Problema 01) Números inteiros positivos
Escreva um programa em C que possui uma função RECURSIVA, para imprimir os **n** primeiros números naturais.
Entrada: Uma única linha com um número inteiro **n** (n>0), indicando a quantidade de números a ser impressos.
Saída: Uma única linha, contendo os **n** primeiros números naturais separados por um espaço em branco entre si.
Exemplo

Entrada	Saída
`12`	`1 2 3 4 5 6 7 8 9 10 11 12`

Problema 02) Números inteiros positivos 2
Escreva um programa em C que possui uma função RECURSIVA, para imprimir os **n** primeiros números inteiros positivos que são divisíveis por um outro número inteiro positivo **q**.
Entrada: Uma única linha com um número inteiro **n** (n>0), indicando a quantidade de números a ser impressos, seguido de um espaço em branco e de um número inteiro **q** (q>0), tal que **n** seja divisível por **q**.
Saída: A primeira linha deve conter os **n** primeiros números inteiros positivos divisíveis por **q**; a segunda linha possui o somatório dos números impressos.
Exemplo

Entrada	Saída
`4 3`	`3 6 9 12` `30`

Problema 03) Sequência de Fibonacci
Dado um número inteiro **n** (n>0), escreva um programa em C que possui uma função RECURSIVA para imprimir o n-ésimo termo da série de Fibonacci. Considere que $F_1 = 1$ e $F_2 = 1$.

Entrada: Uma única linha da entrada contém um número inteiro **n** (n>0), indicando a posição do termo desejado na sequência de Fibonacci.
Saída: Uma única linha, contendo o n-ésimo termo da sequência de Fibonacci.
Exemplo

Entrada	Saída
`11`	`89`

Problema 04) Batalha naval
Pedro e Paulo gostam muito de jogar Batalha Naval. Apesar de serem grandes amigos, Pedro desconfia que Paulo não esteja jogando honestamente e, para tirar essa dúvida, decidiu usar um programa de computador para verificar o resultado de cada jogo. Acontece que Pedro não sabe programar e, por isso, pediu a sua ajuda para elaborar este programa.

Cada jogador de Batalha Naval possui um tabuleiro retangular com n linhas e m colunas (n e m são números inteiros positivos|), onde:
• cada posição é um quadrado que pode conter água (’.’) ou uma parte de um navio (’N’);
• dois quadrados são ditos vizinhos se possuem um lado comum (ou seja, uma aresta que pertence a ambos quadrados); se duas partes de navio estão em posições vizinhas, então essas duas partes pertencem ao mesmo navio.

Cada disparo que um jogador faz em direção ao tabuleiro do seu oponente deve ser feito tendo como alvo um único quadrado daquele tabuleiro. Um jogador informa ao outro a linha L (1 <= L <= n) e a coluna C (1 <= C <= m) do quadrado alvo de seu disparo. Para que um navio seja destruído, o jogador deve acertar pelo menos uma das partes deste navio.

Você deve escrever um programa que, dadas a configuração do tabuleiro e uma sequência de disparos feitos por um jogador, determina o número de navios do outro jogador que foram destruídos.

Entrada: A primeira linha da entrada contém dois números inteiros n e m (1 <= n,m <= 100) representando, respectivamente, o número de linhas e de colunas do tabuleiro. As n linhas seguintes correspondem ao tabuleiro do jogo. Cada uma dessas linhas contém m caracteres, sendo que cada caractere indica o conteúdo da posição correspondente no tabuleiro. Se esse caractere for '.', essa posição contém água; se for 'N', essa posição contém uma parte de um navio.
A próxima linha contém um número k (1 <= k <= n×m) que representa o número de disparos feitos pelo jogador em direção ao tabuleiro de seu oponente. As próximas k linhas indicam os disparos feitos pelo jogador, em que cada contém dois inteiros L e C, indicando, respectivamente, a linha e a coluna do disparo feito (1 <= L <= n e 1 <= C <= m)
Saída: A primeira linha da saída possui uma sequência de três números inteiros, separados por espaços entre si: número de disparos que não acertaram qualquer navio; número de disparos que acertaram a posição [original] de qualquer navio; e número de navios destruídos. As próximas n linhas, cada qual com m caracteres, representam o estado do tabuleiro após a batalha, contendo os caracteres '.', 'N' e 'X', que denotam água, navio não destruído e navio destruído, respectivamente.

