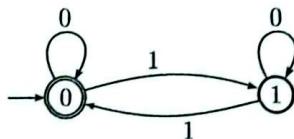


**Problema A**  
**Autômato Automático**  
Arquivo: automato.[c|cpp|java|py]  
Limite de tempo: 1 s

**O Problema:**

Você se entusiasmou pela disciplina de Teoria da Computação e quis fazer um programa para simular autômatos finitos determinísticos (AFD) e verificar se eles aceitam as cadeias de entrada (os não determinísticos você vai deixar para a próxima ProgBASE). Veja o exemplo:



Este autômato aceita todas cadeias com um número par de 1s. Inclusive a cadeia vazia.

**Entrada:**

Cada caso de entrada possui na primeira linha três valores inteiros:  $Q$  o número de estados, ( $0 < Q \leq 20$ ), os estados são numerados de 0 a  $Q - 1$ , sendo 0 sempre o estado inicial.  $S$  o número de símbolos do alfabeto ( $0 < S \leq 10$ ) e  $W$  o número de consultas de cadeias de entrada. ( $0 < W \leq 50$ ). A segunda linha contém uma cadeia de tamanho  $S$  com os símbolos do alfabeto, cada símbolo é representado por um caractere ASCII padrão imprimível alfanumérico. O símbolo '\*' será usado para representar uma cadeia de entrada vazia. A seguir são  $S$  linhas com  $Q$  números espaçados em branco, a  $s_i$ -ésima linha representa as transições do  $s_i$ -ésimo símbolo do alfabeto da cadeia de símbolos, para cada estado. Em seguida são  $Q$  valores 0 e 1 indicando com 0, os estados que não são de aceitação e 1 os que são de aceitação. Por fim seguem-se  $W$  linhas onde cada linha contém uma cadeia com no máximo 100 símbolos como cadeia de entrada, em especial a cadeia "\*" representará uma cadeia vazia, sem qualquer símbolo.

**Saida:**

Espera-se na saída  $W$  linhas com as palavras: "Aceita" ou "Rejeita", caso o AFD aceite ou rejeite tal cadeia.

**Exemplo de entrada:**

```
2 2 3
01
0 1
1 0
1 0
01100010
100011101
*
```

```
5 2 10
ab
1 1 1 4 4
3 2 2 3 3
0 1 0 1 0
aa
ab
bb
ba
aabbaa
bababa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaa
b
*
```

**Saída para o Exemplo de entrada:**

```
Aceita
Rejeita
Aceita
```

```
Aceita
Rejeita
Aceita
Rejeita
Aceita
Rejeita
Aceita
Rejeita
Aceita
Rejeita
```

Observe que o primeiro caso de teste representa o autômato do exemplo no texto

**Problema B**  
**Boi-Bumbá**  
Arquivo: *boi.[c|cpp|java|py]*  
Limite de tempo: 1 s

**O Problema:**

O Boi Bumbá é um folguedo folclórico praticado em todo Brasil, mas com ênfase na região Norte-Nordeste. Também conhecido como Bumba-meu-boi, Boi-Calemba, Boi-de-reis, Boi-zumbi entre outros. A história na sua forma mais simples vem da escrava Catarina que grávida queria comer língua do boi mais bonito, pediu ao marido Chico (também conhecido como Pai Francisco ou Mateus) que matou o boi. O casal foi perseguido pelo fazendeiro, e no final com a ajuda de curandeiros o Boi foi ressuscitado, então o fazendeiro fez uma festa. O folclore se baseia na importância do boi nas atividades da época. O Maranhão é onde mais se comemora esta festa, mas seu ápice acontece em Parintins-PA onde a festa envolve a disputa de dois Bois Bumbá: Caprichoso e Garantido representados pelas cores azul e vermelho respectivamente. O prefeito da cidade de São Bento do Capão (SBC) vendo o Boi-Bumbá como uma festa alternativa às festas-juninas que já têm dominância nas cidades vizinhas quer trazer o folguedo na forma da disputa de Parintins, neste caso já pediu às agremiações carnavalescas para passarem a representar algum Boi-Bumbá em disputa na festa. Só que ainda precisa decidir a forma de premiação. Serão computados pontos (inteiro ou meio) de 0 a 10 em várias alegorias, como: Fantasia, Bateria, Enredo, entre outras e cada alegoria tem seu peso. No final ganha quem tiver mais pontos. Mas se houver empate, o critério de desempate segue por aquele que tiver mais pontos em determinada alegoria por ordem. Se tudo seguir empatado então será por ordem de apresentação. A você foi pedido um algoritmo para definir a ordem final de classificação dos bois.

