Centro de Educação Superior a Distância do
Estado do Rio de Janeiro – CEDERJ

Curso de Tecnologia em Sistemas de Computação – TSC

EAD-05.009 Fundamentos de Programação

Caderno de Exercícios
Aula 4

(Funções, Passagem de Parâmetros e Recursividade)

Professores

Dante Corbucci Filho
Leandro A. F. Fernandes

Instruções

- Utilize Python 3 e a IDE PyCharm na elaboração de soluções para os problemas propostos;
- A entrada de cada problema deve ser lida da entrada padrão (teclado);
- A saída de cada problema deve ser escrita na saída padrão (tela);
- Siga o formato apresentado na descrição da saída, caso contrário não é garantido que a saída emitida será considerada correta;
- Na saída, toda linha deve terminar com o caractere `'\n';`
- Utilize o URI Online Judge (http://www.urionlinejudge.com.br) e submeta sua solução para correção automática.

Referências Autorais

Os exercícios apresentados nesta lista foram extraídos do URI Online Judge (http://www.urionlinejudge.com.br). Acesse a URL apresentada abaixo do título de cada problema para proceder com a correção automática de sua solução e, também, para consultar a autoria do enunciado.

Problema A: Fibonacci, quantas chamadas?

https://www.urionlinejudge.com.br/judge/pt/problems/view/1029

Quase todo programador recebe em algum momento no início de seu curso de graduação algum problema envolvendo a sequência de Fibonacci. Tal sequência tem como os dois primeiros valores 0 (zero) e 1 (um) e cada próximo valor será sempre a soma dos dois valores imediatamente anteriores:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

Por definição, podemos apresentar a seguinte fórmula para encontrar qualquer número da sequência de Fibonacci:

$$fib(0) = 0,$$
$$fib(1) = 1,$$
$$fib(n) = fib(n-1) + fib(n-2).$$

Conforme mostra as expressões acima, uma das formas de encontrar o número de Fibonacci é através de chamadas recursivas à função fib.

Entrada

A primeira linha da entrada contém um único inteiro N, indicando o número de casos de teste. Cada caso de teste contém um inteiro X $(1 \leq X \leq 39)$.

Saída

Para cada caso de teste de entrada deverá ser apresentada uma linha de saída, no seguinte formato: `fib(n) = num_calls calls = result`, aonde `num_calls` é o número de chamadas recursivas, tendo sempre um espaço antes e depois do sinal de igualdade, conforme o exemplo abaixo.

Exemplo

Entrada	Saída
2 5 4	fib(5) = 14 calls = 5 fib(4) = 8 calls = 3

Problema B: Sequências de granizo

https://www.urionlinejudge.com.br/judge/pt/problems/view/1441

Considere a sequência formada iniciando-se por um inteiro positivo h_0 e iterando com $n = 1, 2, ...$ com a seguinte definição, até que $h_n = 1$:

$$h_n = \begin{cases} \frac{1}{2}h_{n-1}, & \text{se } h_{n-1} \text{ é par} \\ 3h_{n-1} + 1, & \text{se } h_{n-1} \text{ é ímpar} \end{cases}$$

Por exemplo, se iniciarmos com $h_0 = 5$ a seguinte sequência é gerada: 5, 16, 8, 4, 2, 1. Se começarmos com $h_0 = 11$, a sequência gerada é 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Como você pode ver nos exemplos, os números aumentam e diminuem, mas eventualmente terminam em 1 (isto é verdade para pelo menos para todos os números que já foram testados). Estas sequências são chamadas de Sequências de Granizo porque são similares à formação do granizo, pois são carregados para cima pelos ventos várias vezes, até que finalmente caem no chão.

Neste problema, dado um inteiro positivo, sua tarefa é computar o maior número na Sequência de Granizo que inicie com este o número dado.

Entrada

Cada caso de teste é descrito por uma única linha. A linha contém um inteiro H que representa o valor inicial para construir a sequência ($1 \leq H \leq 500$). O último caso de teste é composto por uma linha contendo um único zero.

