Maratona de Programação FACENS 2025

CADERNO DE PROBLEMAS

Informações gerais

Este caderno contém 11 problemas; as páginas estão numeradas de 1 a 21, não contando esta folha de rosto.

Verifique se o caderno está completo.

1) Sobre os nomes dos programas

Sua solução deve ser chamada arquivo.c, arquivo.cpp, arquivo.py ou arquivo.java, onde "arquivo" é especificado em cada problema. Lembre que em Java o nome da classe principal deve ser igual ao nome do arquivo.

2) Sobre a entrada

A entrada de seu programa deve ser lida da entrada padrão.

A entrada pode ser composta de um ou mais casos de teste, descritos em um número de linhas que depende do problema.

Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco. Cada linha, incluindo a última, contém exatamente um caractere final-de-linha.

3) Sobre a saída

A saída de seu programa deve ser escrita na saída padrão.

Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.

Cada linha, incluindo a última, deve conter exatamente um caractere final-de-linha.

3) Sobre o limite de tempo

O limite de tempo dos exercícios para C, C++ e Python 2/3 é 1 segundo.
O limite de tempo para Java é 5 segundos (sinceramente, Java demora muito para compilar!)

PROBLEMA A: PADRAO

Arquivo: `padrao.[cpp/c/java/py]`

Cor: laranja

Descrição do problema

Você é o mais novo programador contratado em uma empresa de tecnologia inovadora e foi incumbido de uma tarefa desafiadora, que vai ser muito útil para todos na empresa.

Mas, claro, quase toda empresa tem algo legado e aí você entra. Para fazer sua tarefa no trabalho você tem que determinar se cada sequência segue o padrão “resource/id/method” onde “resource” pode ser uma sequência alfanumérica seguida de uma “/” e por “id”, que obrigatoriamente é um número inteiro, seguido por outra “/” e mais uma sequência alfanumérica.

Entrada

A primeira linha contém um número inteiro N ($1 \leq N \leq 100$), representando a quantidade de casos de teste.

Em seguida, haverá N linhas, cada uma contendo uma string S com no máximo 500 caracteres, que deve ser processada conforme as regras descritas no problema.

Saída

Você deve imprimir “Segue padrao” quando a string S segue o padrão descrito. Caso a string não siga o pradrao você deve imprimir “Nao segue o padrao”. A saída deve ser seguida por fim de linha

Exemplos de testes

Entrada	**Saída**
2 Pessoa/123/contratar Pessoa/abc/contratar	Segue o padrao Nao segue o padrao

PROBLEMA B: BALÕES RGB

Arquivo: `baloes.[cpp/c/java/py]`

Cor: azul cobalto

Descrição do problema

Na cidade de Balonópolis, está acontecendo o Festival Anual dos Balões Coloridos. Os balões são iluminados por LEDs internos, e cada balão pode brilhar com uma cor específica baseada na ativação de três canais de cor: R (vermelho), G (verde) e B (azul). Esses canais são representados por três números binários (0 ou 1), indicando se o canal está desligado (0) ou ligado (1).

Os organizadores querem que você os ajude a identificar rapidamente a cor final de um balão com base na combinação dos três canais. As cores possíveis são:

R	G	B	Cor resultante
0	0	0	Preto
0	0	1	Azul
0	1	0	Verde
0	1	1	Ciano
1	0	0	Vermelho
1	0	1	Rosa
1	1	0	Amarelo
1	1	1	Branco

Entrada

A primeira linha contém um número inteiro N ($1 \leq N \leq 100$), representando a quantidade de casos de teste.

Cada uma das próximas N linhas contém três números inteiros R, G e B separados por espaço, cada um podendo ser 0 ou 1. Eles representam, nessa ordem, os valores dos canais **R, G e B**.

Saída

Para cada caso de teste, imprima uma linha contendo o nome da cor resultante da combinação.