**Entrada:**

A entrada começa com 2 números inteiros em uma linha:  $A$  ( $0 < A \leq 10$ ) e  $B$  ( $1 < B \leq 10$ ), representando respectivamente as alegorias e os bois. Na segunda linha são 10 valores inteiros representando o peso de cada alegoria na nota, os valores estão na ordem de desempate por nota mais alta nas alegorias, da primeira à última. Em sequida são  $B$  linhas onde a  $b_i$ -ésima linha representa o  $b_i$ -ésimo boi, na ordem das apresentações, e cada linha contém, uma palavra  $N$  representando o nome do boi com no máximo 10 caracteres (ASCII padrão sem acentuação) sendo a primeira maiúscula, e  $A$  valores inteiros ou em frações de meio  $a_j$  ( $0 \leq a_j \leq 10$ ) que são as notas na ordem das alegorias.

**Saida:**

Como saída, deve ser impressa  $B$  linhas com os nomes dos bois em ordem de classificação na competição

**Exemplo de entrada:**

```
3 3
3 5 1
Mimoso 2.5 1.5 3.0
Gracioso 4.0 1.0 1.0
Furioso 3.0 2.0 1.0
```

```
2 2
5 5
Saudoso 10.0 10.0
Medroso 10.0 10.0
```

**Saída para o Exemplo de entrada:**

```
Furioso
Gracioso
Mimoso
```

```
Saudoso
Medroso
```

**Problema D**  
**Dois Dados (Jogo do Porquinho)**  
Arquivo: *dados.[c|cpp|java|py]*  
Limite de tempo: 1 s

**O Problema:**

Trata-se de um jogo de dados, que se joga com dois dados, que se lançam sucessivamente, multiplicando-se sempre os números de cada lance um pelo outro, até se atingir a pontuação de 300. Quem atingir o teto dos 300 pontos, com o menor número de lances ganha. Neste jogo, os dobles (duas faces iguais) são sempre duplicados, ou seja, além da multiplicação de um número pelo outro, o resultado ainda é dobrado. Com exceção do doble de 1, que é contado como perfazendo 30 pontos. Nesta versão do jogo cada jogador pode lançar os dois dados 5 vezes, quem obtiver mais pontos ganha.

**Entrada:**

A entrada começa com um número  $N$  ( $0 < N \leq 10$ ) o número de jogadores, numerados de 1 a  $N$ . Em seguida há  $N$  linhas onde em cada linha está o  $i$ -ésimo jogador. Em cada linha há 5 conjunto de números:  $(a, b)$ , separados por um espaço em branco, onde  $1 \leq a, b \leq 6$  que são as faces dos dados de cada uma das 5 jogadas.

**Saida**

Na saída deve sair impresso na primeira linha: "Pontos do ganhador: " seguido dos pontos obtidos pelo campeão do jogo. Na segunda linha a frase: "Ganhador(es): ", e após um espaço, os ganhadores (os jogadores que atingiram a pontuação máxima), se houver mais de 1, separados por vírgula e espaço após a vírgula, em ordem crescente do número do jogador.

