Desafio de Programadores – UFSCar 2014

03 de outubro de 2014

Este caderno de problemas contém 8 problemas em páginas numeradas de 1 a 15.
Verifique se o caderno está completo.

Apoio:

Informações Gerais

Exceto quando explicitamente indicado, as seguintes condições valem para todos os problemas:

Entrada

- A entrada deve ser lida da entrada padrão.
- A entrada pode ser composta por um ou mais casos de testes. Cada caso de teste é descrito usando um número de linhas que depende do problema.
- Quando uma linha de dados contém vários valores, estes são separados por um único espaço. Não aparecem outros espaços na entrada. Não existem linhas vazias.
- Cada linha, incluindo a última, possui um indicador de fim de linha.
- O fim da entrada é especificado no problema, podendo ser indicado por uma linha contendo valores específicos. Neste caso, esta linha não deve ser processada como um caso de teste.

Saída

- A saída deve ser escrita na saída padrão.
- O resultado de cada teste deve aparecer na saída usando um número de linhas que depende do problema.
- Quando uma linha de saída contém vários valores, estes devem ser separados por um único espaço. Não podem aparecer outros espaços na saída. Não devem existir linhas vazias.
- Cada linha, incluindo a última, deve possuir um indicador de fim de linha.
- Nenhum indicador especial deve ser escrito para indicar o fim da saída.
- O formato da saída deve seguir rigorosamente o padrão descrito no problema.

Problema A

Números Palíndromos

Nome do arquivo: palindromo.c ou palindromo.cpp

Limite de tempo: 1 segundo

Um número é **palíndromo** se ele possui o mesmo valor quando lido da esquerda para direita ou da direita para esquerda. Por exemplo, o número 12321_{10} é um palíndromo.

Obviamente, essa propriedade depende da base utilizada em sua representação. O número 27_{10} não é palíndromo na base 10, porém sua representação na base 2 (11011_2) é um palíndromo.

Alguns números possuem a propriedade de serem palíndromos em diferentes bases. Por exemplo, o número 313_{10} (100111001_2) é palíndromo nas bases 2 e 10.

Problema

Encontre a quantidade de inteiros positivos menores que N ($1 < N < 10^8$) que possuem a propriedade de serem palíndromos nas bases 10 e b ($1 < b < 10$). Por exemplo, se b = 2 e N = 100, temos 7 inteiros positivos que satisfazem a propriedade de serem palíndromos nas bases 2 e 10.

Representação base 2	Representação base 10
1_2	1_{10}
11_2	3_{10}
101_2	5_{10}
111_2	7_{10}
1001_2	9_{10}
100001_2	33_{10}
1100011_2	99_{10}

Entrada

A primeira linha de entrada contem dois inteiros positivos b e N.

Restrições

- $1 < N < 10^8$
- $1 < b < 10$

Saída

A saída consiste na quantidade de inteiros positivos menores que N que possuem a propriedade de serem palíndromos nas bases 10 e b.

Exemplos

Entrada:	**Saída:**
2 10	5

Entrada:	**Saída:**
2 100	7

Entrada:	**Saída:**
2 1000	10

Entrada:	**Saída:**
3 100	4

Entrada:	**Saída:**
5 1000	10

Entrada:	**Saída:**
9 1000	15

Problema B
Índice Carrasco Mamata

Nome do arquivo: mamata.c ou mamata.cpp

Os estudantes da Universidade Federal dos Senhores Carrascos (UFSCar) passam por uma situação muito estressante todo início de semestre. Esses alunos precisam se inscrever nas disciplinas que irão cursar e dependendo das escolhas feitas, o semestre pode se tornar excessivamente carrasco.

Para auxiliar os alunos nessa importante tarefa, os estudantes coletaram alguns dados sobre as disciplinas ofertadas nos anos anteriores e criaram o índice CM. O índice CM é um valor inteiro de 0 a 100, que classifica as ofertas de disciplinas como carrascas, quando próximas de zero, ou mamatas, quando próximas de cem. Esse índice CM foi calculado para todas as ofertas de todas as disciplinas e agora os alunos procuram se matricular prioritariamente nas disciplinas com alto valor de CM, ou seja, as disciplinas mais mamatas possível.

