Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

Sumário

1 Cursos

 (+)

Uma universidade particular possui uma tabela de valores de créditos por curso com os campos:

- código do curso (int),
- valor por crédito (float).
- nome do curso (cadeia com no máximo 100 caracteres),

Faça um programa que primeiramente carregue a tabela de cursos acima e depois leia registros de alunos com as seguintes informações:

- nome do aluno (cadeia com no máximo 500 caracteres),
- o código do curso onde o mesmo está matriculado (int), e
- o número de créditos que ele está cursando (int).

Calcule a mensalidade a ser paga pelo aluno e imprima as informações em um boleto de pagamento com as seguintes informações: nome do aluno, nome do curso, total de créditos que o aluno cursa, valor do crédito do curso e o valor final da mensalidade a pagar.

Entrada

A primeira linha da entrada contém o número n ($5 \leq n \leq 30$) de cursos a universidade. Em seguida há $3n$ linhas contendo as três informações dos n cursos. A próxima linha contém um valor inteiro que corresponde ao número m ($1 \leq m \leq 1000$) de alunos da universidade. Em seguida, há $3m$ linhas, com os três dados dos m alunos.

Saída

A saída é formada por m linhas, cada uma no seguinte formato: Aluno(a): a Curso: b Num. Creditos: c Valor Credito: d Mensalidade: e, onde a e b são cadeias de caracteres, c é um número inteiro e d e e são números em ponto flutuante com duas casas decimais.

Exemplo

Entrada
4 298 234.5 Direito 123 150.00 Eng. Eletrica 452 132.00 Eng. Civil 341 120.00 Farmacia 5 Joao Luiz 298 5 Paula Lima 452 4 Maria 123 6 Luiz Andre 341 3 Antonio Luiz 341 8
Saída
Aluno(a): Joao Luiz Curso: Direito Num. Creditos: 5 Valor Credito: 234.50 Mensalidade: 1172.50 Aluno(a): Paula Lima Curso: Eng. Civil Num. Creditos: 4 Valor Credito: 132.00 Mensalidade: 528.00 Aluno(a): Maria Curso: Eng. Eletrica Num. Creditos: 6 Valor Credito: 150.00 Mensalidade: 900.00 Aluno(a): Luiz Andre Curso: Farmacia Num. Creditos: 3 Valor Credito: 120.00 Mensalidade: 360.00 Aluno(a): Antonio Luiz Curso: Farmacia Num. Creditos: 8 Valor Credito: 120.00 Mensalidade: 960.00

2 Vetores Ordenados

 (++)

Faça um programa que leia vários pares de pontos no espaço de quatro dimensões e calcule a norma do vetor correspondente a cada vetor e imprima as normas dos vetores em ordem crescente. A norma de um vetor $A(a_u, a_x, a_y, a_z)$ no espaço tetradimensional corresponde a sua distância e o ponto de origem e $O(0,0,0,0)$ e é calculada como:

$$||A|| = \sqrt{a_u^2 + a_x^2 + a_y^2 + a_z^2} \tag{1}$$

Voce deve usar um vetor de structs para armazenar as coordenadas e a norma de cada ponto.

Entrada

A entrada consiste de várias linhas. A primeira linha apresenta um número de pontos N, com $2 \leq N \leq 1000$. As N linhas seguintes apresentam pontos no espaço na forma $u,x,y.z$ com u,x,yez números reais tais que $-1000 \leq u,x,y,z \leq 1000$.

Saída

A saída consiste de N linhas, cada uma no formato: "Vetor: (a, b, c, d) Norma: x", onde a,b,c,d correspondem à coordenadas de um vetor lido, com duas casas decimais cada e x o valor de sua norma com duas casas decimais. As linhas devem conter os vetores em ordem crescente de norma.