**Exemplo de entrada:**

```
2
(1,2) (2,3) (3,4) (4,5) (5,6)
(2,2) (3,3) (4,4) (3,3) (2,2)
```

```
2
(6,5) (2,5) (1,4) (5,2) (5,6)
(2,2) (3,3) (4,4) (3,3) (2,2)
```

**Saída para o Exemplo de entrada:**

```
Pontos do ganhador: 84
Ganhador(es): 2
```

```
Pontos do ganhador: 84
Ganhador(es): 1, 2
```

## Problema C

### Casal Certo

Arquivo: casal.[c|cpp|java|py]

Límite de tempo: 1 s

**O Problema:** Carolzinha é uma cachopa muito crente consorte de Castor. O casório carece de certificado de contra-conexão. Ou seja, Carol e Castor não podem ter parentesco de grau algum. Castor contratou você para construir um código que certificasse este grau de parentesco. Então dada uma linhagem de pais e filhos, você precisa indicar se entre Carol e Castor há algum grau de parentesco.

#### Entrada:

Cada caso de teste possui várias linhas, a primeira linha contém 4 números inteiros  $P$  (pessoas)  $2 \leq P < 50$ , representando quantas pessoas existem nas árvores genealógicas de Carol e Castor.  $L$  (linhagem)  $0 \leq L < P$ , representando o número de ligações direta de pai-prole entre duas pessoas das árvores.  $A$  e  $B$   $1 \leq A \neq B \leq P$  representando a posição de Carolzinha:  $A$  e Castor:  $B$  na árvore. Em seguida são  $L$  linhas com dois valores  $G$  e  $J$  quem é o(a) Genitor(a) ( $G$ ) diretamente e quem é Júnior ( $J$ ),  $1 \leq G \neq J \leq P$ . Os casos de entrada terminam com  $P = L = A = B = 0$  que não deve ser considerado. É importante ressaltar que a relação de parentesco nunca é cíclica em nenhum caso de teste. Também na ascendência direta é somente considerado o genitor que poderia levar a algum grau de parentesco, ou seja, cada “filho” possui somente um “pai”.

#### Saída:

A saída pode ser uma de três possíveis: “Primos de grau  $n$ ”, quando Carolzinha e Castor tiverem um ancestral comum e a maior distância ao ancestral é  $n$ . “Ascendentes/Descendentes de grau  $n$ ”, quando Carolzinha e Castor têm uma relação direta de ascendência/descendência de grau  $n$ , se fosse uma descendência direta, quando um é genitor(a) e o(a) outro(a) prole,  $n$  é 1. Ou “Casal Certo” se não houver relação de parentesco.

#### Exemplo de entrada:

```
10 9 1 5
4 1
4 3
5 2
7 6
9 8
9 7
6 4
9 5
10 9
5 4 2 4
5 4
5 1
4 3
3 2
4 2 1 3
4 1
3 2
0 0 0 0
```

