Maratona de Programação

Caderno de Problemas

UFV, 18 de maio de 2023

Instruções:

- Este caderno contém 11 problemas: com páginas numeradas de 1 a 12, não contando esta página de rosto.
- Em todos os problemas, a entrada deve ser lida da entrada padrão e a saída deve ser escrita na saída padrão.

Informações gerais

- A entrada deve ser lida da entrada padrão e a saída escrita na saída padrão
- Todas as linhas, inclusive a última, tanto da entrada quanto da saída, devem ter fim-de-linha
- Sempre que uma linha contém vários valores, eles são separados por um único espaço em branco e nenhum outro espaço deve aparecer (nem linha em branco), tanto na entrada quanto na saída
- O alfabeto inglês é sempre usado, não deve haver acentos, cedilhas, etc, nem na entrada nem na saída

Tempos limites de execução

- Rodizio de mini pedaços de mini pizza – 0,2 s
- Nas demais – 1,0 s
- ATENÇÃO! Os tempos foram calibrados para C++ e são os mesmos para todas as linguagens; não há garantia que códigos em outras linguagens executem dentro destes tempos.

"*Esta é uma obra de ficção, qualquer semelhança com nomes, pessoas, fatos ou situações da vida real terá sido mera coincidência.*"

Fontes das figuras ilustrativas usadas nos problemas da prova:

- Campina Grande: arquivo pessoal
- Data da maratona: arquivo pessoal
- Eleição para reitor: gerada em https://imgflip.com/memegenerator/Two-Buttons
- Eleição para presidente: gerada em https://imgflip.com/memegenerator/Two-Buttons e criação com assistência do DALL-E 2
- PCV - O Problema da Capivara Visitante: https://www.artstation.com/artwork/8wB13G
- Rodízio de mini pedaços de mini pizza: https://br.pinterest.com/pin/27443878971727714/
- Lagoal: https://tenor.com/view/pantanal-logo-novela-pantanal-pantanal2022-gif-25498936 e criação com assistência do DALL-E 2
- πx: adaptada de http://wallyinuruguay.blogspot.com/2011/02/two-pesos.html
- Meia pizza: https://www.reddit.com/r/Pizza/comments/9kuwv0/half_pepperoni_half_margherita/
- Capibonacci e Pizzibonacci: https://www.theshirtlist.com/pizzibonacci-t-shirt/ e https://www.reddit.com/r/capybara/comments/p9gkx8/carpincho_flawless_auric_proportion/
- Baking Bread: https://twitter.com/victor_capote/status/1247543700919668737

Problema A. Campina Grande

Arquivo-fonte: "`campina.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

A final da Maratona Brasileira do ano passado foi em Campo Grande. Mas teve gente achando que fosse em Campina Grande... Os nomes das cidades são realmente parecidos. Felizmente nem todos confundiram e nenhum time foi para a Paraíba em vez do Mato Grosso do Sul.

A Maratona Mineira deste ano será realizada em Ouro Branco. Para evitar que os participantes confundam com outras cidades, por exemplo, Ouro Preto, Rio Branco, entre outras, você deve fazer um programa que faça um alerta em caso de nomes que causam confusão. Todos os nomes possuem duas palavras. Um nome causa confusão se sua primeira ou segunda palavra for igual respectivamente à primeira ou segunda palavra do nome correto.

Time do DPI/UFV na final nacional

Entrada

A entrada começa com uma linha contendo o nome correto da cidade onde será disputada a maratona. Em seguida, há um número inteiro N indicando quantas cidades serão testadas ($1 \leq N \leq 10$). Por fim, N linhas, cada uma com um nome de cidade (o nome correto não aparece nesta lista).

Todas os nomes de cidade da entrada são compostas por exatamente duas palavras, separadas por um espaço em branco, com todas as letras minúsculas.

Saída

Escreva uma linha para cada cidade a ser testada, contendo: "`CUIDADO!`", se a cidade pode causar confusão, segundo o critério mencionado acima; ou "`OK`", se não causar confusão.