Exemplo

Entrada
4 1 1 5 2 2 -1 3 0.2 4 2 -1 0.9 -3 4 2 34.2
Saída
Vetor: (2.00, -1.00, 3.00, 0.20) Norma: 3.75 Vetor: (4.00, 2.00, -1.00, 0.90) Norma: 4.67 Vetor: (1.00, 1.00, 5.00, 2.00) Norma: 5.57 Vetor: (-3.00, 4.00, 2.00, 34.20) Norma: 34.62

3 Strings Econômicas

 (+)

Escreva um programa em C para ler n nomes de pessoas na entrada e armazenar esses nomes em um vetor em que cada elemento deve ter espaço suficiente apenas para armazenar um nome lido e o caractere '\0'. Cada nome deve ser lido primeiramente em um *buffer* que é uma string com memória suficiente para armazenar mais do que o maior nome esperado (ex. 10000 caracteres). Em seguida deve ser alocado espaço num elemento do vetor que seja suficiente apenas para armazenar a string que está no buffer mais o caractere delimitador de cadeia. Em seguida, o programa copia (com strcpy) a cadeia no buffer para o espaço alocado no vetor e volta a ler outra cadeia no buffer. O programa deve imprimir todas as strings no vetor e, antes de terminar, deve liberar todo espaço alocado dinamicamente no vetor.

Observação

Você pode usar as funções *malloc* e *free* da stdlib para alocar e liberar espaço para armazenar os elementos do vetor. Pode usar a função *strcpy* para copiar uma string lida no buffer para um elemento alocado no vetor.

Entrada

A primeira linha da entrada contém o número n ($1 \leq n \leq 30$) de nomes a serem lidos. Em seguida, aparecem n linhas, cada uma com um nome.

Saída

A saída é formada por n linhas cada uma contendo um nome armazenado no vetor.

Exemplo

Entrada
3 Joao Antonio Maria Pedro Rezende de Souza Adriana Lucia de Assis
Saída
Joao Antonio Maria Pedro Rezende de Souza Adriana Lucia de Assis

4 Coelhos Mutantes

 (++)

Temos coelhos posicionados em uma linha, numerados de 1 a x. Os coelhos em posições ímpares possuem 2 orelhas, já os coelhos em posições pares possuem 3 orelhas, por terem sofrido mutações genéticas. Recursivamente retorne o número de orelhas de todos os coelhos enfileirados. (Sem estruturas de repetição ou multiplicação).

Entrada

A primeira linha na entrada contém um número inteiro n que corresponde ao número de casos de teste. Cada caso de teste contém o limite superior x do intervalo fechado $[1, x]$ que corresponde às posições onde há coelhos.

Saída

A saída é formada por n linhas cada uma contendo a soma do número de orelhas de coelhos de um caso de teste.

Exemplo

Entrada
4 1 3 5 7
Saída
2 7 12 17

5 Cursos - Versão 2

 (+)

Uma universidade particular possui uma tabela de valores de créditos por curso com os campos:

- código do curso (int),
- valor por crédito (float).
- nome do curso (cadeia com no máximo 100 caracteres),

Faça um programa que primeiramente carregue a tabela de cursos acima e depois leia registros de alunos com as seguintes informações:

- nome do aluno (cadeia com no máximo 500 caracteres),
- o código do curso onde o mesmo está matriculado (int), e
- o número de créditos que ele está cursando (int).

Os nomes dos cursos e dos alunos devem ter espaço suficiente apenas para armazenar os caracteres que formam os nomes e o caractere delimitador da string. Para isso, seu programa deve ler inicialmente os nomes (de cursos e de alunos) em um buffer que é uma string com 1000 caracteres. Para cada nome lido deve ser alocado um espaço para ele mais o '\0' no elemento do vetor onde esse nome será armazenado. Calcule a mensalidade a ser paga pelo aluno e imprima as informações em um boleto de pagamento com as seguintes informações: nome do aluno, nome do curso, total de créditos que o aluno cursa, valor do crédito do curso e o valor final da mensalidade a pagar. **Ao final, seu programa deve liberar todos os espaços alocados para armazenar os nomes de cursos e de alunos.**

Entrada

A primeira linha da entrada contém o número n ($5 \leq n \leq 30$) de cursos a universidade. Em seguida há $3n$ linhas contendo as três informações dos n cursos. A próxima linha contém um valor inteiro que corresponde ao número m ($1 \leq m \leq 1000$) de alunos da universidade. Em seguida, há $3m$ linhas, com os três dados dos m alunos.