Exemplos de Testes

Entrada	Saída
3 1 0 0 0 1 1 1 1 1	Vermelho Ciano Branco

PROBLEMA C: AVALIAÇÃO ESCOLAR

Arquivo: `avaliacao.[cpp/c/java/py]`

Cor: verde folha

Descrição do problema

A professora Helena está corrigindo as provas dos alunos da sua turma e precisa calcular a média aritmética das notas de cada aluno para saber quem atingiu a média mínima.

Crie um programa que, dadas as quatro notas de um aluno, calcule e exiba a média aritmética dessas notas com duas casas decimais.

Entrada

A primeira linha da entrada contém um número inteiro T ($1 \leq T \leq 100$), representando o número de alunos a serem avaliados.

Em seguida, para cada um dos T casos de teste, haverá uma linha com quatro números inteiros representando as notas (de 0 a 10) recebidas por um aluno.

Saída

Para cada aluno, imprima uma linha com a média aritmética de suas quatro notas, com duas casas decimais.

Exemplos de testes

Entrada	**Saída**
2 1 1 1 1 6 9 7 5	1.00 6.75

PROBLEMA D: DE VOLTA PARA O FUTURO

Arquivo: `fluxocapacitor.[cpp/c/java/py]`

Cor: lilás

Descrição do problema

Durante uma de suas viagens temporais, o Dr. Emmett Brown descobre um defeito no capacitor de fluxo do DeLorean que só pode ser corrigido com uma sequência numérica especial. Para que o carro volte a viajar no tempo, é necessário inserir uma sequência de números naturais dentro de uma faixa de valores. No entanto, qualquer número pertencente à sequência de Fibonacci (1, 2, 3, 5, 8...) causará uma pane temporal.

Marty McFly, ansioso para retornar ao presente e evitar consequências catastróficas no espaço-tempo, se oferece para encontrar a sequência correta. Enquanto isso, Dr. Brown é perseguido por agentes temporais que querem impedir qualquer alteração no passado. A única chance de sucesso é inserir a sequência exata a tempo, evitando os números proibidos.

Entrada

A entrada começa com um número inteiro T ($1 \leq T \leq 100$), representando a quantidade de casos de teste. Cada um dos T casos de teste contém dois números inteiros M e N ($0 \leq M, N \leq 100$), separados por espaço, que representam a faixa de valores da sequência.

Saída

Para cada entrada é produzida uma saída com a sequência numérica de código que faz com que o DeLorean funcione. Se nenhum número fizer parte da sequência deverá ser impresso ERRO. Também deve ser impresso ERRO caso o usuário não informe o início da faixa menor que o fim da faixa ou qualquer valor fora do faixa aceita.

Exemplos de testes

Entrada	**Saída**
5 2 1 1 3 3 7 1 10 20 30	ERRO ERRO 4 6 7 4 6 7 9 10 20 22 23 24 25 26 27 28 29 30

PROBLEMA E: ESTRELA SILVA – PRIORIDADE MÁXIMA

Arquivo: `estrelasilva.[cpp/c/java/py]`

Cor: branco polar

Descrição do problema

A equipe da nave Estrela de Silva está analisando amostras raras coletadas em planetas desconhecidos. Cada amostra possui quatro atributos:

- **nome (nome do elemento):** Composto apenas por letras minúsculas sem acento (entre 1 e 50 caracteres), sem espaços
- **categoria (categoria do elemento):** Uma das cinco possíveis: cristalino, gasoso, metalico, organico ou sintetico
- **analise_quimica:** Um valor inteiro associado à análise química entre entre 0 e 10^5
- **estabilidade_energetica:** Um valor em ponto flutuante representando estabilidade energética de até duas casas decimais entre 0.00 e 1000.00

As categorias possuem prioridade fixa, conforme a seguinte ordem (da mais alta para a mais baixa):

- cristalino
- gasoso
- metalico
- organico
- sintetico

A nave deseja identificar os três elementos mais relevantes em cada planeta visitado, respeitando dois critérios:

Prioridade da categoria: elementos de categorias mais prioritárias sempre vêm antes.

