Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

OBI2020

Caderno de Tarefas

Modalidade **Programação** • **Nível 1** • Fase **Local** • **Turno B**

15 e 16 de setembro de 2020

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 5 páginas (não contando a folha de rosto), numeradas de 1 a 5. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como "Digite o dado de entrada:" ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 2 devem ser arquivos com sufixo *.py2*; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py3*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
  - em Pascal: *readln, read, writeln, write*;
  - em C: *scanf, getchar, printf, putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
  - em Python: *read,readline,readlines,input,print,write*
  - em Javascript: *scanf, printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Divisão do Tesouro

Nome do arquivo: "`tesouro.x`", *onde* `x` *deve ser* `c`, `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

O Capitão Olho Roxo e seus marinheiros encontraram uma arca com uma grande quantidade de moedas de ouro idênticas. Para a divisão das moedas, todos concordaram com a seguinte sugestão do Capitão:

- cada marinheiro exceto o Capitão deveria receber exatamente o mesmo número de moedas; e
- o Capitão deveria receber o dobro de moedas que um marinheiro recebe.

Pode ser que o fato de o Capitão ser o único com uma pistola a bordo tenha contribuído para a concordância de todos, mas também contribuiu o fato de que na forma proposta a divisão era perfeita, não sobrando ou faltando moedas.

Dados o número de moedas na arca e o número de marinheiros, escreva um programa para determinar quantas moedas o Capitão Olho Roxo recebeu.

Entrada

A primeira linha da entrada contém um número inteiro A, o número de moedas na arca. A segunda linha contém um inteiro N, o número de marinheiros (não contando o Capitão).

Saída

Seu programa deve produzir na saída uma única linha, contendo um único inteiro, o número de moedas que o Capitão Olho Roxo deve receber.

Restrições

- $3 \leq A \leq 10000$
- $1 \leq N \leq 1000$

Exemplo de entrada 1	**Exemplo de saída 1**
221 11	34

Exemplo de entrada 2	**Exemplo de saída 2**
1000 8	200

Exemplo de entrada 3	**Exemplo de saída 3**
3 1	2

Três por Dois

Nome do arquivo: "`tres.x`", *onde* `x` *deve ser* `c`, `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

Uma loja de chocolates está fazendo uma promoção do tipo "leve três pague dois". Mais precisamente, você pode escolher quaisquer três chocolates da loja e paga apenas pelos dois mais caros que escolher, levando o mais barato gratuitamente. Se você levar mais do que três chocolates, você pode agrupá-los em grupos de três e levar o chocolate mais barato de cada grupo gratuitamente.

Por exemplo, se você escolher chocolates de preços (em reais) 8, 5, 10, 2, 5, 10 e 4, você pode agrupá-los como (8,5,10), (2,5,10) e (4), pagará um total de (10+8) + (10+5) + 4, ou seja, 37 reais. Mas você pode agrupá-los de uma forma melhor e assim conseguir pagar menos. Você consegue ver como?

Dados os preços dos chocolates escolhidos, escreva um programa para determinar o menor preço a pagar.

Entrada

A primeira linha da entrada contém um inteiro N, o número de chocolates escolhidos. Cada uma das N linhas seguintes contém um número inteiro P, o preço de um chocolate escolhido.

Saída

Seu programa deve produzir uma única linha, contendo um único número inteiro, o menor preço a pagar pelos chocolates escolhidos.

Restrições

- $1 \leq N \leq 100000$
- $1 \leq P \leq 1000$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 50 pontos, $N \leq 1000$.

Exemplo de entrada 1	**Exemplo de saída 1**
4 5 6 6 5	17

Exemplo de entrada 2	**Exemplo de saída 2**
4 3 3 3 3	9

Exemplo de entrada 3	**Exemplo de saída 3**
6 7 8 7 7 5 7	29

Exemplo de entrada 4	**Exemplo de saída 4**
7 8 5 10 2 5 10 4	32

Emoticons

Nome do arquivo: "`emoticons.x`", *onde* `x` *deve ser* `c`, `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

Emoticons são símbolos usados para expressar o sentimento de quem escreve uma mensagem. Scott Fahlman, um professor de uma universidade americana, foi o primeiro a propor o uso das sequências de caracteres `:-)` e `:-(`, que viraram respectivamente símbolos para "divertido" e "chateado".

Vamos definir o *sentimento* expresso em uma mensagem como sendo:

- `neutro`: se o número de símbolos "divertido" é igual ao número de símbolos "chateado";
- `divertido`: se o número de símbolos "divertido" é maior do que número de símbolos "chateado";
- `chateado`: se o número de símbolos "chateado" é maior do que número de símbolos "divertido";

Dada uma mensagem composta por uma cadeia de caracteres, escreva um programa para determinar o sentimento expresso na mensagem.

Entrada

A primeira e única linha da entrada contém a mensagem M como uma cadeia de caracteres.

Saída

Seu programa deve produzir uma única linha, contendo uma única palavra, o sentimento expresso na mensagem da entrada.

Restrições

- $3 \leq$ comprimento de $M \leq 280$
- M é composta por letras não acentuadas, espaços em branco e os caracteres '`(`' (abre parênteses), '`)`' (fecha parênteses), '`:`' (dois-pontos), '`-`' (hífen), '`.`' (ponto) e '`,`' (vírgula).

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 25 pontos, há no máximo um emoticon na mensagem.

Exemplo de entrada 1	**Exemplo de saída 1**
`Espero que esteja tudo bem :-)`	`divertido`

Exemplo de entrada 2	**Exemplo de saída 2**
`Achei o filme muito divertido.`	`neutro`

Exemplo de entrada 3	**Exemplo de saída 3**
`:-):-(:-(:-)`	`neutro`

Exemplo de entrada 4	**Exemplo de saída 4**
`Sonhei com a prova :-( (vou estudar).`	`chateado`