#### Saída para o Exemplo de entrada:

```
Primos de grau 4
Ascendente/Descendente de grau 2
Casal Certo
```

**Problema E**  
**Estradas Enlameadas**  
Arquivo: estrada.[c|cpp|java|py]  
Limite de tempo: 1 s

**O Problema:**

É muito agradável viajar pelo nosso país, e também retornar a pontos já visitados. Retornando nós conseguimos ver o que o tempo (ou o progresso) fez com o local. Por exemplo, Jericoacoara - CE era uma canto agradável, lá tinha a barraca do por-do-sol (único lugar do Brasil onde o sol se põe no mar). Agora a praia virou um esgoto em detrimento à construção de hotéis. Por outro lado, Ilha de Boipeba - BA cresceu com qualidade ruas que eram de barro estão calçadas mas a Ilha continua com seu charme. O benefícios nos locais acontecem ao longo do tempo, pois há custo. Sabendo disso o prefeito da cidade onde fica a Vila de São José do Piriá, à beira do rio Piriá no MA quer transformar aquela Vila num ponto turístico. No entanto todas as ruas são de terra e na época da chuva vira um lamaçal terrível. O prefeito mapeou todas as ruas e definiu o custo de calçar a rua em relação ao retorno que esta rua trará se aumentar seu movimento. Por exemplo se uma rua é cheia de comércio e tem o custo 10, mas aumentando a circulação traz uma arrecadação de 5, então o custo de fato será 5. Por outro lado, uma rua com o custo de 8 que não traz nenhuma arrecadação tem o custo 8, de fato. O mapa é definido por pontos de cruzamentos e as ruas são trechos que ligam 2 cruzamentos. O prefeito quer que todos os cruzamentos sejam acessíveis com calçamento, mas quer evitar rotas redundantes, ou seja, entre dois cruzamentos distintos só pode haver uma rota calçada. Ele pediu sua ajuda para escolher as ruas a serem calçadas.

**Entrada:**

A entrada contém vários casos de teste (ele já está pensando em outras vilas) cada caso de teste começa com dois números inteiros  $C$ , ( $0 < C \leq 100$ ) que são os cruzamentos, numerados de 1 a  $C$ , e  $R$ , ( $C \leq R < 3 * C^2$ ), a quantidade de ruas. Em seguida  $R$  linhas onde a  $i$ -ésima linha representa a  $i$ -ésima rua e traz 2 valores inteiros  $C_1, C_2$  ( $0 < C_1 \neq C_2 \leq C$ ) que representam os dois cruzamentos ligados pela rua e 2 valores de ponto flutuante,  $P$  e  $B$ , onde  $P$  ( $0 < P \leq 500.0$ ) é o preço a ser pago para calçar a rua e  $B$  o benefício esperado pelo calçamento ( $0 \leq B \leq 500$ ). Os casos de teste terminam com o arquivo de entrada.

**Saida:**

Para cada caso de teste o prefeito deseja que você imprima em uma linha única, com duas casas decimais, qual o custo a ser pago para o calçamento das ruas escolhidas.

**Exemplo de Entrada**

```
12 17
1 2 327.50 89.95
2 3 327.50 17.40
3 4 327.50 300.43
5 6 327.50 18.55
6 7 327.50 304.82
7 8 327.50 406.18
9 10 327.50 340.23
10 11 327.50 19.33
11 12 327.50 200.33
1 5 250.40 300.27
2 6 250.40 302.80
3 7 250.40 139.88
4 8 250.40 1.34
5 9 250.40 183.33
6 10 250.40 5.38
7 11 250.40 158.83
8 12 250.40 233.00
3 3
1 2 3 0
2 3 2 0
3 1 2 0
```

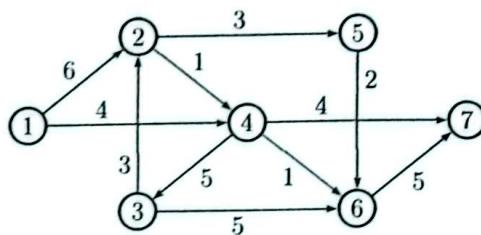
**Saída para o Exemplo de Entrada**

```
3139.90
4.00
```

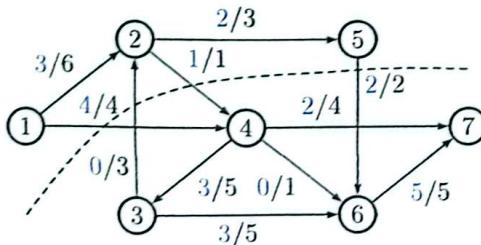
**Problema F**  
**Ford-Fulkerson**  
*Arquivo: f.[c|cpp|java|py]*  
 Limite de tempo: 1 s

**O Problema:**

Ford-Fulkerson é uma fábrica de vitaminas em cápsulas. No processo de produção existem várias máquinas que processam determinados minerais, ou sais ou outros produtos necessários em vitaminas. O processo é automático, existem várias esteiras ligando as diversas máquinas desde uma origem, onde começa o processo até o término onde são entregues as vitaminas concluídas. Porém dada unidade tem sua velocidade de produção e as esteiras entregam na velocidade máxima de cada máquina, para cada tipo de vitamina. No entanto, dadas as interligações algumas esteira trabalham na velocidade máxima e outras trabalham abaixo deste limite. No exemplo abaixo, temos um exemplo desta "rede" e as velocidades ou "fluxos" máximo em cada esteira.



