-------	----------------	-------	---------	---------	--------

Run #	Time	Problem	Language	Answer	File
470194301	6	A	C	YES	A.c
471642102	29	D	C	NO - Time limit exceeded	D.c
473195903	55	B	C	Not answered yet	B.c

To submit a program, just fill in the following

Problem: -- v

Language: -- v

Source code: Procurar... Nenhum arquivo selecionado.

Send Clear




Esta tabela contém informações sobre as submissões já realizadas.

Menu: Score

BOCA Username: Avengers (site=1) 3h 56 min(s) left




Problems Runs **Score** Clarifications Tasks Backups Options Logout

Available scores: Global

#	User/Site	Name	A	B	C	D	Total
1	team01/1	Avengers	 1/6			1/-	1 (6)
2	team03/1	Galo Doido		 1/11			1 (11)
3	team04/1	Desesperados				 1/28	1 (28)
4	team02/1	Huey, Dewey, Louie					0 (0)

Placar da competição. Mostra a classificação de todos os times, os problemas resolvidos e o tempo.

Menu: Score

BOCA Username: Avengers (site=1) 3h 56 min(s) left						
Problems	Runs	Score	Clarifications	Tasks	Backups	Options Logout
Available scores: <u>Global</u>						
#	User/Site	Name	A	B	C	D Total
1	team01/1	Avengers	 1/6			1/- 1 (6)
2	team03/1	Galo Doido		 1/11		1 (11)
3	team04/1	Desesperados			 1/28	1 (28)
4	team02/1	Huey, Dewey, Louie				0 (0)

Placar congelado: na última hora da competição, este placar não se altera mais. Porém, pelo menu Runs, o time consegue ver se uma submissão passou ou não; mas não vê o resultado dos outros times.

Menu: Score

BOCA Username: Avengers (site=1) 3h 56 min(s) left						
Problems	Runs	Score	Clarifications	Tasks	Backups	Options Logout
Available scores: <u>Global</u>						
#	User/Site	Name	A	B	C	D Total
1	team01/1	Avengers	1/6			1/- 1 (6)
2	team03/1	Galo Doido		1/11		1 (11)
3	team04/1	Desesperados				1/28 1 (28)
4	team02/1	Huey, Dewey, Louie				0 (0)

Juízes calados: na última meia hora de prova, nem mesmo pelo menu Runs é possível ver o resultado de uma submissão. O resultado final é divulgado após o encerramento da prova.

Menu: Score

BOCA Username: Avengers (site=1) 3h 56 min(s) left						
Problems	Runs	Score	Clarifications	Tasks	Backups	Options Logout
Available scores: <u>Global</u>						
#	User/Site	Name	A	B	C	D Total
1	team01/1	Avengers	1/6			1/- 1 (6)
2	team03/1	Galo Doido		1/11		1 (11)
3	team04/1	Desesperados			1/28	1 (28)
4	team02/1	Huey, Dewey, Louie				0 (0)

Dica: a ordem de dificuldade dos problemas no caderno de prova é aleatória. Pode ser útil olhar o placar e ver qual problema está sendo resolvido primeiro pelos times mais experientes. Isso indica que, possivelmente, aquele problema é mais fácil.

Menu: Clarifications

BOCA Username: Avengers (site=1) 3h 55 min(s) left

Problems Runs Score Clarifications Tasks Backups Options Logout

Time	Problem	Question	Answer
------	---------	----------	--------

NO CLARIFICATIONS AVAILABLE

To submit a clarification, just fill in the following fields

Problem: General ▾

Clarification:

Send Clear

Dúvidas **pertinentes** sobre os problemas podem ser enviadas ao comitê de prova por meio deste menu. Selecione a letra do problema e escreva a dúvida. Respostas dadas a outros times podem aparecer aqui.

Menu: Tasks

BOCA Username: Avengers (site=1) 3h 54 min(s) left

Problems	Runs	Score	Clarifications	Tasks	Backups	Options	Logout
----------	------	-------	----------------	-------	---------	---------	--------

Task #	Time	Description	File	Status
--------	------	-------------	------	--------

NO TASKS FOUND

Upload de arquivo a ser impresso.

To submit a file for printing, just fill in the following field:

File name: Nenhum arquivo selecionado.

If you needed staff assistance, please click on the button above and wait.

Botão S.O.S: usado para requisitar auxílio da organização de prova em situações de emergência.

Dúvidas?



Aula 1:

Problemas Ad hoc

Disciplina: Maratona de Programação 1

Profs. Edmilson Marmo e Luiz Olmes

edmarmo@unifei.edu.br, olmes@unifei.edu.br



Prática

- ▶ Tarefa (para casa) da Aula 01 disponível no Beecrowd.
 - ▶ Prazo: 28 de agosto, 18:00.
- ▶ Warmup de final de aula, usando o Boca
 - ▶ Apenas problemas ad hoc.
- ▶ Endereço: boca.unifei.edu.br