Saída

A saída é formada por m linhas, cada uma no seguinte formato: Aluno(a): a Curso: b Num. Creditos: c Valor Credito: d Mensalidade: e, onde a e b são cadeias de caracteres, c é um número inteiro e d e e são números em ponto flutuante com duas casas decimais.

Exemplo

Entrada
4 298 234.5 Direito 123 150.00 Eng. Eletrica 452 132.00 Eng. Civil 341 120.00 Farmacia 5 Joao Luiz 298 5 Paula Lima 452 4 Maria 123 6 Luiz Andre 341 3 Antonio Luiz 341 8
Saída
Aluno(a): Joao Luiz Curso: Direito Num. Creditos: 5 Valor Credito: 234.50 Mensalidade: 1172.50 Aluno(a): Paula Lima Curso: Eng. Civil Num. Creditos: 4 Valor Credito: 132.00 Mensalidade: 528.00 Aluno(a): Maria Curso: Eng. Eletrica Num. Creditos: 6 Valor Credito: 150.00 Mensalidade: 900.00 Aluno(a): Luiz Andre Curso: Farmacia Num. Creditos: 3 Valor Credito: 120.00 Mensalidade: 360.00 Aluno(a): Antonio Luiz Curso: Farmacia Num. Creditos: 8 Valor Credito: 120.00 Mensalidade: 960.00

6 Logaritmo na Base Dois Recursivo

 (++)

O **piso** de um número real x é o único número inteiro i tal que $i \leq x \leq i+1$. O piso de x é denotado por $\lfloor x \rfloor$. Em computação usa-se muito computar o piso do logaritmo na base 2 de um número x, isto é, $\lfloor \log_2 x \rfloor$. Lembrando que $\log_2 x = y$, se $2^y = x$. Por exemplo, a tabela abaixo mostra alguns números positivos e os respectivos pisos dos seus logaritmos na base 2:

x	$\lfloor \log_2 x \rfloor$
15	3
16	4
31	4
32	5
63	5
64	6

O cálculo de $\lfloor \log_2 x \rfloor$ pode ser obtido, gerando-se uma série de números inteiros resultantes de sucessivas divisões inteiras por dois, iniciado-se com o quociente de x dividido por dois e continuando-se com as divisões por dois dos sucessivos quocientes resultantes, até que se obtenha um quociente igual a um. O número de elementos na série resultante corresponde ao $\lfloor \log_2 x \rfloor$. Veja alguns exemplos:

$$\lfloor \log_2 15 \rfloor = |\{7,3,1\}| = 3$$
$$\lfloor \log_2 16 \rfloor = |\{8,4,2,1\}| = 4$$
$$\lfloor \log_2 31 \rfloor = |\{15,7,3,1\}| = 4$$
$$\lfloor \log_2 32 \rfloor = |\{14,8,4,2,1\}| = 5$$
$$\lfloor \log_2 63 \rfloor = |\{31,15,7,3,1\}| = 5$$
$$\lfloor \log_2 64 \rfloor = |\{32,16,8,4,2,1\}| = 6$$

Escreva uma função recursiva para computar o piso do logaritmo na base dois de um número x fornecido como entrada. Lembre-se que $\lfloor \log_2 1 \rfloor = \log_2 1 = 0$.

Entrada

A primeira linha da entrada contém o número n $(0 < n \leq 100)$ de casos de teste. Cada caso de teste é formado por uma linha contendo um único número inteiro positivo x para o qual deseja-se coputar $\lfloor \log_2 x \rfloor$.

Saída

Para cada caso de teste o seu programa deve imprimir uma linha contendo o valor do piso do logaritmo na base dois do número que corresponde ao caso de teste.

Exemplo

Entrada:
6
15
16
31
32
63
64

Saída:
3
4
4
5
5
6

7 Ordenação por Data

 (++)

Uma determinada professora quer ordenar seus alunos em ordem crescente de idade. Escreva um programa em C que leia os dados dos alunos, entre eles a data de nascimento, ordene os alunos em ordem crescente de idade. Para isso seu programa deve ter uma função *ComparaDataNasc()* que recebe dois parâmetros. O primeiro corresponde a uma struct (ou um ponteiro para uma struct) que contém o dia, o mês e o ano de nascimento de uma aluno. O segundo parâmetro tem o memo tipo de dado do primeiro e contém a data de nascimento do segundo aluno. Essa função retorna 1 se o primeiro aluno é mais novo ou tem a mesma idade do segundo aluno e retorna zero em caso contrário. Essa função deve ser chamada pela função que ordena os alunos em ordem crescente de idade.

