UNIVERSIDADE FEDERAL DE ITAJUBÁ

ECOX21 - MARATONA DE PROGRAMAÇÃO I

Profs. Edmilson Marmo e Luiz Olmes

Warmup 2

18/09/2024

Regras

1. Há 6 problemas que devem ser resolvidos no tempo estipulado.
2. Os dados de entrada devem ser lidos a partir da entrada padrão.
3. Os dados de saída devem ser escritos na saída padrão.
4. Quando uma linha contém vários valores, eles estarão separados por um único espaço. Não há espaços extras na entrada/saída. Não há linhas em branco na entrada.
5. A entrada e a saída usam o alfabeto latino. Não haverá letras com til, acentos, tremas ou outros sinais diacríticos.
6. Todas as linhas da entrada e da saída, incluindo a última contêm o caractere de fim de linha.
7. O código fonte de cada solução deve ser enviado pelo Boca: `boca.unifei.edu.br`

Ambiente de Testes

A correção das soluções enviadas é realizada no sistema operacional Red Hat Enterprise Linux, versão 8.6 (Ootpa), usando os seguintes compiladores/interpretadores:

```
C: gcc versão 8.5.0 20210514 (Red Hat 8.5.0-10)
C++: g++ versão 8.5.0 20210514 (Red Hat 8.5.0-10)
Java: openjdk versão 1.8.0_342
Python: python3 versão 3.6.8
```

Limites

```
Memória (C, C++, Python): 1GB
Memória (Java): 1GB + 100MB stack
Tamanho máximo do código fonte: 100KB
Tamanho máximo do arquivo executável: 1MB
```

Códigos que extrapolem os limites permitidos receberão *Runtime Error* como resposta.

Comandos de Compilação

```
C: gcc -g -O2 -std=gnu11 -static -lm
C++: g++ -g -O2 -std=gnu++17 -static -lm
Java: javac
```

Códigos C/C++

- O programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- Para entradas grandes, objetos `iostream` podem ser lentos, devido a questões de sincronização de buffer com a biblioteca `stdio`. Recomenda-se desabilitar este mecanismo de sincronização em programas que empreagam `std::cin` e `std::cout` através dos seguintes comandos:

  ```
  std::ios::sync_with_stdio(0);
  std::cin.tie(0);
  ```

  Note que, neste caso, deve-se evitar usar `printf` e `scanf` no mesmo programa, pois a separação dos buffers pode levar a resultados inesperados.

Códigos Java

- O programa não deve estar encapsulado em um `package`.
- Para cada problema, o arquivo `.java` e a `public class` devem ter o mesmo nome `basename` mostrado no Boca.
- Comando de execução: `java -Xms1024m -Xmx1024m -Xss100m`

Códigos Python

- Apenas Python 3 é suportado. Python 3 não é compatível com Python 2.
- **Atenção**: não é garantido que soluções escritas em Python executarão dentro do tempo limite especificado para cada problema.
- Comando de execução: `python3`

Problema $\mathcal{A}$

AJUDANDO O OG

Autoria: Edmilson Marmo
Timelimit: 1s

Og, o ogro, possui vários filhos. E seus filhos, por sua vez, possuem vários filhos. Og quer saber quantos netos ele tem. Mas ogros, como você sabe, são péssimos em matemática. Portanto, Og quer sua ajuda: dado o número de filhos que cada filho de Og tem, determine o número total de netos de Og.

Entrada

A entrada começa com uma linha contendo um inteiro T ($1 \leq T \leq 20$), que representa o número de casos de teste. Cada caso de teste é descrito em duas linhas. A primeira linha contém um inteiro N ($1 \leq N \leq 1000$), indicando o número de filhos de Og. A segunda linha de cada caso de teste possui N inteiros $F_1, F_2, \ldots, F_N$. O número F_i ($0 \leq F_i \leq 1000$, para todo i entre 1 e N inclusive) representa o número de filhos que o i-ésimo filho de Og possui.