Conhecendo todas as ofertas de disciplinas que um aluno pode se matricular no semestre, sua tarefa é descobrir qual o maior valor de CM acumulado possível de se matricular no semestre, matriculando-se no maior número possível de disciplinas. O CM acumulado é obtido somando-se todos os CM das disciplinas que o aluno irá se matricular. Cada disciplina é oferecida em um horário fixo semanal.

Entrada

A primeira linha de entrada contém um número inteiro $\boldsymbol{D}$ ($1 \leq \boldsymbol{D} < 80.000$), indicando o número de ofertas de disciplinas. As $\boldsymbol{D}$ linhas seguintes contêm cada uma quatro campos: $\boldsymbol{S}$ ($\boldsymbol{S}$ = { seg | ter | qua | qui | sex | sab | dom}), $\boldsymbol{C}$ ($8 \leq \boldsymbol{C} \leq 22$), $\boldsymbol{F}$ ($9 \leq \boldsymbol{F} \leq 23$) e $\boldsymbol{I}$ ($0 \leq \boldsymbol{I} \leq 100$), onde $\boldsymbol{S}$ indica o dia da semana, $\boldsymbol{C}$ e $\boldsymbol{F}$ indicam a hora de início e a hora de término e $\boldsymbol{I}$ indica o índice CM da oferta da disciplina. O campos $\boldsymbol{C}$, $\boldsymbol{F}$ e $\boldsymbol{I}$ são números inteiros.

Saída

Seu programa deve produzir uma única linha, contendo o maior CM acumulado possível para se inscrever no maior número de disciplinas possível.

Exemplos

Entrada:	**Saída:**
5 seg 8 12 20 seg 8 12 80 ter 14 18 30 ter 16 18 60 qui 10 12 10	150

Entrada:	**Saída:**
7 ter 8 12 100 seg 20 22 77 ter 8 10 60 dom 8 12 30 seg 20 23 75 sex 19 22 50 ter 10 14 60	277

Problema C
Coffee Break

Nome do arquivo: coffee.c ou coffee.cpp

Certo ano o patrocínio conseguido pelos organizadores da Semana da Computação do DC-UFSCar foi tão polpudo que foi possível oferecer no coffee break quitutes caros. Havia, por exemplo, canapés de queijos e frios importados, salmão defumado, caviar, bolinho de bacalhau e chocolate suíço, além dos itens baratos como refrigerante, biscoitos, salgadinhos e sanduíches. Alguns alunos quiseram aproveitar ao máximo esse coffee break especial, comendo o máximo que couber no estômago e ainda maximizando o custo total dos alimentos ingeridos. No entanto, a pedido dos organizadores, os alunos devem respeitar algumas regras. Eles nunca devem pegar mais que uma porção de um mesmo tipo de comida e devem evitar o desperdício fracionando as porções oferecidas, se for o caso, para garantir que ninguém pegará mais do que pode comer.

Como os alunos cursam Computação, eles querem um programa que resolva o problema de acordo com o volume do estômago de cada um deles. Mas eles perderam muitas aulas recentemente e precisam da sua ajuda para escrever esse programa.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros N ($1 \leq N \leq 1.000$) e A ($1 \leq A \leq 100$), separados por um espaço em branco, que representam quantos tipos diferentes de comida são oferecidos e o número de alunos que deseja resolver o problema. A segunda linha contém N números reais c_i ($0 < c_i \leq 100$) que representam o custo (em reais) por porção da comida i. A terceira linha contém N números reais v_i ($0 < v_i \leq 10$) que representam o volume (em cm^3) de uma porção da comida i. As A linhas subsequentes contêm um número real cada, representando o volume e_i ($0 < e_i \leq 20$) do estômago do aluno i (em cm^3). O último caso de teste é seguido de uma linha contendo dois zeros.