Entrada

A entrada contém apenas um caso de teste. A primeira linha da entrada contém um número inteiro n, ($1 \leq n \leq 30$) que corresponde ao número de alunos da turma. Em seguida há n linhas, contendo cada uma:

- a matrícula de um aluno (int);
- o dia de nascimento de um aluno (int);
- o mês de nascimento de um aluno (int);
- o ano de nascimento de um aluno (int);
- o nome de um aluno (no máximo 200 caracteres);

Saída

A saída é formada por n linhas, cada uma correspondendo ao um aluno, ordenadas em ordem crescente de idade dos alunos. Cada linha deve ter o seguinte formato: Matric.: *m* Nome: *n* Data Nasc.: *dd/mm/aa*, onde *m* é a matricula de um aluno, *n*, o seu nome, e *dd*, *mm* e *aa*, são respectivamente o dia, o mês e o ano do seu nascimento.

Exemplo

Entrada
5 12345 12 07 1978 Felizbina Freitas 23489 11 03 2009 Joao Feliz da Tristeza 98762 05 12 1976 Maria Batista de Souza 34561 11 07 1978 Roberto de Assis 34599 07 05 1976 Luiz Alberto Ferreira
Saída
Matric.: 23489 Nome: Joao Feliz da Tristeza Data Nasc: 11/3/2009 Matric.: 12345 Nome: Felizbina Freitas Data Nasc: 12/7/1978 Matric.: 34561 Nome: Roberto de Assis Data Nasc: 11/7/1978 Matric.: 98762 Nome: Maria Batista de Souza Data Nasc: 5/12/1976 Matric.: 34599 Nome: Luiz Alberto Ferreira Data Nasc: 7/5/1976

8 Palíndromo Recursivo

 (++)

Um palíndromo é uma palavra ou sequência de caracteres que tem a propriedade de poder ser lida tanto da esquerda para direita como da direita para a esquerda. Por exemplo, as seguintes palavras são palíndromos: *Ana, Bob, Otto, Mussum*. Escreva um programa que leia várias sequências de letras e que para cada uma delas chame uma função RECURSIVA que indique se uma sequência forma ou não um palíndromo.

Entrada

A entrada é formada por uma linha inicial que contém um número inteiro que indica o número de casos de testes. Para cada caso de teste há duas linhas. A primeira contém um número que indica o tamanho da sequência de letras a ser lida. O tamanho máximo da sequência é 2.000 letras. A segunda linha contém a sequência de letras, sendo que entre uma letra e outra há um espaço. A última letra da sequência vem seguida do caractere de quebra de linha.

Saída

Para cada seqüência da entrada o seu programa deve emitir uma das seguintes respostas: PALINDROMO, se a sequência for um palíndromo, ou NAO PALINDROMO, em caso contrário. Após cada palavra impressa deve haver apenas um caractere de quebra de linha.

Exemplos

Entrada:
5 5 N r C q K 4 M S K Y 3 L t V 2 K K 1 J

Saída:
NAO PALINDROMO NAO PALINDROMO NAO PALINDROMO PALINDROMO PALINDROMO

Observação

Cuidado ao ler a sequência de caracteres. Se você utilizar *scanf("%c", &vet [j])*, dentro do comando *while* para a leitura, o programa vai ler o espaços também e armazena-los no vetor. Sugestão: `while(j<tam-1) { {scanf("\%c " &vet[j]); scanf("\%c" &vet[j]) }` getchar()}. Repare que o `scanf()` dentro do `while` contém um espaço após %c e que o `scanf()` fora do coando `while` não tem esse espaço.

9 Potência Recursiva

 (++)

Dados dois números inteiros positivos M e N, escreva uma função recursiva que calcule o valor de M^N. Lembre-se que $M^0 = 1$.