Exemplos

Entrada	Saída
campo grande 5 campina grande abre campo pantano grande campo belo campos gerais	CUIDADO! OK CUIDADO! CUIDADO! OK

Entrada	Saída
ouro branco 6 ouro preto rio branco pato branco areia branca ouro fino belo horizonte	CUIDADO! CUIDADO! CUIDADO! OK CUIDADO! OK

Entrada	Saída
big field 4 campo grande big apple spring field valadares governator	OK CUIDADO! CUIDADO! OK

Problema B. Data da maratona

Arquivo-fonte: "`data.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Como muitos sabem, é tradição as Maratonas de Programação ocorrerem justamente em dias nos quais os alunos da UFV possuem provas de Cálculo (o que faz com que eles precisem fazer uma segunda chamada).

A oganização da Maratona Mineira de Programação lhe contratou para desenvolver um programa para ajudá-los a escolher a melhor data para a maratona de 2023 (obviamente, a melhor data é a na qual há mais provas!)

Professores de Cálculo anotaram a data da maratona

Entrada

A entrada começa com um número N indicando o número de provas de Cálculo marcadas pelo Registro Escolar da UFV ($1 \leq N \leq 1000$).

A seguir, há N linhas, cada uma indicando a data de uma prova de Cálculo. Cada linha contém um inteiro K (indicando que é uma prova de Cálculo K, com $1 \leq K \leq 1000$), seguido por uma data no formado DD/MM/AAAA (obviamente, AAAA será sempre 2023).

Saída

Imprima uma linha contendo a data com mais provas. Se houver empate, imprima a data mais próxima do início do ano.

Exemplos

Entrada	Saída
7 1 10/02/2023 2 10/02/2023 50 10/02/2023 1 15/04/2023 2 15/04/2023 3 15/04/2023 3 16/04/2023	10/02/2023

Explicação: no dia 10/02/2023 teremos 3 provas: Cálculo 1, Cálculo 2 e Cálculo 50. No dia 15/04/2023 teremos a mesma quantidade de provas, mas 10/02/2023 é a data mais próxima do início do ano.

Entrada	Saída
7 3 16/04/2023 200 10/02/2023 2 10/02/2023 50 10/02/2023 3 15/04/2023 200 10/02/2023 2 10/02/2023	10/02/2023

No dia 10/02/2023 temos 5 provas! Os alunos de Cálculo 200 vão fazer inclusive duas provas no mesmo dia!!! (apesar de serem no mesmo dia, elas contam como duas provas)

Problema C. Eleição para reitor

Arquivo-fonte: "`eleicao1.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Há poucos meses tivemos eleição para a reitoria da UFV. A apuração dos votos foi fácil porque havia apenas 2 candidatos, com números 1 e 2. O desafio da próxima eleição é que são previstos muitos candidatos e, com isso, surgiu um problema para criar o software que conta o número de votos obtidos por cada candidato.

A DTI criou um formulário para descobrir quais são os candidatos e quais números cada um deles utilizará para seu partido. Infelizmente, muitos deles acharam simplório e injusto que as urnas permitam apenas números naturais nas eleições e, assim, as urnas terão que ser adaptadas para suportar inúmeros tipos de números.

Primeiramente, o PANN (Partido dos Admiradores de Numeros Negativos) decidiu concorrer com o número -1. O PN (Partido Nulo), também lançou um candidato (com número 0). O PMDB (Partido dos Matemáticos do Brasil) teve uma ideia não muito racional de usar π. Porém, para evitar que seus eleitores não fiquem votando por uma eternidade, decidiram limitar o número de dígitos que os eleitores poderão digitar entre 1 e 15. Seguindo a mesma lógica, o PC do B (Partido das Capivaras do Brasil) terá um candidato com o número de Euler[1] e (limitado da mesma forma que o código do PMDB). Por fim, também teremos um candidato da Computação (do partido PDPI), com número 100111001.

Para facilitar a sua vida, a candidatura do partido dos Físicos foi impugnada (imaginem a complexidade que o software deveria ter para suportar um número imaginário, que seria a escolha deles!)

A No Bugs foi contratada para desenvolver o software da urna eletrônica capaz de ler os votos e, então, imprimir na tela o vencedor das eleições e repassou a tarefa para você.