Exemplo

Entrada	Saída
5 8 .N.N.N.N .N...N.. ...N.... NN.N.NN. .N.NN... 8 1 3 1 4 1 5 2 1 3 4 4 6 4 4 5 2	3 5 4 .N.X.N.N .N...N.. ...X.... XX.X.XX. .X.XX...

Problema 05) Números primos
Escreva um programa que descobre quais são os números primos em um intervalo fechado de números inteiros. Uma função recursiva deve determinar se um dado número é primo.

Entrada: Um intervalo definido por dois números inteiros N1 e N2 (1 < N1 <= N2), separados por um espaço em branco.
Saída: A primeira linha apresenta a quantidade de números primos do intervalo definido por N1 e N2. Se no intervalo houver pelo menos um número primo, a segunda linha possui uma lista dos números inteiros presentes no intervalo, ordenados de forma crescente.

Exemplo

Entrada	Saída
`2 19`	`8` `2 3 5 7 11 13 17 19`
`14 16`	`0`

Problema 06) Mínimo múltiplo comum

Considere o seguinte problema:
"Um automobilista dá uma volta em uma pista circular em 12 minutos e um motociclista em 18 minutos. Os dois partem ao mesmo tempo às 08 horas. A que horas voltam a se encontrar no ponto de partida e, nesse instante, quantas voltas terão dado cada um?"
Esse é um clássico problema de Mínimo Múltiplo Comum (MMC).

Escreva um programa que calcula o MMC para um conjunto (com cardinalidade maior do que um, N>1) de números inteiros , tal que qualquer ni pertencente ao conjunto seja maior do que 1 (ni > 1). O programa deve ter uma função recursiva que calcula o MMC entre dois números inteiros.

Entrada: A primeira linha define a quantidade N de números inteiros (N>1). A segunda linha representa os números inteiros ni (ni > 1) separados por espaço em branco entre si.
Saída: O valor do MMC dos números ni.

Exemplo

Entrada	Saída
`5` `2 3 4 5 6`	`60`
`2` `14 16`	`112`

Problema 07) Máximo divisor comum

Considere o seguinte problema:
"O piso de uma sala retangular, medindo 352 cm × 416 cm, será revestido com ladrilhos quadrados, de mesma dimensão, inteiros, de forma que não fique espaço vazio entre ladrilhos vizinhos. Os ladrilhos serão escolhidos de modo que tenham a maior dimensão possível. Qual a medição do lado de cada quadrilho?"
Esse é um clássico problema de Máximo Divisor Comum (MDC).

Escreva um programa que calcula o MDC para um conjunto (com cardinalidade maior do que um, N>1) de números inteiros , tal que qualquer ni pertencente ao conjunto seja maior do que 1 (ni > 1). O programa deve ter uma função recursiva que calcula o MDC entre dois números inteiros.

Entrada: A primeira linha define a quantidade N de números inteiros (N>1). A segunda linha representa os números inteiros ni (ni > 1) separados por espaço em branco entre si.
Saída: O valor do MDC dos números ni.

Exemplo

Entrada	Saída
`3`	`4`

8 24 12	
5 2 3 4 5 6	1

Problema 08) Caixa automático

Caixas automáticos (ATMs – Automated Teller Machines) nos bancos são uma ótima invenção. Algumas vezes, precisamos de dinheiro trocado e a máquina nos entrega notas maiores, tal como $50,00. Por outro lado, desejamos sacar um valor um pouco maior e, por questão de segurança, gostaríamos de receber todo o valor em notas de $50,00, mas a máquina nos entrega um monte de notas de $2,00. O Banco Inteligente (BI) está tentando minimizar este problema dando aos clientes a possibilidade de escolher o valor das notas na hora do saque. Para isso, eles precisam da sua ajuda para saber, dado o valor S do saque (em reais) e quantas notas de cada valor a máquina tem, quantas formas distintas há para entregar o valor S ao cliente.

O BI disponibiliza notas de 1, 2, 5, 10, 30 e 50. Por exemplo, se S = 22 e o número de notas de cada valor é N1 = 5, N2 = 5, N5 = 4, N10 = 3, N30 = 10 e N50 = 0, então há VÁRIAS maneiras distintas da máquina entregar o valor do saque. Por exemplo, 20+2, 20+1+1, 10+10+2, 10+5+5+1+1, dentre outras.

A sua tarefa é determinar o número de maneiras possíveis de atender à solicitação de saque. Implemente uma rotina recursiva para descobrir as VÁRIAS maneiras distintas da máquina entregar o valor do saque.