Saída

Para cada caso de teste, imprima uma linha contendo um único inteiro: o número de netos de Og.

Exemplos

Entrada	*Saída*
2 3 7 5 9 2 0 98	21 98

Problema $\mathcal{B}$

ERRO DOS PROFESSORES

Autoria: Edmilson Marmo
Timelimit: 1s

Os professores da disciplina de Maratona de Programação I enviaram uma mensagem para os alunos informando que não haveria aula presencial na primeira semana de setembro para que os alunos pudessem participar da Semana de Computação da universidade.

Após cadastrar a mensagem no sistema acadêmico, o prof. Edmilson percebeu um erro de digitação na mensagem:

> [...] Estamos **flexibibilizando** este prazo devido à Semana de Computação e outros eventos que estão acontecendo na UNIFEI [...].

Como não havia mais condições de alterar a mensagem, que já tinha sido enviada, ele aproveitou o erro para implementar um programa que verifique se suas mensagens não vão apresentar o mesmo erro no futuro. Assim, também gostaríamos que vocês fizessem um código para resolver o mesmo problema.

Entrada

A entrada contém vários casos de teste. A única linha de um caso de teste contém uma frase com várias palavras, sem acentuação ou caracter especial. Sua missão é verificar quantas repetições de dois caracteres aparecem no texto. A sua solução não fará distinção entre letras maiúsculas e letras minúsculas. O final da entrada é representado pelo fim do arquivo (`EOF`). Cada frase possui, no máximo, 2000 caracteres.

Saída

Para cada caso de teste, imprima um inteiro indicando a quantidade de repetições encontrada.

Exemplos

Entrada	*Saída*

`Estamos flexibibilizando este prazo`
`A baba nao veio trabalhar`
`Eu nao vou`
	`1`
	`1`
	`0`

Problema C

CÁLCULO DA SOMA

Autoria: Luiz Olmes
Timelimit: 0.1s

Dado um vetor A com N números inteiros, você deve realizar as seguintes operações:

Operação 1: *op1(E, D)*

Dados dois inteiros E e D, deve-se retornar a soma de todos os elementos entre os índices E e D do vetor (E e D inclusive). Isto é, se os elementos do vetor são $A_1, A_2, A_3, \ldots, A_{N-1}, A_N$, mostre a soma dos elementos $A_E, A_{E+1}, A_{E+2}, \ldots, A_{D-1}, A_D$.

Operação 2: *op2(X)*

Dado um inteiro X, insira X na primeira posição do vetor. Após esta operação, X se torna o elemento A_1, o elemento A_1 anterior se torna A_2, e assim por diante. O tamanho do vetor aumenta em 1 unidade.

Entrada

A entrada é composta por diversos casos de teste. A primeira linha de um caso de teste contém um inteiro N ($1 \leq N \leq 10^7$) que indica a quantidade de elementos do vetor. A segunda linha de um caso de teste contém N inteiros $A_1, A_2, \ldots, A_N$ separados por um espaço ($-10^9 \leq A_i \leq 10^9$). A terceira linha de um caso de teste contém um inteiro Q ($1 \leq Q \leq 10^5$) indicando a quantidade de operações a serem realizadas. As próximas Q linhas descrevem uma operação. As operações 1 e 2 estão, respectivamente, no seguinte formato:

1 E D, onde $1 \leq E \leq D \leq N$.

2 X, onde $-10^9 \leq X \leq 10^9$.

Um valor $N = 0$ indica o fim da entrada.

Saída

Para cada operação do tipo 1, imprima o valor da soma dos elementos em uma linha separada. Nenhuma saída é necessária para operações do tipo 2.