Saída

Para cada caso de teste, imprima uma linha com um inteiro representando o maior número na Sequência de Granizo que inicia com o número da entrada.

Exemplo

Entrada	Saída
5 11 27 0	16 52 9232

* Problema originalmente proposto no ACM/ICPC South America Contest 2011.

Problema C: Jogo de varetas

https://www.urionlinejudge.com.br/judge/pt/problems/view/1366

Há muitos jogos divertidos que usam pequenas varetas coloridas. A variante usada neste problema envolve a construção de retângulos. O jogo consiste em, dado um conjunto de varetas de comprimentos variados, desenhar retângulos no chão, utilizando as varetas como lados dos retângulos, sendo que cada vareta pode ser utilizada em apenas um retângulo, e cada lado de um retângulo é formado por uma única vareta. Nesse jogo, duas crianças recebem dois conjuntos iguais de varetas. Ganha o jogo a criança que desenhar o maior número de retângulos com o conjunto de varetas.

Dado um conjunto de varetas de comprimentos inteiros, você deve escrever um programa para determinar o maior número de retângulos que é possível desenhar.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N que indica o número de diferentes comprimentos de varetas ($1 \leq N \leq 1.000$) no conjunto. Cada uma das N linhas seguintes contém dois números inteiros C_i e V_i, representando respectivamente um comprimento ($1 \leq C_i \leq 10.000$) e o número de varetas com esse comprimento ($1 \leq V_i \leq 1.000$). Cada comprimento de vareta aparece no máximo uma vez em um conjunto de teste (ou seja, os valores C_i são distintos). O final da entrada é indicado por $N = 0$.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha na saída, contendo um número inteiro, indicando o número máximo de retângulos que podem ser formados com o conjunto de varetas dado.

Exemplo

Entrada	Saída
1 10 7 4 50 2 40 2 30 4 60 4 5 15 3 6 3 12 3 70 5 71 1 0	1 3 2

* Problema originalmente proposto na Maratona de Programação da SBC 2007.

Problema D: Procurando Nessy

https://www.urionlinejudge.com.br/judge/pt/problems/view/1428

Em julho de 2003, a rede BBC fez uma grande investigação sobre o Lago Ness, usando 600 sonares separados. Nenhum vestígio de nenhum "mostro marítimo" (isto é, um grande animal, conhecido ou desconhecido) foi encontrado no lago. A equipe da BCC concluiu que Nessie não existe. Agora, nós queremos repetir este experimento.

Dada uma grade de N linhas e M colunas representando o lago, $6 \leq N, M \leq 10000$, encontre o menor número de sonares que você precisa colocar no lago de tal forma que podemos controlar todas as posições da grade, com as seguintes condições:

- Um sonar ocupa uma posição da grade; O sonar controla sua própria posição, além das suas posições adjacentes;
- As posições nas bordas da grade não precisam ser controladas, pois Nessie não conseguiria se esconder nelas (ela é grande demais para isso).

Entrada

A primeira linha da entrada contém um inteiro **t**, indicando o número de casos de teste. Cada caso de teste é descrito por uma linha contendo dois inteiros separados por um espaço, N e M ($6 \leq N, M \leq 10000$), indicando o tamanho da grade (N linhas e M colunas).

Saída

Para cada caso de teste, imprima uma linha contendo o menor número de sonares necessários.

Exemplo

Entrada	Saída
3 6 6 7 7 9 13	4 4 12

Problema E: O jogo matemático de Paula

https://www.urionlinejudge.com.br/judge/pt/problems/view/1192

Paula simplesmente adora matemática. Seu maior passatempo é ficar inventando jogos ou atividades que a envolvam para brincar com seus amiguinhos. Obviamente, nem todos eles não são tão apaixonados assim por matemática e têm muita dificuldade para resolver as brincadeiras propostas por ela. Agora Paula inventou um pequeno passatempo que envolve 3 caracteres: um dígito numérico, uma letra e outro dígito numérico.