Abaixo temos uma distribuição de velocidade máxima que se consegue na rede acima. Observe que exceto a origem e destino, cada máquina tem um saldo de fluxo nulo, ou seja o fluxo de entrada de vitaminas na máquina é igual ao fluxo de saída, assim não há consumo ou produção de fluxo pelas máquinas intermediárias. A origem só produz fluxo e o destino só consome fluxo. Também podemos ver que temos uma linha pontilhada indicando o corte mínimo desta rede, o conjunto de arcos que saturam no fluxo máximo.



Edmonds Karp o CEO da Ford-Fulkerson veio a você pedir que a partir das configurações de máquinas e esteiras, faça um algoritmo para indicar o fluxo máximo de saída das vitaminas produzidas.

**Entrada:**

São vários casos de teste, cada caso começa, na entrada, com dois valores inteiros:  $N$  e  $M$ , indicando o número de máquinas e o número de esteiras, sendo  $0 < N \leq 20$  e  $N - 1 \leq M \leq N(N - 1)/2$ . Cada máquina é numerada de 1 a  $N$ , onde a máquina 1 é a origem onde se inicia o processo de produção, e a máquina  $N$  o destino, onde se conclui a produção e entrega-se as vitaminas. A seguir são  $M$  linhas onde cada linha são três valores inteiros:  $A \ B \ F$ ,  $A$  é a máquina onde a esteira começa (recebe o fluxo),  $B$  a máquina onde a esteira termina (entrega o fluxo),  $1 \leq A \neq B \leq N$  e  $F$  o valor máximo de fluxo da esteira  $1 \leq F \leq 10$ . Os casos de teste terminam com o fim de entrada.

**Saida**

Na saída deve ser impresso o fluxo máximo de produção de vitaminas pelo complexo.

**Exemplo de entrada:**

```

7 11
1 2 6
1 4 4
2 4 1
2 5 3
3 2 3
3 6 5
4 3 5
4 6 1
  
```

7 4  
6 2  
6 7 5  
10 18  
1 2 9  
1 3 4  
2 4 6  
2 5 3  
3 4 1  
3 6 5  
4 5 7  
4 6 3  
5 7 3  
6 5 1  
6 7 5  
7 4 4  
7 8 2  
7 9 7  
8 5 5  
8 10 8  
9 6 3  
9 10 5

Saída para o Exemplo de entrada:

7  
7

## Problema G

### Gato Gaiato

Arquivo: gato.[c|cpp|java|py]

Límite de tempo: 1 s

#### O Problema:

Guilhermina gosta do gato Gatito. Mas ela fala que Gatito é Gaiato quando ele sai de casa. Guilhermina colocou um chip no Gatito para obter sua localização, e de vez em vez ela olha no aplicativo as coordenadas da posição de Gatito. Só que ela acha que o aplicativo poderia ser melhorado, pois ela precisa relacionar a localização do gato com a própria casa, ou casa de campo, ou casa da praia, ou casa da avó, ... seja lá onde ela estiver com o Gatito. Ela pediu para você construir um algoritmo para adicionar ao aplicativo dela informando se Gatito é "Gato" ou "Gaiato". No mundo de Guilhermina, a casa sempre é redonda.