Saída

Para cada aluno de cada caso de teste da entrada seu programa deve imprimir uma única linha, contendo um número real (com quatro casas decimais) que representa, dentre todas as possíveis combinações de frações de porções que o aluno consegue ingerir, o custo daquela de custo total máximo.

Exemplo

Entrada:	**Saída:**
6 4 15 3.2 7 20 1.5 4 1.1 2 5.3 3 4 2 2.5 5 7.1 10 0 0	24.3333 36.8000 40.6000 44.7094

Problema D
Livros da Biblioteca

Nome do arquivo: biblioteca.c ou biblioteca.cpp

A Biblioteca Controlada (BCo), principal biblioteca de uma universidade muito conhecida, decidiu alterar as regras de empréstimos dos seus livros. Como ultimamente muitos livros não estão sendo devolvidos, a direção proibiu a entrada dos estudantes na biblioteca e limitou o empréstimo diário para apenas um livro por estudante.

Agora, para retirar um livro, cada aluno precisa fornecer uma lista de, no máximo, 5 livros até o meio dia. No final da tarde a biblioteca divulgará qual será o livro que cada aluno poderá retirar, dentre os livros que forneceu na sua lista.

Para minimizar o *mimimi*, ou seja, as reclamações dos estudantes, a biblioteca solicitou que a lista de livros seja ordenada pela prioridade desejada pelo aluno. Desta forma, o primeiro livro da lista teria prioridade 5, o segundo livro teria prioridade 4, e assim por diante. O último livro teria a prioridade mínima, ou seja, 1.

Com essa ordem, a biblioteca irá buscar atender o maior número possível de alunos e respeitar ao máximo os livros das primeiras posições de cada lista. Por exemplo, seria ideal permitir que todo aluno retire sempre o primeiro livro da sua lista, mas nem sempre isso será possível.

Como você conseguiu uma bolsa trabalho na biblioteca justamente quando adotaram esse novo controle, você ficou encarregado de fazer um programa para auxiliar a biblioteca nessa polêmica restrição de empréstimo.

Seu trabalho é encontrar o melhor cenário possível de empréstimos, ou seja, emprestar o maior número de livros que tenham a maior prioridade possível. Informe ao seu supervisor qual será a maior soma de prioridades possível, atribuindo um livro por estudante e atendendo ao maior número de alunos.

Entrada

A primeira linha contém um número inteiro N_A ($1 \leq N_A \leq 200$), correspondendo ao número de alunos que desejam emprestar um livro naquele dia. Cada uma das N_A linhas seguintes descreve a lista ordenada de livros de interesse de um aluno. A linha é formada pelo valor N_L ($1 \leq N_L \leq 5$), que representa o número de livros da lista, seguido pelos valores L_1, L_2, …, L_{NL} ($0 \leq L_i \leq 1.000.000$), correspondendo ao identificador de cada livro na biblioteca.

Saída

A saída corresponde a maior soma possível das prioridades de todos os livros emprestados no dia.

Exemplos

Entrada:	**Saída:**
3 2 0 1 3 3 2 0 2 3 0	14

Entrada:	**Saída:**
5 3 71 55 32 2 32 4 4 55 32 21 71 1 71 2 21 32	22

Problema E

IsaSeq

Nome do arquivo: isaseq.c ou isaseq.cpp

Isabela tem 7 anos e gosta muito de brincar com um joguinho que tem muitas letras e números. Ela é muito inteligente e gosta de pedir aos adultos para que criem desafios para ela resolver. Certo dia, ela estava organizando seu brinquedo e começou a construir várias sequências intercalando letras e números e sempre iniciando e terminando com letra. Essas sequências foram chamadas de **SeqInicial** e podemos considerar exemplos válidos as sequências: A2S5E2B5S9H4K5W, A2B, U5E2J5D6G5K4C, X1X1X. Isabela então criou uma regra para criar sua sequência **IsaSeq**, de maneira que é possível transformar qualquer **SeqInicial** em **IsaSeq** e vice-versa. A **IsaSeq** sempre começa com uma letra e termina com um número.