Entrada

A primeira linha da entrada contém o número $T, 1 \leq T \leq 300$ de casos de teste. Cada caso de teste é formado por uma linha, contendo dois números inteiros positivos separados por um espaço.

Saída

Para cada caso de teste, seu programa deve imprimir a potência cuja base é o primeiro número e o expoente o segundo número.

Exemplo

Entrada:
3 2 5 3 9 1 50

Saída:
32 19683 1

10 Prefixo Recursivo

 (++)

Um prefixo de uma cadeia de caracteres é qualquer sub-cadeia que possa ser obtida apagando zero ou mais caracteres da extremidade direita da cadeia original. Escreva uma função RECURSIVA que recebe como parâmetro o endereço de memória onde está armazenada uma cadeia de caracteres e imprima todos os prefixos válidos da cadeia, exceto a cadeia vazia.

Escreva um programa para ler várias cadeias de caracteres da entrada e imprimir para cada cadeia o conjunto de prefixos não vazios da cadeia, utilizando a função mencionada acima.

Entrada

A primeira linha da entrada contém o número $C, 1 \leq C \leq 300$ de casos de testes. As C linhas seguintes contêm, cada uma, um caso de teste formado por uma cadeia de caracteres.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma linha contendo a frase : *Caso de teste #*. Em seguida deve imprimir todos os prefixos não vazios da cadeia correspondente ao caso de teste. Deve ser impresso um prefixo por linha e a primeira linha deve corresponder ao maior prefixo. A segunda linha deve conter o segundo maior prefixo, e assim por diante.

Exemplo

Entrada:
3
oi ola
casa
mesa

Saida:
Caso de teste1
oi ola
oi ol
oi o
oi
oi
o
Caso de teste 2
casa
cas
ca
c
Caso de teste 3
mesa
mes
me
m

11 Produto Recursivo

 (++)

Dados dois números inteiros positivos M e N, o produto de M por N é igual a zero se um dos dois é zero, caso contrário, o produto pode ser definido como N somas de M por ele mesmo. Escreva uma função recursiva que calcule o produto de um número inteiro por outro, utilizando somas.

Entrada

A primeira linha da entrada contém o número $T, 1 \leq T \leq 300$ de casos de teste. Cada caso de teste é formado por uma linha, contendo dois números inteiros positivos separados por um espaço.

Saída

Para cada caso de teste, seu programa deve imprimir o produto entre os dois números que formam o caso de teste.

Exemplo

Entrada:
3 23 11 45 50 0 900

Saída:
253 2250 0

12 Próximo Elemento 10 Vezes o Atual

 (+++)

Crie uma função recursiva que receba um vetor de inteiros e o seu respectivo tamanho como parâmetros. Essa função deverá verificar se há algum valor nesse vetor que, multiplicado por 10, seja igual ao próximo elemento, e então retornar 1. Caso contrário, retornar 0.

Entrada

A primeira linha da entrada contém o número de casos de teste. Cada caso de teste é formado por duas linhas. A primeira, contém o tamanho n ($1 \leq n \leq 30$) de um vetor. A segunda, contém n valores inteiros, separados entre si por um espaço.

Saída

A saída para cada caso de teste é formada por n linhas, cada uma contém uma das palavras: VERDADEIRO, se a condição é satisfeita pelo menos uma vez no vetor, ou FALSO, em caso contrário.

Exemplo

Entrada
3 6 0 10 100 1000 10000 100000 4 20 30 3 30 5 1 20 2 30 4
Saída
VERDADEIRO VERDADEIRO FALSO

13 Quantas Vezes ‘x’ Ocorre?

 (+)

Crie uma função recursiva que receba como entrada uma string e também o seu tamanho. Essa função deverá retornar quantas vezes o caractere ‘x’ aparece na string.

Entrada

A primeira linha na entrada contém o número n de casos de teste. Em seguida ocorrem n linhas, cada uma contendo uma string de tamanho máximo igual 5000 caracteres

Saída

A saída é formada por n linhas, cada uma contendo um número inteiro que indica o número de vezes que a letra ’x’ ocorre na entrada. Todas as strings estão escritas em letras minúsculas.