Exemplos

Entrada	*Saída*
10 1 2 3 4 5 6 7 8 9 10 6 1 1 10 1 1 1 1 10 10 1 2 7 2 1 1 1 11 0	55 1 10 27 56

Problema $\mathcal{D}$

REVISÃO

Autoria: Edmilson Marmo
Timelimit: 1s

Após a aula do prof. Luiz Olmes sobre STL e a lista de exercícios proposta no BeeCrowd, queremos verificar se vocês compreenderam como se utiliza as estruturas básicas Fila e Pilha.

Neste exercício, vamos testar apenas o seu entendimento de uma estrutura do tipo *Last-In-First-Out*. Portanto, dada uma sequência de operações que retornam valores, você deve verificar se a estrutura utilizada é uma pilha.

Entrada

Existem muitos casos de testes. Cada caso de teste começa com a linha contando um único inteiro N ($1 \leq N \leq 1000$). Cada uma das seguintes N linhas é um comando do tipo 1, ou um número inteiro 2, seguido de um número inteiro X. Isso significa que depois de executar um comando do tipo 2, obtemos um elemento X sem erros. O valor de X é sempre um número inteiro, positivo e não maior do que 100. O final da entrada é determinado pelo final do arquivo (`EOF`).

Saída

Para cada caso de teste, responda "`Eh uma pilha`" ou "`Nao eh uma pilha`".

Exemplos

Entrada	*Saída*
6 1 1 1 2 1 3 2 1 2 2 2 3 6 1 1 1 2 1 3 2 3 2 2 2 1 2 1 1 2 2 4 1 2 1 1 2 1 2 2	Nao eh uma pilha Eh uma pilha Nao eh uma pilha Eh uma pilha

Problema $\mathcal{E}$

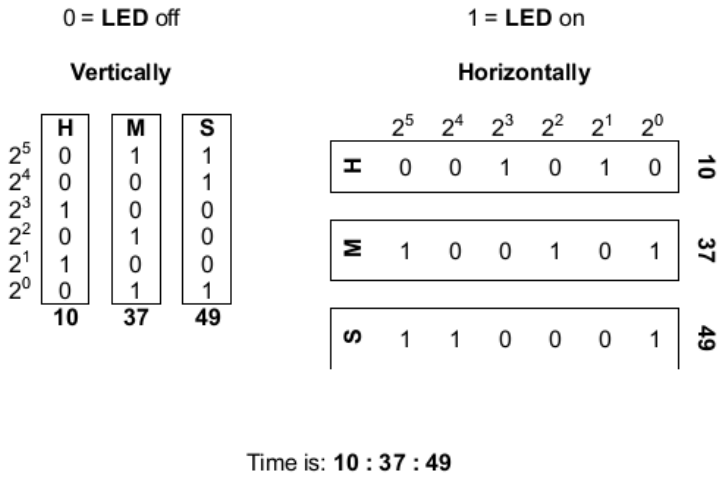
RELÓGIO BINÁRIO

Autoria: Edmilson Marmo
Timelimit: 1s

Um relógio binário é um relógio que mostra o tempo em formato binário. Seu tipo mais comum utiliza três colunas ou três filas de LEDs, que representam zeros e uns. Cada coluna (ou linha) representa o valor de uma unidade de tempo (hora, minuto ou segundo).

Quando utilizamos três colunas (verticalmente), a linha de baixo de cada coluna representa 2^0 (ou 1). Cada linha acima representa potências mais altas de dois, até 2^5 (ou 32). Para ler cada unidade individual (horas, minutos e segundos), o usuário adiciona os valores que cada LED iluminado representa e, depois, os lê da esquerda para a direita: a primeira coluna representa a hora, a segunda minutos, e a última representa os segundos.