Ordenação interna: dentro da mesma categoria, os elementos devem ser ordenados com base em regras específicas:

- cristalino: ordenar por analise_quimica (menor primeiro), depois por nome
- gasoso: ordenar por estabilidade_energetica (maior primeiro), depois por nome
- metalico: ordenar por nome (alfabeticamente) e, em caso de empate, pelo analise_quimica (maior primeiro)
- organico: ordenar por nome, depois por estabilidade_energetica (menor primeiro)
- sintetico: ordenar por estabilidade_energetica (menor primeiro), depois por analise_quimica (maior primeiro)

Entrada

A primeira linha contém um inteiro N (1 ≤ N ≤ 10^5), representando a quantidade de planetas visitados. Cada um dos próximos N blocos contém um inteiro M (1 ≤ M ≤ 10^5), representando a quantidade de elementos analisados em cada planeta. Cada uma das próximas M linhas descreve um elemento no seguinte formato:

nome categoria analise_quimica estabilidade_energetica

Saída

Imprima os três elementos mais relevantes em cada planeta, um por linha, seguindo os critérios de prioridade e ordenação acima. Cada linha deve conter:

nome analise_quimica estabilidade_energetica

Entre os casos de teste, imprima uma linha em branco para separá-los. Após o último caso, não imprima linha em branco extra, mas finalize a saída com um único \n.

Exemplos de testes

Entrada	Saída
2 7 zirconio metalico 500 3.50 neon gasoso 300 0.90 helio gasoso 200 2.10 diamante cristalino 50 1.20 polimero organico 200 2.30 carbonio cristalino 80 0.50 ferro metalico 450 4.10 3 diamante cristalino 50 1.20 zirconio metalico 500 3.50 neon gasoso 300 0.90	diamante 50 1.20 carbonio 80 0.50 helio 200 2.10 diamante 50 1.20 neon 300 0.90 zirconio 500 3.50

PROBLEMA F: PIZZAS

Arquivo: `pizzas.[cpp/c/java/py]`

Cor: preto ebano

Descrição do problema

Um grupo de amigos se reuniu para uma tradicional noite de jogos e pizzas. Eles decidiram pedir P pizzas, e a pizzaria local oferece S sabores diferentes. Cada pizza pode ter:

- Apenas 1 sabor (ex: "Calabresa"), ou;
- 2 sabores distintos (ex: "Calabresa e Quatro Queijos"), nesse caso, os sabores são considerados sem ordem, ou seja, "Calabresa e Quatro Queijos" é igual a "Quatro Queijos e Calabresa".

Os amigos estão curiosos para saber de quantas formas diferentes eles podem pedir P pizzas, considerando que:

- Cada pizza pode ter 1 ou 2 sabores distintos.
- A ordem das pizzas no pedido não importa, ou seja, dois pedidos com as mesmas pizzas em ordens diferentes são considerados iguais.
- A ordem dos sabores em uma pizza de dois sabores também não importa.

Ajude-os a calcular o número total de maneiras distintas de fazer o pedido! Você deve ajudar a calcular isso para N casos de teste diferentes.

Entrada

A primeira linha contém um número inteiro N ($1 \leq N \leq 100$), representando o número de casos de teste.

Cada uma das próximas N linhas contém dois inteiros P e S:

- P ($1 \leq P \leq 10$) — o número de pizzas no pedido.
- S ($1 \leq S \leq 15$) — o número de sabores diferentes disponíveis.

Saída

Para cada caso de teste, imprima uma linha com a quantidade de maneiras distintas de fazer o pedido de P pizzas com S sabores disponíveis.