Exemplo

Entrada
3 abra cadabra taxi xadrez bacaxi taxa caixa puxa lixa xarope xxx
Saída
0 7 4

14 Quantas Vezes “hi” Ocorre?

 (+)

Crie uma função recursiva que receba como entrada uma string e também o seu tamanho. Essa função deverá retornar quantas vezes o caractere a subcadeia “hi” aparece na string.

Entrada

A primeira linha na entrada contém o número n de casos de teste. Em seguida ocorrem n linhas, cada uma contendo uma string de tamanho máximo igual 5000 caracteres

Saída

A saída é formada por n linhas, cada uma contendo um número inteiro que indica o número de vezes que a cadeia “hi” ocorre na string de entrada. Todas as strings estão escritas em letras minúsculas.

Exemplo

Entrada
4 hipotenusa hipotermia hilux hifi hi hihihi xavante he
Saída
4 1 3 0

15 Soma Recursiva de Elementos de um Vetor

 (+)

Calcule a soma de todos os elementos de um dado vetor de inteiros usando uma função recursiva.

Entrada

A primeira linha da entrada contém o número de casos de teste. Cada caso de teste é formado por duas linhas. A primeira, contém o tamanho n ($1 \leq n \leq 30$) de um vetor. A segunda, contém n valores inteiros, separados entre si por um espaço.

Saída

A saída para cada caso de teste é formada por n linhas, cada uma contém um número inteiro que é a soma dos elementos do vetor correspondente na entrada.

Exemplo

Entrada
3 5 3 2 1 4 5 5 1 2 3 4 5 10 1 2 3 4 5 6 7 8 9 100
Saída
15 15 145

16 Distância entre pontos

 (++)

Faça um programa que leia vários pares de pontos no espaço de quatro dimensões e calcule a distância entre eles. Considere que a distância entre dois pontos no espaço tetradimensional $A(a_u, a_x, a_y, a_z)$ e $B(b_u, b_x, b_y, b_z)$ é calculada como:

$$d(A,B) = |BA| = \sqrt{(a_u - b_u)^2 + (a_x - b_x)^2 + (a_y - b_y) + (a_z - b_z)^2} \tag{2}$$

Voce deve usar uma struct para representar cada ponto.

Entrada

A entrada consiste de várias linhas. A primeira linha apresenta um número de pontos N, com $2 \leq N \leq 1000$. As N linhas seguintes apresentam pontos no espaço na forma u, x, y, z, com u, x, y e z números reais tais que $-1000 \leq x, y, z \leq 1000$. Faça um programa que calcule a distância entre dois pontos consecutivos nesta lista. Note que, com exceção do primeiro e último valor de entrada, todos os pontos serão utilizados duas vezes, uma para o cálculo de distância com o ponto que veio antes na lista e outra para o ponto que veio depois.

Saída

A saída consiste de $(N-1)$ linhas, cada uma contendo a distância entre os pontos com 2 casas decimais após a vírgula. Após a impressão do valor de uma distância, o programa deve imprimir o caractere de quebra de linha.

Exemplo

Entrada
2 4 1 0 1 -1 2 1 2
Saída
5.29

Entrada
4 1 1 5 1 2 -1 3 0 4 2 -1 2 -3 4 2 2
Saída
3.16 5.74 7.87

Entrada
4 15.89 0.7 0.53 0.33 0.45 0.38 0.22 0.11 0 0 0 1 0 0 1 0.5
Saída
15.45 1.09 1.12

17 Soma Recursiva

 (+) Crie uma função recursiva que calcule a soma de 1 à n, onde n é um número inteiro positivo.

Entrada

A a primeira linha da entrada contém um número inteiro T que indica o número de casos de teste. Em seguida há T linhas, cada uma contendo um número inteiro positivo

Saída

A saída é formada por T linhas, cada uma contendo um número inteiro que representa a soma de 1 a n de uma linha correspondente na entrada.