Eleitor confuso...

Entrada

A entrada começa com um número N, indicando a quantidade de eleitores ($1 \leq N \leq 1000000$). A seguir, há uma linha contendo N números, cada um representando um voto.

Saída

Escreva o nome do partido que obteve mais votos: "`PANN`", "`PN`", "`PMDB`", "`PC do B`", "`PDPI`"; ou a palavra "`empate`", caso haja empate.

Exemplos

Entrada	Saída
6 3.14 -1 100111001 0 100111001 2.71	PDPI

Entrada	Saída
6 3.14 -1 100111001 0 100111001 0	empate

[1] $e = 2.7182818284590...$

Problema D. Eleição para presidente

Arquivo-fonte: "`eleicao2.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

No último final de semana ocorreu a eleição para presidente da comunidade de capivaras. Havia dois candidatos: Capinaro e Capissilva. Para votar em Capinaro deveriam pressionar 22 e para votar em Capissilva deveriam pressionar 13.

Como as capivaras não têm boa memória (não conseguem memorizar mais que 1 dígito), foram instruídas pelos candidatos a guardarem apenas a soma dos dígitos que deveriam pressionar. Assim, seria só olhar as opções na tela da urna eletrônica e escolher o número que dava a soma que tinham memorizado.

Determine o vencedor da eleição a partir dos valores memorizados pelas capivaras.

Capivara votando

Entrada

A entrada começa com um número N, o número de capivaras eleitoras ($1 \leq N \leq 10000$). Em seguida, uma linha com N inteiros de 1 dígito, indicando o número memorizado por cada capivara eleitora.

Saída

Escreva uma linha contendo uma das seguintes opções, de acordo com a entrada:

- `capissilva`: se Capissilva, 13, recebeu a maioria dos votos
- `capinaro`: se Capinaro, 22, recebeu a maioria dos votos
- `nenhum`: se a maioria dos números memorizados são votos inválidos
- `indeterminado`: se não for possível determinar o vencedor

Exemplos

Entrada	Saída
6 4 1 2 3 5 4	nenhum

Entrada	Saída
3 4 4 4	indeterminado

Problema E. PCV - O Problema da Capivara Visitante

Arquivo-fonte: "`pcv.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Na UFV há vários lagos e inúmeros riachos, que conectam todos os pares de lagos. Capivaristo, um famoso capivaro da UFV, adora passar as férias visitando todos os lagos e conhecendo as belezas naturais da nossa universidade.

Capivaristo de férias

Sua ideia é começar do seu lago em Dezembro e passar por cada um dos lagos exatamente uma vez, retornando ao local inicial após o fim das férias. Porém, temos um problema: alguns riachos possuem jacarés, que estão cobrando um pedágio para que as capivaras passem de uma lagoa para outra. Nos riachos sem jacaré, Capivaristo gasta sempre 1 real para atravessá-los (dinheiro que ele sempre gasta comprando um pouco de grama para lanchar nas lanchonetes que ficam ao longo dos riachos).

Você foi contratado para desenvolver um programa capaz de determinar o valor mínimo que Capivaristo gastará para sair de sua casa (na lagoa 1) e visitar todas outras lagoas exatamente uma vez (por meio dos riachos) e retornar para sua casa.

Entrada

A entrada começa com N, o número de lagoas na UFV ($5 \leq N \leq 10000$). A seguir, há N linhas, cada uma descrevendo os riachos de uma das lagoas. A i-ésima linha descreve os riachos da lagoa $i = 1 \ldots N$, no seguinte formato:

1. Um número K_i ($0 \leq K_i \leq \frac{N}{3}$), indicando o número de riachos com jacaré na lagoa i;
2. K_i pares de valores no formato $X : Y$, indicando que a lagoa i está ligada à lagoa X ($1 \leq X \leq N$) e que o jacaré no respectivo riacho cobra Y ($1 \leq Y \leq 2000000000$) reais para que a capivara passe pelo local (os riachos desta lagoa não descritos não possuem jacarés).