Isabela usou o seguinte raciocínio para a montagem da sequência **IsaSeq**. Procure a primeira subsequência [letra1 número letra2] cujo número é a primeira ocorrência do maior dígito presente na sequência (da esquerda para a direita). Em seguida, transforme-a em **IsaSeq**, isto é [letra1 letra2 número] e substitua essa sequência por uma letra especial. Repita esse processo até que sobre apenas uma letra especial. Então substitua todas as letras especiais pelas **IsaSeq**. Veja o exemplo a seguir:

SeqInicial: A1B2C2D3E

A1B2C2D3E

D3E => DE3 => X

A1B2C2X

B2C => BC2 => Y

A1Y2X

Y2X => YX2 => Z

A1Z

A1Z => AZ1 => W

W

Para construir a **IsaSeq**:

W = AZ1

=> AZ1

Z = YX2

=> AYX21

Y = BC2

X = DE3

=> ABC2DE321

Ela quer mostrar para você que ela consegue transformar qualquer **IsaSeq** em **SeqInicial**. Para isso, você deve conseguir transformar uma **SeqInicial** em **IsaSeq**.

Entrada

A entrada contém vários casos de teste. Cada um deles é dado em exatamente uma linha contendo um conjunto não vazio da sequência **SeqInicial** (letra número letra número ... letra). O tamanho máximo de **SeqInicial** é 10^4 caracteres. Após o último caso de teste segue uma linha contendo um zero.

Saída

Para cada caso de teste, a saída será uma linha que representa a **IsaSeq** da **SeqInicial** de entrada.

Exemplos

Entrada:	**Saída:**
A1B2C2D3E A1B3C3D4E3F Y0V1I 0	ABC2DE321 ABC3DE43F31 YVI10

Problema F

Festa na Futebolândia

Nome do arquivo: festa.c, festa.cpp

Limite de tempo: 1 segundo

O futebol é o esporte nacional na república da Futebolândia, um longínquo país da América do Sul. Neste país existem dois times de futebol – *Real Soccer* e *Soccer United* – e seus habitantes são torcedores fanáticos desses times.

Noraldo Zanário, um ilustre habitante desse país, decidiu organizar uma festa para comemorar a sua recente nomeação ao cargo de presidente da OMF (Organização Mundial de Futebol). Porém preparar organizar a lista de convidados não será uma tarefa fácil. É notório que, nesse longínquo país, o encontro entre torcedores rivais costuma ser bastante problemático.

Neste problema você terá que decidir qual habitante convidar e qual habitante não convidar, a fim de evitar o encontro entre torcedores rivais.

Problema

Temos uma lista de N pessoas, habitantes desse país, que podem ser convidadas para a festa ou não. Para cada pessoa *i*, temos uma lista de seus rivais: R_1, R_2, ..., R_p. A relação "rival"[1] tem as seguintes propriedades:

- **Anti-transitiva**. Se A é "rival" de B, e B é "rival" de C, então A é "aliado" de C. Além disso, os "rivais" dos "aliados" de A são seus "rivais" e os "aliados" dos "aliados" de A são seus "aliados".

- **Simétrico**. Se A é "rival" de B, então B é "rival" de A (embora possa não está indicada em sua lista de rivais).

Uma pessoa aceitará seu convite para a festa, se, e somente se, ele for convidada, todos os seus "aliados" forem convidados e nenhum de seus "rivais" for convidado. Você tem que encontrar o número máximo de pessoas que podem ser convidadas, para que todos aceitem o convite.

Por exemplo, se N = 5, e é notório que: A é "rival" de C, B é "rival" de A e D é "rival" de E, então poderíamos convidar um máximo de 3 pessoas. Estes poderiam ser B, C e D, mas para este problema apenas é necessário determinar o número de pessoas convidadas.

1 A relação "rival" indica que A e B torcem para diferentes times.

Entrada

A primeira linha do caso de teste contém um inteiro N, indicando o número de pessoas que precisam ser consideradas. Você pode considerar que N ≤ 500. Para cada uma destas N pessoas, existe uma lista contendo a sua lista de rivais.