Exemplo

Entrada
3 1 45 3
Saída
1 1035 6

18 Ordena palavras

 (++++)

Faça um programa que leia no máximo 100 palavras de no máximo 32 caracteres e as apresente na tela de forma ordenada crescente. Você deverá implementar as funções:

```
/**
 * Compara duas strings.
 * @param s1
 * @param s2
 * @return s1-s2
 */
int string_cmp(const unsigned char * s1, const unsigned char * s2);

/**
 * Troca o conteudo das strings s1 e s2
 * @param s1
 * @param s2
 */
void string_swap(char * s1, char * s2);
```

Entrada

O programa deve ler um número inteiro N, correspondente à quantidade de palavras a serem lidas e N palavras.

Saída

O programa deve apresentar N linhas com as palavras ordenadas de modo crescente.

Exemplo

Entrada	Saída
3 tempo lua cama	cama lua tempo

19 Soma e Subtração de Polinômios

 (++++)

Faça um programa para somar ou subtrair polinômios

Entrada

A primeira linha da entrada corresponde ao número de casos de teste. Cada caso de teste é formado por várias linhas. A primeira linha contém o caractere '+' ou o caractere '-', indicando se a operação é, respectivamente de soma ou de subtração de polinômios. A próxima linha contém um número inteiro n_1 que indica o número de termos do primeiro polinômio. Em seguida, há n_1 ($n_1 \leq 50$) linhas cada uma contendo um par de números c e e separados entre si por um espaço. O número c é um valor float, com sinal e corresponde ao coeficiente de um termo do polinômio. O valor e é um número inteiro positivo e corresponde ao expoente do termo do polinômio. Após o par n_1. Há um outro número inteiro n_2 ($n_2 \leq 50$) que corresponde ao número de termos do segundo polinômio, seguido, de $n-2$ linhas contendo pares de coeficientes e expoentes, como explicando para o caso do primeiro polinômio. Os pares de cada polinômio estão ordenados em ordem decrescente de expoente.

Saída

Para cada caso de teste o programa deve imprimir uma linha contendo o polinômio resultante escrito no seguinte formato:

$$sc_1X \wedge e_n sc_2X \wedge e_{n-1} \cdot sc_nX \wedge e_1$$

onde $s \in \{'+','-'\}$ é o sinal do coeficiente do polinômio, c_i é o valor do deficiente e e_i é o valor do expoente do termo i. Se durante a operação se soma ou subtração de polinômios um dos termos do polinômio resultante ficar com coeficiente igual a zero, esse termo não deve aparecer no polinômio resultante.

Sugestão

Represente cada termo do polinômio como uma struct com dois campos: *coeficiente* e *expoente*. Represente cada polinômio de entrada e também o polinômio resultante como vetores de structs. Escreva uma função para cada uma das seguintes ações:

- Leitura de um polinômio
- Impressão de um polinômio
- Soma de dois polinômios
- Subtração de dois polinômios

Exemplo

Entrada
2 + 5 -0.5 7 +4.2 4 -3 2 +1 1 -4 0 4 +3 5 +2 4 -1 2 +3 0 + 2 -5 7 -3 0 6 +2 6 +2 5 +3 4 -2.5 3 +3 2 -1.2 1
Saída
$-0.50X \wedge 7 + 3.00X \wedge 5 + 6.20X \wedge 4 - 4.00X \wedge 2 + 1.00X \wedge 1 - 1.00$ $-5.00X \wedge 7 + 2.00X \wedge 6 + 2.00X \wedge 5 + 3.00X \wedge 4 - 2.50X \wedge 3 + 3.00X \wedge 2 - 1.20X \wedge 1 - 3.00$

20 Quantidade de Cincos

 (+)

Crie uma função recursiva que tenha como entrada números inteiros, e verifique quantidade de dígito 5 que aparece em cada número.

Entrada

A primeira linha na entrada contém um número inteiro *n* que corresponde ao número de casos de teste. Em seguida, há uma linha para cada um todos *n* casos de teste. Cada linha contém um número inteiro positivo.

Saída

A saída é formada por *n* linhas cada uma contendo um número inteiro que corresponde ao número de vezes em que um dígito 5 ocorre no caso de teste correspondente na entrada.

Exemplo

Entrada
5 505505550 589559955 90000000 35 5
Saída
6 5 0 1 1