Se houver um riacho ligando a lagoa A à B com custo C, então na entrada também haverá esse mesmo riacho conectando B a A, também com custo C. Não há mais de um riacho ligando o mesmo par de lagoas.

Saída

Imprima o valor mínimo que Capivaristo gastaria para sair de sua casa (na lagoa 1), visitar todas as lagoas exatamente uma vez e retornar para sua casa ao final das férias.

Exemplos

Entrada	Saída
6 2 2:3 5:4 2 1:3 3:6 2 2:6 4:3 1 3:3 1 1:4 0	6

Nesse exemplo, Capivaristo poderia utilizar a rota $1 - 3 - 5 - 6 - 2 - 4 - 1$ e passar apenas por riachos sem jacarés, gastando 1 real em cada um deles.

Problema F. Rodizio de mini pedaços de mini pizza

Arquivo-fonte: "`rodizio.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Como manda a tradição, durante a Maratona Mineira de Programação de 2022 em Timóteo, os alunos e coaches dos times da UFV aproveitaram um momento de folga em um rodízio de pizza (TODO: lembrar de procurar rodízio em Ouro Branco). Porém, o local escolhido era um rodízio de mini pedaços de mini pizza (infelizmente não era um mini preço). Apesar da decepção com a demora para o rodízio começar, esse momento de descontração foi muito divertido.


Garçom servindo uma mini pizza

Além dos pedaços de mini pizza serem minúsculos, os garçons às vezes demoravam muitos minutos para trazer o mini alimento. Adicionalmente, como havia muitas pessoas na mesa da UFV, a mini pizza que um garçom trazia nunca era suficiente para servir a todos os presentes.

Por fim, outro problema era decidir se você deveria aceitar um novo pedaço ou aguardar. Se o garçom voltasse e seu prato ainda tivesse pizza, ele não lhe ofereceria um novo mini pedaço. Assim, uma pessoa gulosa pode aceitar uma pizza Capivara (mini Calabresa, Pimentão, Vagem e Rabanete) e, pouco tempo depois, perder (por não ter terminado de comer) a chance de comer a tão sonhada mini pizza de Sorvete.

Pensando nesse mini desafio, você deverá desenvolver um software capaz de escolher as melhores oportunidades de comer pizza. Assim, você poderá levá-lo em um mini notebook na próxima vez que for a um rodízio e, com isso, aproveitar melhor essa iguaria.

Entrada

A entrada começa com dois números N e K ($0 \leq N, K \leq 5000$), indicando, respectivamente, o número de vezes que o garçom trará pizza e o número máximo de pedaços que você consegue comer (na Computação K costuma ser grande).

A seguir, há N linhas, cada uma descrevendo um dos momentos em que o garçom traz pizza para sua mesa. Cada uma dessas linhas possui 3 inteiros t_i, q_i e c_i ($0 \leq t_i, q_i \leq 100000$, $1 \leq c_i \leq 100000$):

- t_i indica o momento, contado em minutos que faltam para o final do rodízio, em que o garçom serve esse pedaço de pizza (há um relógio com contagem regressiva nesse rodízio, para o desespero dos clientes da Computação – o que eles não sabem é que muitas vezes o garçom para de servir antes do tempo chegar a 0!!!);
- q_i indica a qualidade da pizza (em outras palavras, o tanto você gosta dessa pizza);
- c_i indica quantos minutos você gasta para comer o i-ésimo pedaço.

Se uma pessoa aceitar um pedaço no momento t_i, então ela poderá aceitar outro apenas a partir (inclusive) do momento $t_i - c_i$.

Sabe-se que dois mini pedaços de pizza podem ser oferecidos no mesmo instante de tempo, mas infelizmente o cliente pode aceitar apenas um deles. Um garçom pode oferecer pizza até no tempo 0 e o restaurante não se importa caso o cliente continue comendo mesmo após o final do rodízio.

Saída

Escreva uma linha na saída informando a soma máxima das qualidades das pizzas que você decidiu comer.