A primeira linha contém a lista de "rivais" da pessoa 1, a segunda linha contém a lista de "rivais" da pessoa 2, e assim por diante. Cada lista de "rivais" começa com um número inteiro *r* (o número de "rivais" conhecidos daquela pessoa) e, em seguida, existem *r* inteiros (os *r* "rivais" dessa pessoa). Assim, por exemplo, se os rivais de uma pessoa são 4 e 6, a sua lista de rivais seria: 2 4 6.

Restrições

- N ≤ 500 e 1 ≤ r ≤ N

Saída

A saída deve consistir de uma única linha contendo um número inteiro, o número máximo de pessoas que podem ser convidadas, para que todos eles aceitem o convite.

Exemplos

Entrada:	**Saída:**
5 1 3 1 1 0 1 5 0	3

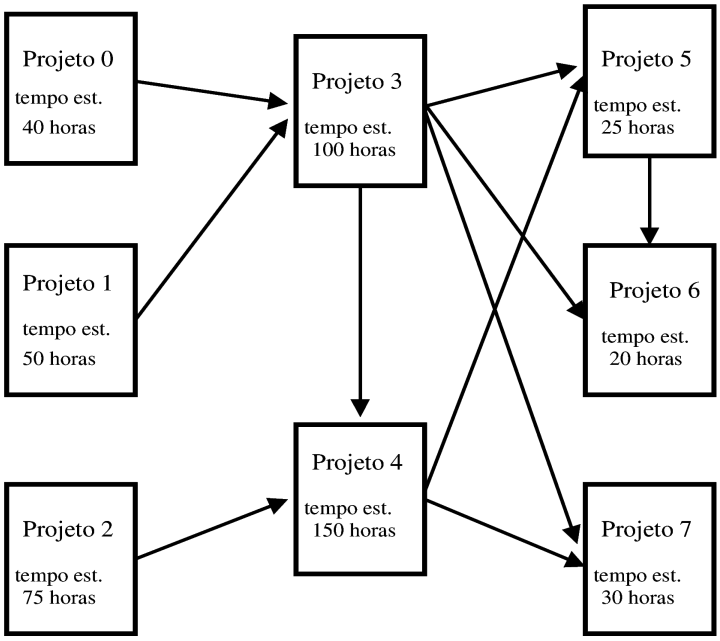
Entrada:	**Saída:**
8 2 4 5 2 1 3 0 0 0 1 3 0 1 5	5

Problema G
Gerente de Projetos

Nome do arquivo: gerente.c ou gerente.cpp

Um gerente de projetos acabou de receber uma excelente notícia de sua chefe: assim que entregar todos os projetos sob sua responsabilidade no momento será promovido e seu salário dobrará! Imediatamente o gerente de projetos ficou ansioso para saber quando será a promoção e pediu (para ontem) uma estimativa, de cada equipe de cada projeto, do número de horas necessárias para concluir seu projeto. Suas equipes responderam rapidamente com suas estimativas, no entanto, lembraram que existem algumas relações de dependência entre os projetos gerenciados por ele, de forma que cada equipe só consegue começar quando todos os projetos de que dependem estiverem concluídos.

A figura abaixo mostra um exemplo de um conjunto de projetos com os respectivos tempos estimados para conclusão e os projetos dos quais dependem. No exemplo da figura o tempo mínimo para a conclusão completa dos projetos (e para o gerente conseguir a promoção) é de 345 horas, dado pela soma dos tempos de conclusão dos projetos 1, 3, 4, 5 e 6 que dependem sequencialmente um do outro, nessa ordem. Qualquer outra sequência de dependências leva a um tempo total de conclusão menor.