#### Entrada:

Cada caso de teste constituído de duas linhas, a primeira linha contém três números inteiros  $H$ ,  $V$  e  $R$  ( $-50 \leq H, V \leq 50$  e  $0 < R \leq 50$ ) representando a localização leste-oeste ( $H$ ) e sul-norte ( $V$ ) da casa que Guilhermina e Gatito estão, e o raio ( $R$ ) da casa. Na segunda linha são dois números inteiros  $Gh$  e  $Gv$  ( $-100 \leq Gh, Gv \leq 100$ ), a posição de Gatito com  $Gh$  a coordenada leste-oeste e  $Gv$  a coordenada sul-norte.

#### Saída:

Dada a posição do gato, se ele estiver no exterior do imóvel deve ser impresso a palavra "Gaiato" caso contrário, "Gato".

#### Exemplo de entrada:

```
1 3 2  
3 1
```

```
0 1 3  
-2 1
```

#### Saída para o Exemplo de entrada:

Gaiato

Gato

**Problema H**  
**Herói Homônimo**  
Arquivo: *heroi.[c|cpp|java|py]*  
Limite de tempo: 1 s

**O Problema:**

O quadrinista Stan Lee da Marvel, adora dar nomes aliterativos para seus personagens, tudo começou com o quarteto fantástico, com Reed Richards e Susan Storm. No mundo do homem aranha ele se superou, temos: Peter Parker, Betty Brant, Otto Octavius, J. Jonah Jameson, como exemplos. Existem outros, como no X-men: Scott Summers, Warren Worthington III e Sebastian Shaw. O mais curioso é que ele teve esta ideia dos personagens do Superman da DC comics: Louis Lane, Lana Lang, Lori Lemaris e Lyla Lerrol. O próprio nome do Superman: Clark Kent poderia ser considerado aliterativo, pois tem o mesmo som na primeira letra de cada nome. A você foi pedido para que, dada uma lista de nomes, indique se estes são aliterativos ou não.

**Entrada:**

Cada caso de teste é um nome escrito com letras sem acentuação, padrão ASCII, onde a primeira letra de cada palavra que contém o nome é maiúscula. O nome contém no máximo 100 letras. Um nome somente é aliterativo se a primeira letra de cada palavra que contém o nome é a mesma.

**Saida**

Se o nome for aliterativo deve-se imprimir na saída: "Viva!", caso contrário, "Bah!".

**Exemplo de entrada:**

Curt Connor

Mary Jane Watson

Stephen Strange

Norman Osborn

**Saída para o Exemplo de entrada:**

Viva!

Bah!

Viva!

Bah!

**Problema I**  
**Ilhas Isoladas II**  
Arquivo: *ilhas.[c|cpp|java|py]*  
Limite de tempo: 1 s

**O Problema:**

O município de Cairu na Bahia envolve algumas Ilhas, entre elas de Tinharé e de Boipeba. São vários pontos turísticos a visitar, e havia briga nas agências se os transportes seriam terrestres ou marítimos. Foi estabelecido um acordo, se dois locais possuem ligação por terra, então será por terra, senão, será por água. Com isto o prefeito de Cairu pediu quais os pontos que as agências exploram o turismo e traçou as ligações por terra que existem entre os pontos. Os pontos são os seguintes:

- |                        |                           |
|------------------------|---------------------------|
| 1 - Cairu              | 7 - Cueira                |
| 2 - Gamboa             | 8 - Moreré                |
| 3 - Morro de São Paulo | 9 - Ponta dos Castelhanos |
| 4 - Quarta Praia       | 10 - Cova do Onça         |
| 5 - Pratigi            | 11 - Canavieiras          |
| 6 - Velha Boipeba      | 12 - Torrinhas            |

Ficou definido chamar de rota terrestre uma rota que liga 2 pontos e permite a passagem de veículo, no exemplo: 2-3, vai de Gamboa a Morro de São Paulo. Apesar de existir caminho por terra, este envolve escarpas e escadas estreitas que não permite a passagem de veículos, então a rota é marítima.

