Competidor(a): _____

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

OBI2020

Caderno de Tarefas

Modalidade **Programação** • **Nível 2** • Fase **Local** • **Turno B**

15 e 16 de setembro de 2020

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando a folha de rosto), numeradas de 1 a 8. 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.

Irmãos

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

Otávio tem dois irmãos, um mais velho (Orlando) e um mais novo do que ele (Oscar). As idades dos três irmãos formam uma *progressão aritmética*: a diferença de idade dos dois irmãos mais novos (Otávio e Oscar) é igual à diferença de idade dos dois irmãos mais velhos (Orlando e Otávio).

Dadas as idades de Otávio e de seu irmão mais novo, escreva um programa para determinar a idade do irmão mais velho.

Entrada

A primeira linha da entrada contém um inteiro N, a idade do irmão mais novo de Otávio. A segunda linha contém um inteiro M, a idade de Otávio.

Saída

Seu programa deve produzir na saída uma única linha, contendo um único número inteiro, a idade do irmão mais velho de Otávio.

Restrições

- $1 \leq N \leq 40$
- $N \leq M \leq 40$

Exemplo de entrada 1	**Exemplo de saída 1**
`13` `16`	`19`

Exemplo de entrada 2	**Exemplo de saída 2**
`14` `14`	`14`

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

Camisetas da Olimpíada

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

A Olimpíada Municipal de Programação vai distribuir camisetas para os melhores colocados, e por isso solicitou que os premiados informassem o tamanho preferido da camiseta, entre os tamanhos pequeno e médio.

A empresa que confeccionou as camisetas, por uma falha, pode ter se enganado na quantidade de camisetas para cada tamanho. Foram produzidas camisetas em número suficiente para todos os premiados, mas talvez não do tamanho preferido.

Dadas a lista com os tamanhos preferidos pelos premiados e a quantidade de camisetas de cada tamanho produzidas pela empresa, escreva um programa para determinar se todos os premiados receberão camisetas do tamanho escolhido.

Entrada

A primeira linha contém um inteiro N, o número de premiados. A segunda linha contém N inteiros T_i, indicando os tamanhos solicitados pelos premiados, sendo que $T_i = 1$ representa o tamanho pequeno e $T_i = 2$ representa o tamanho médio. A terceira linha contém um inteiro P, o número de camisetas de tamanho pequeno produzidas. A quarta e última contém um inteiro M, o número de camisetas de tamanho médio produzidas.

Saída

Seu programa deve produzir uma única linha, contendo um único caractere, que deve ser a letra maiúscula '`S`' se todos os premiados serão atendidos com a camiseta do tamanho que escolheram, ou a letra maiúscula '`N`' caso contrário.

Restrições

- $1 \leq N \leq 1000$
- $0 \leq P \leq 1000$
- $0 \leq M \leq 1000$
- $N \leq P + M$
- $1 \leq X_i \leq 2$ para $1 \leq i \leq N$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $1 \leq N \leq 10$.

Exemplo de entrada 1	**Exemplo de saída 1**
5 1 1 2 1 2 3 2	S

Exemplo de entrada 2	**Exemplo de saída 2**
4 2 2 2 2 1 3	N

Exemplo de entrada 3	**Exemplo de saída 3**
6 1 1 1 2 1 1 4 4	N

Ralouim

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

Para a tradicional festa infantil de Ralouim, o rei da Nlogônia instalou tendas de distribuição de guloseimas no seu extenso Jardim Real, onde está também situado o Palácio Real.

Cada tenda tem uma quantidade ilimitada de guloseimas. As crianças devem sair do Palácio Real e visitar as tendas para ganhar guloseimas, mas o Rei estabeleceu algumas regras:

- a cada visita a uma tenda, a criança ganha exatamente uma guloseima.
- uma tenda pode ser visitada mais de uma vez pela mesma criança, desde que as visitas não sejam consecutivas (ou seja, uma imediatamente após a outra).
- as distâncias que uma criança percorre para chegar à "próxima tenda" devem ser estritamente decrescentes. Ou seja, a distância que a criança percorre do Palácio até a primeira tenda que a criança visita deve ser maior do que a distância que a criança percorre entre a primeira tenda e a segunda tenda, que por sua vez deve ser maior do que a distância que a criança percorre entre a segunda tenda e a terceira tenda, e assim por diante.

Pedrinho percebeu que se planejar direito suas visitas, pode ganhar muitas guloseimas! Escreva um programa para ajudar Pedrinho a ganhar o maior número possível de guloseimas no Ralouim.

Entrada

A primeira linha da entrada contém um inteiro N, o número de tendas. Cada uma das N linhas seguintes contém dois inteiros X e Y, as coordenadas de uma tenda no Jardim Real. A localização do Palácio Real é (0,0) e não existe tenda com essas coordenadas, todas as tendas têm localizações distintas.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o maior número de guloseimas que Pedrinho pode ganhar.

Restrições

- $1 \leq N \leq 2000$
- $-10000 \leq X \leq 10000$
- $-10000 \leq Y \leq 10000$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $1 \leq N \leq 50$.
- Para um conjunto de casos de testes valendo 40 pontos adicionais, $1 \leq N \leq 200$.
- Para um conjunto de casos de testess valendo 40 pontos adicionais, nenhuma restrição adicional.

Exemplo de entrada 1	**Exemplo de saída 1**
4 6 0 5 0 1 0 2 0	5

Exemplo de entrada 2	**Exemplo de saída 2**
2 0 3 3 0	1

Exemplo de entrada 3	**Exemplo de saída 3**
5 6 8 2 1 2 2 2 3 5 9	6