Dicas

- O número de combinações possíveis de sabores por pizza é:
  - S opções com 1 sabor.
  - S × (S - 1) / 2 combinações possíveis com 2 sabores (sem ordem).
- Este é um problema de contagem com repetições e equivalências, pense em como gerar todas as combinações únicas de P pizzas considerando os critérios.

Exemplos de testes

Entrada	Saída
2 2 3 3 2	21 10

Explicações

Caso 1:

- 6 pizzas possíveis: A, B, C, AB, AC, BC
- 21 combinações:

A A A B A C A AB (AB A, A BA, BA A são equivalentes) A AC A BC B B B C B AB B AC	B BC C C C AB C AC C BC AB AB AB AC AB BC AC AC AC BC BC BC

Caso 2:

- 3 pizzas possíveis: A, B, AB
- 10 combinações:

A A A A A B A A AB A B B A B AB	A AB AB B B B B B AB B AB AB AB AB AB

PROBLEMA G: A INVOCAÇÃO DO YAMI YUGI

Arquivo: `invocacao.[cpp/c/java/py]`

Cor: verde limão

Descrição do problema

Sempre que Yugi Muto entra em apuros, ele ativa seu Enigma do Milênio e se transforma em Yami Yugi, o seu alter ego mais confiante e habilidoso. No momento da transformação, Yugi grita com toda sua força:

yu gi ohhhhhhhh

No entanto, a força de seu grito depende de sua energia espiritual no momento.

Você foi encarregado de programar essa lógica para que o sistema da arena duelo reproduza fielmente o grito de transformação de Yugi, de acordo com sua energia.

Entrada

A entrada começa com um número inteiro T ($1 \leq T \leq 100$), representando a quantidade de casos de teste. Em seguida, haverá T linhas, cada uma com um inteiro E ($0 \leq E \leq 50$), representando a energia de Yugi naquele momento.

Saída

Para cada caso de teste, imprima uma linha com o grito do yugi para cada energia.

Exemplos de testes

Entrada	**Saída**
3 1 3 0	yu gi ohhh yu gi ohhhhhhh yu gi oh

PROBLEMA H: CESAR'S GAME

Arquivo: `cesar.[cpp/c/java/py]`

Cor: amarelo citrino

Descrição do problema

César é um garoto que adora inovar criando e adaptando jogos com os seus amigos, sua mais recente criação é o César's Game, que se baseia em: dada uma lista de inteiros l de tamanho N, para cada elemento da lista, contabilizamos quantos elementos da lista l são menores que l[i] e, em seguida, retornamos uma nova lista de resposta R, também de tamanho N, onde R[i] será a quantidade de elementos em l menores que l[i].

Por haver uma grande dificuldade em executar o algoritmo do jogo manualmente para um valor N muito grande, César pediu a sua ajuda para desenvolver um programa que resolva esse problema.

Entrada

A entrada começa com um número inteiro T ($1 \leq T \leq 100$), representando a quantidade de casos de teste. Em seguida, um inteiro N, onde $1 \leq N \leq 10^4$, sinalizando o tamanho das listas e uma lista de inteiros l, onde $1 \leq l[i] \leq 10^4$.

Saída

Você deverá imprimir a lista R seguindo o padrão: separando cada um dos elementos por espaço e finalizando a saída com uma quebra de linha.

Exemplos de testes

Entrada	**Saída**
3 5 8 1 2 2 3 4 6 5 4 8 4 8 8 8 8	4 0 1 1 3 2 1 0 3 0 0 0 0

PROBLEMA I: PALAVRA SOLTEIRA

Arquivo: `solteira.[cpp/c/java/py]`

Cor: azul turquesa

Descrição do problema

O problema é descobrir em uma lista de palavras qual é aquela que não tem par, ou seja, aparece um número ímpar de vezes na lista. O tamanho T das palavras podem variar de 1 a 8. Atenção, ao resolver esse problema, cuidado com a quantidade de memória consumida pelo seu programa durante a execução.