Quando utilizamos três linhas (horizontalmente), a coluna mais a direita de cada linha representa 2^0 (ou 1), e cada coluna à sua esquerda representa potências mais altas de dois, até 2^5 (ou 32). Para ler as unidades individuais (horas, minutos e segundos) do tempo, o usuário adiciona os valores que cada LED iluminado representa e depois lê de cima para baixo: a linha de cima representa a hora, a seguinte representa os minutos e a de baixo representa os segundos. Por exemplo:



Para este problema, você irá ler cada tempo em formato tradicional e escrever na saída o valor que seria obtido por um relógio binário se utilizasse os padrões vertical e horizontal. A saída será formada pela concatenação dos bits em cada linha para formar duas cadeias de 18 caracteres cada. Por exemplo, 10:37:49 seria escrito verticalmente como 011001100010100011 e horizontalmente como 001010100101110001.

Entrada

A primeira linha da entrada contém um único número inteiro N ($1 \leq N \leq 1000$) que é o número casos de teste que serão utilizados. Cada caso de teste consiste de uma única linha contendo o horário em formato tradicional (`HH:mm:ss`).

Saída

Para cada caso de teste, você deve gerar uma linha na saída com os seguintes valores: o número do caso de teste, como um inteiro decimal (começando a contagem em 1), um espaço, o tempo binário

em formato vertical (18 dígitos binários), outro espaço, e o tempo binário em formato horizontal (mais 18 dígitos).

Exemplos

Entrada	*Saída*
2 10:37:49 00:00:01	1 011001100010100011 001010100101110001 2 000000000000000001 000000000000000001

Problema $\mathcal{F}$

ET'S

Autoria: Edmilson Marmo
Timelimit: 3s

Varg Inha, em sua interminável busca para provar a existência de extraterrestres, conseguiu um número de fotografias noturnas tomadas por um grupo de pesquisadores que está examinando gás do pântano brilhante. Varg quer ver se alguma das fotos mostram, não gás do pântano, mas Homens Cinzentos em ternos brilhantes.

As fotografias consistem em pontos brilhantes que aparecem de encontro a um fundo preto. Infelizmente, no momento em que as fotos foram tiradas, trens estavam passando pela área (há sobre o pântano uma linha de trem de uma famosa empresa mineradora), e as luzes ocasionais das janelas do trem também apareceram nas fotografias.

Varg, sendo um pesquisador experiente, quer eliminar esses pontos das imagens. Ele não pode dizer, a partir das fotos, exatamente onde as faixas estão, ou de que direção as fotos foram tiradas, mas ele sabe em quais trechos as áreas são perfeitamente retas, então ele decidiu realizar a seguinte abordagem: ele vai encontrar a linha com o número máximo de pontos sobre ela e, se houver quatro ou mais pontos na linha, ele eliminará aqueles pontos de seus cálculos, supondo que estes pontos são janelas no trem.

Se duas ou mais linhas tiverem o número máximo de pontos, Varg irá apenas selecionar aleatoriamente um conjunto e excluí-lo da foto (ele não é tão exigente - afinal, ele acredita em extraterrestres). Se houver menos que quatro pontos situados ao longo de uma linha comum, Varg assumirá que não há nenhum trem na fotografia e não apagará quaisquer pontos.

Por favor, escreva um programa para ele processar um conjunto de fotografias.

Entrada

A entrada é constituída de vários casos de teste. Cada caso de teste é uma fotografia descrita por uma linha contendo um inteiro N ($0 \leq N \leq 1000$) representando o número de pontos na fotografia, seguido por N linhas contendo a coordenada dos pontos em inteiro, um par (X, Y) por linha. Todas as coordenadas estão compreendidas entre 0 e 10000. Após a descrição da última foto, segue uma linha contendo zero (0), para marcar o final da entrada. Esta linha não será processada.

Saída

Para cada caso de teste seu programa deve produzir uma linha com o número da foto seguido pelo número de pontos que foram eliminados da fotografia, conforme exemplo de saída seguir.

Exemplos

Entrada	*Saída*
6 0 1 0 2 1 2 2 2 4 5 5 6 4 3 5 4 4 6 5 7 4 0	Foto 1: 4 pontos eliminados Foto 2: 0 pontos eliminados