A você foi pedido que se fizesse um algoritmo para dados dois pontos definir se o transporte será por terra ou por barco.

**Entrada:**

A entrada consiste de vários casos de teste, cada caso de teste começa com 3 números inteiros  $P$ , os pontos de turismo ( $0 < P \leq 100$ ),  $R$  o número de rotas terrestre que ligem 2 pontos ( $0 \leq R \leq 2 * P$ ) e  $T$  o número de transfers solicitados ( $0 < T \leq 10$ ) que deseja-se saber se será por terra ou água. A seguir são  $R$  linhas onde cada linha contém 2 números indicando que os dois pontos turísticos indicados pelos seus números possuem ligação por terra. E seguem-se  $T$  linhas onde cada linha são indicados 2 números perguntando se o transfer entre entre estes dois pontos respectivo é terrestre ou marinho. Os casos de teste terminam com o fim da entrada.

**Saída:**

A saída consiste em  $T$  linhas onde aparecem as palavras “TERRESTRE” se o transfer puder ser realizado por terra entre os dois pontos ou “MARITIMO” caso contrário. Após cada caso de teste deve haver uma linha em branco, inclusive após o último

**Exemplo de Entrada:**

```
12 6 4
3 4
5 3
6 7
8 7
8 9
12 1
1 6
6 9
2 3
3 7
3 1 2
3 2
1 3
2 3
```

**Saída para o Exemplo de Entrada**

```
MARITIMO
TERRESTRE
MARITIMO
MARITIMO
```

```
MARITIMO
TERRESTRE
```

# Problema J

## Jiu-Jitsu

Arquivo: jj.[c|cpp|java|py]  
Limite de tempo: 1 s

### O Problema:

Você e seu colega são professores de Jiu-Jitsu e resolveram criar uma competição entre seus alunos. Vocês conhecem muito bem cada um de seus alunos de forma que cada um têm um valor atribuído como capacidade de luta, e, certamente, aquele que tiver uma capacidade maior vence o que tiver uma capacidade menor. Como são competições rápidas, existe o empate quando os jogadores têm a mesma capacidade. No final, ganha o time que tiver mais vitórias, só vitória conta.

Só que seu colega quer uma escolha mais justa dos times, por ser sénior, você sempre irá escolher primeiro um competidor para seu time, e alternadamente ele e você escolherão, até o último. Sempre há um número par de competidores. Então seu colega propôs colocar os competidores em uma fila de capacidades aleatórias e cada um só pode escolher para si um competidor de uma das pontas. Definidos os times aí quem escolhe o lutador primeiro para o embate é seu colega, ele escolhe um lutador do time dele e você o oponente a partir de seu time.

Você percebeu que ainda assim consegue vantagem. Por fim, dada uma fila de competidores faça um algoritmo que apresente quantas vitórias você terá na competição, no mínimo, se fizer as escolhas para ganhar.

### Entrada:

Cada caso de teste é apresentado em uma linha na entrada. Na linha o primeiro número é o inteiro  $N$  par, ( $0 < N \leq 1000$ ) e em seguida são  $N$  valores onde o  $i$ -ésimo valor ( $0 \leq n_i \leq 10^4$ ) representa a capacidade do  $i$ -ésimo lutador na fila. Os casos de teste terminam com o final da entrada.

### Saída:

Para cada caso de teste deve se colocar em uma linha única o número mínimo de vitórias que é garantido a você conquistar, dadas as regras do jogo e considerando que tanto você quanto seu colega joguem para obter o máximo para si.

### Exemplo de Entrada:

```
10 83 55 32 91 86 34 26 85 93 39  
4 1 2 3 4  
6 1 6 2 5 3 4
```

### Saída para o Exemplo de Entrada:

```
4  
2  
3
```

No primeiro caso uma divisão ótima de jogadores deixa você com os jogadores de capacidades:  $\{83, 39, 85, 34, 91\}$  e para seu colega  $\{55, 26, 86, 32\}$  desta forma você consegue 4 vitórias, no máximo.