Ao perceber que determinar o tempo mínimo até a promoção é mais complicado do que imaginou inicialmente, o gerente de projetos foi lamentar com seus outros colegas gerentes sua ansiedade, e descobriu que todos eles estavam na mesma situação. Eles resolveram então se juntar e contratar você para determinar esse tempo mínimo para cada um deles.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros N e M, separados por um espaço em branco, que representam, respectivamente, quantos são os projetos ($1 \leq N \leq 10.000$) e quantas são as relações são as relações de dependência ($1 \leq M \leq 100.000$). Cada projeto é representado por um inteiro entre 0 e N-1. A segunda linha de cada caso de teste contém N inteiros t_i ($1 \leq t_i \leq 10.000$) que representam os tempos estimados de conclusão de cada projeto. Cada uma das M linhas subsequentes de cada caso de teste contém dois inteiros u e v ($0 \leq u, v \leq N-1$), separados por um espaço em branco, indicando que o projeto v depende do projeto u para poder ser concluído. O final da entrada é indicado por N = M = 0.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha contendo um único inteiro que é o tempo mínimo para o gerente concluir seus projetos atuais e obter a sua promoção.

Exemplos

Entrada:	**Saída:**
8 10 40 50 75 100 150 25 20 30 0 3 1 3 2 4 3 4 3 5 3 6 3 7 4 5 4 7 5 6 5 4 10 20 30 40 50 0 1 1 2 3 1 1 4 6 8 20 70 60 30 100 10 0 1 0 3 1 2 1 4 2 5 3 4 4 2 4 5 1 0 9384 0 0	345 110 260 9384

Problema H
Eleições na Romanolândia

Nome do arquivo: romano.c, romano.cpp

Limite de tempo: 1 segundo

As eleições na Romanolândia se aproximam e o TSE (Tribunal Superior Eleitoral) desse país determinou que:

- Cada número de candidato é composto por 4 digitos decimais na faixa [1000 - 3999], em que os dois primeiros digitos [10 - 39] representam o número do partido do candidato.
- O custo da impressão dos candidatos nas cédulas eleitorais deve ser ressarcido por cada partido. Cada partido pode ter no máximo 100 candidatos e o número de seus respectivos candidatos devem ser sequenciais em uma faixa de valores [PPAA - PPBB] em que:
    - PP representa o número do partido
    - AA representa o sufixo do primeiro candidato do partido
    - BB representa o sufixo do último candidato do partido
- Os números dos candidatos devem ser convertidos e impressos, nas cédulas eleitorais, em algarismos romanos 'I', 'V', 'X', 'L', 'C', 'D' e 'M' que representam os valores decimais 1, 5, 10, 50, 100, 500 e 1000 respectivamente.

Problema

Você é o gerente da gráfica oficial, que imprimirá as cédulas eleitorais, e é responsável por calcular o valor de ressarcimento (de cada partido) do custo de impressão de uma célula eleitoral.

Considerando que cada algarismo romano possui um custo de impressão, você deve escrever um programa que, dados o número do primeiro e último candidato do partido, descubra o valor de ressarcimento (de cada partido) do custo de impressão de uma cédula eleitoral.

Entrada

A primeira linha de entrada contem o inteiro positivo N de casos de teste. A segunda linha contém os inteiros I, V, X, L, C, D, M correspondendo respectivamente aos custos de impressão dos algarismos romanos 'I', 'V', 'X', 'L', 'C', 'D' e 'M'.

As demais N linhas representam os casos de teste. Cada caso de teste contem dois inteiros PPAA e PPBB (de 4 digitos) que representam a faixa de valores dos números de candidatos de um partido.

Restrições

- $0 < N \leq 10000$
- $0 < I, V, X, L, C, D, M \leq 10$
- $10 \leq PP \leq 39$
- $0 \leq AA \leq BB \leq 99$

Saída

Para caso de teste, a saída consiste no valor de ressarcimento (de cada partido) do custo de impressão de uma cédula eleitoral.

Exemplos

Entrada:	**Saída:**
5 1 1 1 1 1 1 1 1300 1300 1500 1599 2110 2121 2520 2555 3989 3999	4 600 71 253 101

Entrada:	**Saída:**
5 1 2 3 4 5 6 7 1300 1300 1500 1599 2110 2121 2520 2555 3989 3999	22 2240 298 1056 487

Entrada:	**Saída:**
5 7 6 5 4 3 2 1 1300 1300 1500 1599 2110 2121 2520 2555 3989 3999	10 2560 270 968 321