Se a letra for maiúscula, deve-se subtrair o primeiro dígito do segundo. Se a letra for minúscula, deve-se somar ambos os dígitos e se os dígitos forem iguais, deve-se desconsiderar a letra e mostrar o produto entre os dois dígitos. Ela pediu para seu amigo Marcelo, que é bom em programação, para criar um programa para que encontre a solução para cada uma das sequências que Paula lhe apresentar.

Entrada

A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro *N*, indicando o número de casos de teste que virão a seguir. Cada caso de teste é uma sequência de três caracteres criada por Paula. Esta sequência contém na primeira posição um caractere de *'0'* a *'9'*, na segunda posição uma letra maiúscula ou minúscula do alfabeto e na terceira posição outro caractere de *'0'* a *'9'*.

Saída

Para cada caso de teste, deve ser impressa uma linha com um valor inteiro que representa a solução da sequência proposta por Paula.

Exemplo

Entrada	Saída
5 4A5 3A3 4f2 2G4 7Z1	1 9 6 2 -6

Problema F: Eletricidade

https://www.urionlinejudge.com.br/judge/pt/problems/view/1374

Martin e Isa se casaram. A vida é diferente nesse novo lugar. Particularmente, a energia elétrica é muito cara, e eles querem manter tudo sob controle. Por isso Martin propôs que mantivessem um histórico diário de quanta eletricidade foi consumida na casa. Eles têm um marcador de eletricidade, que mostra um número com a quantidade de KWh (kilowatts-hora) que foi consumida desde sua chegada.

No começo de cada dia eles consultam o marcador de eletricidade, e anotam o consumo. Alguns dias Martin faz isso, em outros é a Isa quem faz. Desse jeito, eles conseguirão observar as diferenças de consumo entre dias consecutivos e saber quanto foi gasto.

Mas alguns dias eles simplesmente esqueceram de anotar, então, depois de muito tempo, o histórico está incompleto. Eles têm uma lista de datas e consumos, mas nem todas datas são consecutivas. Eles só querem levar em conta os dias para os quais o consumo pode ser determinado precisamente, e precisam de ajuda.

Entrada

A entrada contém diversos casos de teste. A primeira linha de um caso de teste contém um inteiro N indicando o número de medições que eles fizeram ($2 \leq N \leq 103$). Cada uma das N linhas seguintes contém quatro inteiros D, M, Y e C, separados por espaços, indicando respectivamente o dia ($1 \leq D \leq 31$), mês ($1 \leq M \leq 12$), ano ($1900 \leq Y \leq 2100$), e consumo ($0 \leq C \leq 106$) lidos no início de cada dia. Essas N linhas são ordenadas em ordem crescente pela data e podem incluir anos bissextos. A sequência de consumos é estritamente crescente (isto é, duas leituras sempre têm valores diferentes). Você pode assumir que D, M e Y representam datas válidas.

Lembre-se que um ano é bissexto se ele é divisível por 4 e não por 100, ou então, se o ano é divisível por 400. O final da entrada é indicado por uma linha contendo apenas um zero.

Saída

Para cada caso de teste na entrada, seu programa deve imprimir uma única linha contendo dois inteiros separados por um único espaço: o número de dias para os quais o consumo pode ser determinado precisamente e o consumo desses dias.