Exemplos

Entrada	Saída
5 4 8 9 1 7 3 1 4 2 2 3 3 3 1 3 3	17

Nesse exemplo, a melhor solução consiste em:

- Comer o primeiro pedaço no tempo 8 (qualidade 9)
- Comer o segundo pedaço no tempo 7 (qualidade 3)
- Comer o terceiro pedaço no tempo 4 (qualidade 2) – note que esta escolha te impede de aceitar o quarto pedaço, pois você estará comendo o terceiro até o tempo 2.
- Comer o quinto pedaço no tempo 1 (qualidade 3)

Entrada	Saída
3 1 8 9 1 7 3 1 0 3 3	9

Problema G. Lagoal

Arquivo-fonte: "`lagoal.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Como todos sabem, na UFV há uma linda região de lagos com uma bela fauna. Essas lagoas normalmente transbordam na época das chuvas, formando pântanos. Por isso, o local é conhecido como Lagoal.

Velho do lago observando a lagoa

Alguns anos atrás surgiu um temido jacaré. Porém, por enquanto (até onde sabemos) nenhum aluno foi utilizado para saciar a fome desse animal, apesar dos estudantes adorarem caminhar perto dos lagos. Há uma lenda que explica esse fenômeno: segundo ela, vive na UFV um velho (conhecido como Velho do Lago), que se transforma em capivara e salva os alunos mordidos pelo jacaré.

Uma difícil tarefa que o Velho precisa cumprir é encontrar o melhor hospital para o qual uma determinada vítima do jacaré precisa ser levada. Dependendo da gravidade dos ferimentos, algumas ruas de Viçosa não podem ser utilizadas (por causa dos buracos, os solavancos podem piorar a situação do pobre estudante).

Com a alta demanda de salvamentos, o Velho decidiu contratar a No Bugs para desenvolver um software capaz de ajudá-lo a escolher o melhor hospital. Você, como recém contratado, foi encarregado do desenvolvimento desse programa.

Dadas as informações sobre os segmentos de ruas de Viçosa (cada segmento conecta duas esquinas) e o tempo máximo T que o aluno consegue resistir aos ferimentos, sua tarefa é descobrir, para um dado hospital (que sempre fica em uma esquina), dentre todas as possíveis rotas que gastam até T minutos, aquela cuja pior rua tenha o valor mínimo de solavanco possível. Assim, o Velho poderá executar seu programa para os inúmeros hospitais que existem em Viçosa (hahaha) e escolher um cuja rota gasta T minutos ou menos e não gera solavancos fatais no pobre aluno. Assuma que o Velho vai sempre partir da esquina onde ficam as 4 pilastras (esquina 0).

Obs: apesar da lenda ser verdadeira, recomendamos que continuem tomando cuidado no Lagoal (pois o software desenvolvido pela No Bugs pode ainda ter bugs e isso pode atrapalhar salvamentos)

Entrada

A entrada começa com quatro números N ($1 \leq N \leq 30000$), R ($R \leq 1000000$), H ($0 \leq H < N$) e T ($0 \leq T \leq 1000000000$), indicando, respectivamente, o número de esquinas, a quantidade de ruas em Viçosa, o número da esquina onde está o hospital e o tempo máximo, em minutos, no qual o aluno consegue resistir aos ferimentos.

A seguir, há R linhas, cada uma descrevendo uma rua.

Cada linha contém 4 inteiros o_i, d_i, t_i, s_i indicando que a i-ésima rua conecta (em mão única) a esquina o_i à esquina d_i e que uma ambulância passando por ela gasta t_i minutos para atravessá-la e é submetida a s_i unidades de solavancos ($(0 \leq o_i, d_i < N$, $0 \leq t_i, s_i \leq 1000000000$).

Todos valores consecutivos são separados por espaço em branco.

Saída

Imprima a quantidade mínima de solavancos à qual o aluno seria submetido para ir à esquina H de Viçosa partindo da esquina 0 e utilizando uma rota que gaste no máximo T minutos.

Exemplos

Entrada	Saída
5 6 4 10 0 1 5 1 0 2 5 3 0 3 2 1 1 4 6 2 2 4 5 6 3 4 2 7	6

Explicação:

A rota 0 – 1 – 4 gera apenas 2 unidades de solavanco no pior trecho, mas é muito demorada (11 minutos).