Entrada: A primeira linha da entrada contém o número inteiro positivo S, expressando o valor do saque desejado. A segunda linha contém seis inteiros não negativos, separados por espaço em branco entre si: N1, N2, N5, N10, N30 e N50, referentes ao número de notas de 1, 2, 5, 10, 30 e 50, respectivamente, disponíveis na máquina no momento do saque.

Saída: Seu programa deve imprimir um número natural: a quantidade de maneiras distintas da máquina atender ao saque solicitado.

Exemplo

Entrada	Saída
22 5 4 3 10 0 10	12
100 20 20 20 20 20 20	1574

Problema 09) Caixa automático 2

Implemente uma rotina recursiva para descobrir as VÁRIAS maneiras distintas da máquina entregar o valor do saque, conforme a descrição abaixo de entradas e saídas.

Entrada: A primeira linha da entrada possui o número N de tipos de cédulas (N>1), que existe no caixa automático no momento do saque. Cada uma das N linhas seguintes apresenta o valor da cédula e a respectiva número ni de cédulas desse valor que há no caixa automático (ni>0), separados por um espaço em branco. Essas N linhas são ordenadas pelo valor da cédula. Por fim, a próxima linha contém o número inteiro positivo S, expressando o valor do saque desejado.

Saída: Várias linhas, onde cada linha representa uma forma distinta que a máquina pode atender ao saque solicitado, segundo o formato a_1xb_1 a_2xb_2 a_3xb_3 … A_Nxb_N (valores separados por um espaço em branco), tal que: (i) a_i representa a quantidade de cédulas de valor b_i, (ii) os valores em cada

linha devem ser ordenados de forma ascendente pelo valor das cédulas ($b_{i-1} < b_i < b_{i+1}$.), e (iv) as linhas devem ser ordenadas de forma decrescente pela quantidade de cédulas da maior valor. Veja os exemplos: as linhas estão ordenadas de forma decrescente pela quantidade de notas de $20, $10, $5 e $2, nessa ordem.

Exemplo

Entrada	Saída
4 **2 12** **5 14** **10 2** **20 3** **22**	**1x2 0x5 0x10 1x20** **1x2 0x5 2x10 0x20** **1x2 2x5 1x10 0x20** **6x2 0x5 1x10 0x20** **1x2 4x5 0x10 0x20** **6x2 2x5 0x10 0x20** **11x2 0x5 0x10 0x20**
4 **2 12** **5 14** **10 2** **20 3** **32**	**1x2 0x5 1x10 1x20** **1x2 2x5 0x10 1x20** **6x2 0x5 0x10 1x20** **1x2 2x5 2x10 0x20** **6x2 0x5 2x10 0x20** **1x2 4x5 1x10 0x20** **6x2 2x5 1x10 0x20** **11x2 0x5 1x10 0x20** **1x2 6x5 0x10 0x20** **6x2 4x5 0x10 0x20** **11x2 2x5 0x10 0x20**

Problema 10) Setas

Considere um tabuleiro quadrado, com N células de cada lado, em que cada célula possui uma seta que aponta para uma das quatro posições vizinhas. O jogador primeiro escolhe uma célula inicial para se posicionar e deve caminhar na direção para onde a seta em que ele está aponta. Ganha o jogo quem pisar em mais setas corretas durante um período de tempo.

Pode ocorrer que a célula inicial do jogo não é segura, pois leva a um caminho que termina fora do tabuleiro. As figuras abaixo mostram dois tabuleiros, em que os caracteres <, >, A e V referem-se a caminhar: para esquerda, para direita, para cima e para baixo, respectivamente. As posições marcadas demonstram posições que levam para fora do tabuleiro.

```
>>V          >>V<
AV<          A<<<
A<>          AAA>
             >>>A
```

Entrada: A primeira linha da entrada contém um inteiro positivo N (N>0), o tamanho do tabuleiro. Cada uma das N linhas seguintes contém N caracteres, com as direções das setas: <, >, A e V referem-se a caminhar: para esquerda, para direita, para cima e para baixo, respectivamente.

Saída: N linhas, mostrando posições em caminhos em ciclo (caractere L, 'loop'), posições em caminhos que levam a caminhos para fora do tabuleiro (caractere O, 'output'), e posições que não estão em ciclo nem levam a caminhos para fora do tabuleiro (caractere I, 'input').

Exemplo

Entrada Saída

```
3                    LLL
>>V                  LLL
AV<                  LLO
A<>
4                    LLLI
>>V<                 LLLI
A<<<                 IIIO
AAA>                 OOOO
>>>A
4                    ILLI
>>V<                 ILLI
AA<<                 IIIO
AAA>                 OOOO
>>>A
```