Entrada

É composta de C ($1 \leq C \leq 10^3$) casos de teste indicados na primeira linha. Cada caso inicia com um número inteiro N ($1 \leq N \leq 10^5$) que indica o número total de palavras na lista e é seguido por N linhas contendo as palavras da lista, cada palavra tem no máximo 8 caracteres, alfanuméricos.

Saída

É composta pelas palavras solteiras de cada caso de teste, uma por linha.

Exemplos de testes

Entrada	**Saída**
3 1 S 5 XYZ ZYX YXZ ZYX XYZ 9 ABACATE ABACATE MARRECO MACHADO CADEIRA MACHADO MARRECO CADEIRA MACHADO	S YXZ MACHADO

PROBLEMA J: JOGO DA MEMÓRIA

Arquivo: `memoria.[cpp/c/java/py]`

Cor: vermelho quente

Descrição do problema

Em uma gincana de escola foi proposto um jogo da memória. Inicialmente, um dos professores falará os números que participarão da brincadeira, e, ao final, ele questionará um dos alunos e o mesmo deverá responder se o número é um dos números participantes ou não.

Um dos problemas da brincadeira é que os próprios professores acabavam esquecendo dos números, e isso causava confusões ou fazia com que cada rodada demorasse muito tempo. Sabendo disso, a diretora Andrea veio buscar a sua ajuda.

Escreva um programa que receba os números participantes, em seguida, receba o número questionado pelo professor e responda se o número está na brincadeira ou não. Os números da brincadeira serão informados em ordem crescente e não serão repetidos.

Entrada

A entrada começa com um número inteiro T (1 ≤ T ≤ 100), representando a quantidade de casos de teste. Em seguida, um inteiro N, onde 1 ≤ N ≤ 10^5, sinalizando a quantidade de números, em seguida, N inteiros, onde 1 ≤ I ≤ 10^8, e por fim, um número Q representando o número que foi questionado pelo professor, onde 1 ≤ Q ≤ 10^8.

Saída

Você deverá imprimir “Está na brincadeira.” para caso o número tenha sido dito pelo professor ou “Não está na brincadeira.” para caso o número não tenha sido dito pelo professor e finalizando a saída com uma quebra de linha.

Exemplos de testes

Entrada	**Saída**
2 6 0 1 3 5 9 12 9 6 0 1 3 5 9 12 2	Está na brincadeira. Não está na brincadeira.

PROBLEMA K: PRIORIDADE COM FATOR

Arquivo: `prioridade.[cpp/c/java/py]`

Cor: rosa shock

Descrição do problema

Em um sistema de processamento de tarefas, cada tarefa é representada por uma dupla de números inteiros: o tempo estimado de execução (em segundos) e a prioridade da tarefa (quanto maior, mais importante). Para determinar a ordem ideal de execução, define-se um fator de urgência, que é simplesmente o produto entre o tempo e a prioridade:

fator = tempo × prioridade

Tarefas com maior fator de urgência devem ser processadas primeiro. Em caso de empate no fator, a ordem original da entrada deve ser mantida (ou seja, estável).

Sua missão é processar uma lista de tarefas e ordená-las conforme o fator de urgência, da maior para a menor.

Entrada

É composta de C ($1 \leq C \leq 10^3$) casos de teste indicados na primeira linha. Cada caso inicia com um número inteiro N ($1 \leq N \leq 10^4$) que indica o número total de linhas a serem lidas. Cada linha contém dois inteiros T ($1 \leq T \leq 10^4$) e P ($1 \leq P \leq 10^4$) representando, respectivamente, o tempo de execução e a prioridade da tarefa.

Saída

Para cada caso de teste, imprima as duplas T P ordenadas de acordo com o fator de urgência (de forma decrescente). Em caso de empate no fator, mantenha a ordem em que as tarefas foram recebidas.

Exemplos de testes

Entrada	**Saída**
1 3 10 3 5 6 8 2	10 3 5 6 8 2