A rota 0 – 2 – 4 gera 6 unidades de solavanco no pior trecho e gasta 10 minutos (dentro do limite).

A rota 0 – 3 – 4 gera 7 unidades de solavanco no pior trecho e gasta 4 minutos (dentro do limite).

Problema H. πx

Arquivo-fonte: "`pix.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

O Banco Central da Nlogônia contratou o grande Cientista da Computação Capivaristo para implementar um PIX para o país. Assim como muitos participantes de maratonas de programação, Capivaristo leu apenas o título do problema e implementou rapidamente uma solução, sem ler com atenção os requisitos do sistema.

Devido ao nome PIX, Capivaristo pensou que seriam aceitas apenas transferências de C$3.14 (3 Capicoins e 14 centavos) e implementou o sistema para aceitar apenas esse valor fixo! Isso gerou muita confusão nas lojas, pois assim que ficaram sabendo da implementação do PIX, todas as pessoas tinham abandonado o dinheiro em papel e as moedas e passado a utilizar a nova tecnologia.

A No Bugs foi contratada para tirar esse irritante bug do PIX. Enquanto isso, você foi chamado por alguns comerciantes para determinar se eles podem vender um determinado produto fazendo apenas transações pelo PIX ou não. Por exemplo, uma das formas de se pagar um produto que custa C$28.26 é a seguinte: o cliente faz 12 transferências pelo PIX para o comércio (totalizando C$37.68) e a loja faz de volta 3 transferências (totalizando C$9.42) e, com isso, é como se o cliente tivesse pago C$37.68 − C$9.42 = C$28.26.

Antiga moeda usada antes do πx

Entrada

A entrada contém um inteiro N indicando o valor de uma transação, em centavos de Capicoins ($0 \leq N \leq 100000$).

Saída

Escreva uma linha contendo a palavra `SIM`, se for possível vender o produto recebendo uma ou mais transações PIX, com ou sem troco em PIX, e `NAO`, caso contrário.

Exemplos

Entrada	Saída
2826	SIM

Entrada	Saída
314	SIM

Entrada	Saída
2827	NAO

Problema I. Meia pizza

Arquivo-fonte: "`meia.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Após as maratonas de programação na UFV, professores e alunos costumam pedir pizza. Cada um escolhe o sabor preferido e encomenda meia pizza. A pizzaria não entrega meia pizza, é claro, então os sabores são pedidos aos pares. O professor junta os pedidos dois a dois e faz a encomenda. Isso era simples, pois todos sabores tinham o mesmo preço. A única dificuldade era quando havia um número ímpar de pessoas, então meia pizza a mais era pedida e ninguém comia essa meia adicional.

Meia calabresa meia marguerita

Agora a coisa complicou, porque os preços são diferentes para cada sabor. Por exemplo, Marguerita e Calabresa custam 30, Frango com Catupiry custa 32 e Portuguesa 34. Pior ainda, quando a pizza tem dois sabores diferentes, o valor é o do sabor mais caro. Assim, pedir uma pizza meia Marguerita meia Frango com Catupiry custa 32 e uma meia Calabrasa meia Portuguesa custa 34. Total 66. Seria melhor pedir um meia Marguerita meia Calabresa, que custa 30, e uma meia Frango com Catupiry meia Portuguesa, que custa 34. Total 64.

Sua tarefa aqui é decidir como juntar as meia pizzas de forma a minimizar o custo total.

Entrada

A entrada começa com um linha contendo dois inteiros, P e A, definindo, respectivamente, a quantidade de sabores de pizza da pizzaria e a quantidade de alunos que encomendaram meia pizza ($1 \leq P \leq 200, 1 \leq A \leq 200$). A linha seguinte contém P valores, que são os valores das diferentes pizzas (considere numeradas de 1 a P). Por fim, uma linha com A inteiros, informando o sabor pedido por cada aluno.

Saída

Escreva o valor mínimo que pode ser pago para satisfazer todos os alunos.