# Problema K

Klaus Kinski

Arquivo: klaus.[c|cpp|java|py]

Limite de tempo: 1 s

**O Problema:** Klaus Kinski é um renomado ator alemão. Foi paraquedista na Segunda Guerra mas acabou preso pelos aliados. No pós guerra buscou a carreira artística chegou a ser preso em hospital psiquiátrico por perseguir sua mecenas, e chegou a tentar suicídio por não conseguir trabalhos. Mas sua carreira prosperou a partir de "Dr. Jivago" de David Lean. Fez filmes como: "Aguirre, a Cúlera dos Deuses" na floresta amazônica com direção de Herzog, aliás neste contexto outro filme fantástico, também de Herzog: "Fitzcarraldo", com a participação de Cláudia Cardinale, José Lewgoy e Grande Otelo. Participou também de Westerns Spaghetti como "Por mais alguns dólares" e "Trinity e seus Companheiros". Chegou a ser convidado por grandes diretores como Federico Fellini, Pier Paolo Pasolini, Luchino Visconti ou Steven Spielberg, neste último para "Caçadores da arca perdida", mas a exemplo deste último que achou o roteiro "uma pilha bocejante de um monte de merda" ele sempre preferiu diretores mais "mediócrates" para ter mais sossego e ganhar até mais, mesmo assim teve grande parceria com Herzog, com 5 filmes incluindo "Nosferatu, o vampiro da noite". Sua carreira agrupa mais de 130 filmes.

Por conta disso o museu da SBC (Sociedade Brasileira de Cinema) resolveu fazer uma semana de homenagem a Klaus Kinski e pediu a você, que dada a lista cronológica de filmes, fizesse um algoritmo onde o usuário pudesse escolher um ano, ou dois anos, e seria exibido (após uma busca eficiente) a lista, na ordem cronológica, de filmes produzidos no ano ou período.

## Entrada:

A entrada é um arquivo único onde se inicia com dois inteiros:  $F$  ( $1 < F \leq 136$ ) que é a quantidade de filmes protagonizados pelo autor, e  $Q$  que indica a quantidade de consultas a ser realizadas ( $0 < Q \leq 61$ ). A seguir são  $F$  linhas com os filmes, em ordem cronológica, o nome de cada filme tem no máximo 60 caracteres ASCII imprimíveis sem acentuação seguido entre parêntesis pelo ano do filme e não existe parêntesis no nome do filme. A seguir são  $Q$  linhas onde cada linha contém uma ou duas datas representadas pelo ano (com 4 dígitos) todas datas estão no intervalo de 1948 a 2008. Quando duas datas a primeira é uma data menor que a segunda.

## Saída:

A saída conterá  $Q$  blocos de saída, cada bloco contém os filmes, na ordem da lista, no período selecionado. Se não houver filmes no período a saída deverá ser: "Entre Trabalhos". Após cada bloco, também o último, deve ter uma linha em branco.

## Exemplo de Entrada:

```
10 4
A Criatura (1985)
Comando Leopardo (1985)
O ultimo Dragao (1985)
A vinganca das Estrelas Roubadas (1985)
Perversao Assassina (1986)
Cobra Verde (1987)
Minhas Aventuras na Africa (1987)
Grandes Cacadores (1988)
Vampiro em Veneza (1988)
Paganini (1989)
1980 1985
1983
1988 1990
1987
```

## Saída para o Exemplo de Entrada:

```
A Criatura
Comando Leopardo
O ultimo Dragao
A vinganca das Estrelas Roubadas
```

```
Entre Trabalhos
```

```
Grandes Cacadores
Vampiro em Veneza
Paganini

Cobra Verde
Minhas Aventuras na Africa
```