Exemplo

Entrada	Saída
5 9 9 1979 440 29 10 1979 458 30 10 1979 470 1 11 1979 480 2 11 1979 483 3 5 5 2000 6780 6 5 2001 7795 7 5 2002 8201 8 28 2 1978 112 1 3 1978 113 28 2 1980 220 1 3 1980 221 5 11 1980 500 14 11 2008 600 15 11 2008 790 16 12 2008 810 0	2 15 0 0 2 191

Problema G: Loteria de fim de semana

https://www.urionlinejudge.com.br/judge/pt/problems/view/1407

As chances de se ganhar em uma loteria nacional são pequenas, por conta disso seus colegas de classe decidiram organizar uma loteria particular, cujo sorteio se realiza toda sexta-feira. A loteria é baseada em um estilo popular: um estudante que quer apostar escolhe C números distintos entre 1 e K e paga US$ 1.00 (note que as loterias tradicionais como a US National Lotto usam $C = 6$ e $K = 49$). Na sexta-feira durante o almoço, C números (também de 1 a K) são sorteados. O estudante que acertar a maior quantidade de números sorteados recebe o montante coletado nas apostas. O montante é dividido no caso de empates e acumulado para a próxima semana se ninguém acertar qualquer um dos números sorteados.

Alguns de seus colegas não acreditam nas leis da probabilidade e pediram para você para escrever um programa que determine os números que foram sorteados o menor número de vezes considerando todos os sorteios prévios, para que eles possam apostar nesses números.

Entrada

A entrada contém diversos casos de teste. A primeira linha de um caso de teste contém três inteiros N, C e K que indicam, respectivamente, o número de sorteios que já aconteceram ($1 \leq N \leq 10000$), quantos números compõem uma aposta ($1 \leq C \leq 10$) e o valor máximo que pode ser escolhido numa aposta ($C < K \leq 100$). Cada uma das próximas N linhas contém C inteiros distintos X_i indicando os números sorteados em cada concurso prévio ($1 \leq Xi \leq K$, para $1 \leq i \leq C$). O fim da entrada é indicado por $N = C = K = 0$.

Saída

Para cada caso de teste, seu programa deve escrever uma linha de saída, contendo o conjunto de números que foram sorteados o menor número de vezes. Este conjunto deve ser impresso como uma lista em ordem crescente. Deixe um espaço em branco entre dois números consecutivos na lista.

Exemplo

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

Problema H: Cavalo

https://www.urionlinejudge.com.br/judge/pt/problems/view/1513

Rafael gosta tanto de jogar xadrez que resolveu inventar novas maneiras de se desafiar. O desafio é o seguinte: Há um cavalo e K peões no tabuleiro. Dada uma posição inicial do cavalo e dos peões, qual a menor quantidade de movimentos necessários para capturar os K peões e voltar à posição inicial?

Lembre-se que a peça do cavalo pode mover-se usando saltos de formato L, ou seja, duas posições para a vertical e uma posição para a horizontal, ou duas posições para a horizontal e uma posição para a vertical. Para capturar um peão, basta ocupar a mesma posição que ele no tabuleiro.

Entrada

Haverá diversos casos de teste. Cada caso de teste inicia com três inteiro N, M e K ($5 \leq N, M \leq 100, 2 \leq K \leq 15$), representando, respectivamente, a quantidade de linhas e de colunas do tabuleiro, e a quantidade de peões a serem capturados.As próximas N linhas irão conter M caracteres cada, onde o caractere na linha i e coluna j indica que na posição $[i, j]$ do tabuleiro há:

- '.' uma posição válida onde o cavalo pode pular.
- '#' uma posição inválida onde o cavalo não pode pular.
- 'C' a posição inicial do cavalo de Rafael.
- 'P' a posição de um dos K peões o qual Rafael deve capturar.

O último caso de teste é indicado quando $N = M = K = 0$, o qual não deve ser processado.

Saída

Para cada caso de teste, imprima um inteiro, representando a quantidade mínima de saltos que o cavalo de Rafael deve fazer para capturar os K peões e retornar à posição inicial. É garantido que sempre haverá ao menos uma maneira de capturar todos os peões.

Exemplo

Entrada	Saída
5 5 2 ..... .P... ...P. ....C.. 4 6 2 .P##.P ..##.. ..##.. ..C... 0 0 0	4 8

*Problema originalmente proposto no Contest Bonilha 2014.