Exemplos

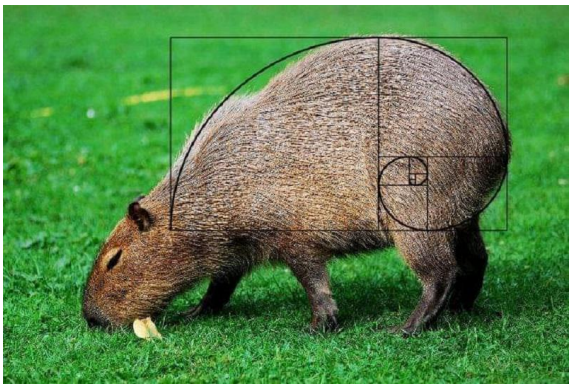
Entrada	Saída
5 4 30 30 32 34 38 1 4 2 3	64

Entrada	Saída
3 3 30 30 30 1 2 3	60

Problema J. Capibonacci e Pizzibonacci

Arquivo-fonte: "`capibonacci.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Todos nós sabemos por que pizza é tão saborosa e por que capivaras são tão simpáticas: elas seguem a proporção áurea.



Um fato menos interessante é que proporção áurea também é encontrada na razão de dois números consecutivos da sequência de Fibonacci. Aqui estamos interessados apenas no último dígito do n-ésimo termo da sequência de Fibonacci.

Entrada

A entrada contém de 1 a 200 linhas, cada uma com um número inteiro N ($0 \leq N \leq 10^{1000}$).

Saída

Para cada número N da entrada, escreva o último dígito do n-ésimo termo da série.

Exemplos

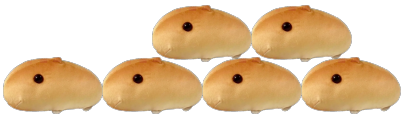
Entrada	Saída
0 1 2 3 4 5 6 7 8	0 1 1 2 3 5 8 3 1

Entrada	Saída
20 40	5 5

Problema K. Baking Bread

Arquivo-fonte: "`bread.x`", *onde* `x` *deve ser* `c`, `cpp`, `java` *ou* `py`

Walter Black é o dono de uma famosa pizzaria de Viçosa, muito frequentada pelos maratonistas de programação. Para aumentar seu lucro, sua pizzaria agora funciona também como padaria pela manhã. Mr. Black adora empilhar seus pães frescos em formato de capivara no balcão da sua padaria.

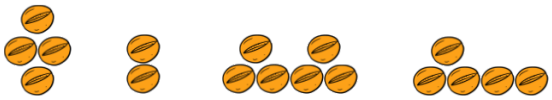


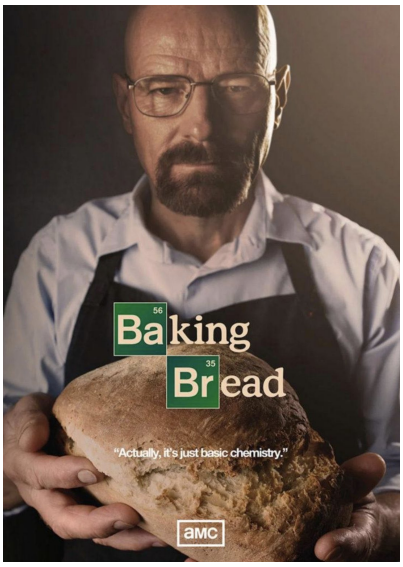
Os pães são dispostos em uma fileira, podendo ter mais fileiras em cima, obedecendo a duas restrições:

- Um pão sempre é colocado sobre dois outros da fileira de baixo;
- Todas as fileiras são alinhadas à direita.

Abaixo algumas das configurações possíveis com 6 pães e uma com 10 pães.

A seguir algumas configurações **não permitidas**. Nas duas primeiras, há pão que não está sobre dois outros. Nas duas últimas, as fileiras não estão alinhadas à direita.





Walter Black e um de seus pães

Entrada

A entrada contém um número N, o número de pães ($1 \leq N \leq 200$).

Saída

Escreva o número de configurações possíveis com N pães.

Exemplos

Entrada	Saída
1	1

Entrada	Saída
7	5