Lista 02 - Funções

Prof. Msc. Elias Batista Ferreira
Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

Sumário

1 Classificação do Aço (+)

 (+)

Um certo aço é classificado de acordo com o resultado de três testes abaixo, que devem determinar se o mesmo satisfaz às especificações:

1. Conteúdo de Carbono abaixo de 7.
2. Dureza Rockwell maior do que 50.
3. Resistência à tração maior do que 80.000 psi.

Ao aço é atribuído o grau "10" se passar por todos os testes; grau "9" se passar somente nos testes 1 e 2; grau "8" se passar no teste 1 apenas; grau "7" caso o aço não se enquadre nos graus, "10", "9", e "8".

Escreva uma função que receba como parâmetros o valor de conteúdo de carbono, o valor da dureza e o valor de resistência à tração de um aço e retorne o grau do aço. Desenvolver um programa que leia o conteúdo do carbono (CC), a dureza Rockwell (DR) e a resistência à tração (RT), chame a função especificada acima e imprima a classificação do aço.

Entrada

A entrada é formada por três linhas. A primeira, contém um valor inteiro correspondendo ao conteúdo do carbono (CC). A segunda linha contém um valor inteiro correspondendo à dureza Rockwell (DR). A terceira linha, contém um valor inteiro correspondendo à resistência à tração (RT).

Saída

O programa deve imprimir uma linha, contento a frase ACO DE GRAU = x, onde x é um dos graus possíveis de classificação do aço (7, 8, 9, ou 10). Após o valor do grau do aço, o program deve imprimir o caractere de quebra de linha '\n'.

Exemplo

Entrada
3 57 96783
Saída
ACO DE GRAU = 10

Entrada
2 61 80000
Saída
ACO DE GRAU = 9

Entrada
4 39 77000
Saída
ACO DE GRAU = 8

Entrada
7 32 65234
Saída
ACO DE GRAU = 7

2 Lê número (+)

 (+)

Escreva um programa que leia n números e apresente sua média. Você deve implementar uma função, denominada `le_numero`, que leia um número via terminal e retorne seu valor. A leitura dos dados deve ser realizada via a função `le_numero`. A função deve seguir o protótipo:

```
1 double le_numero();
```

Entrada

O programa deve ler um número n e em seguida n números reais.

Saída

O programa deve apresentar a média dos números lidos com 2 casas decimais.

Exemplo

Entrada	Saída
3 2 2 2	2.00

3 Quermesse (++)

 (++)

Os alunos do último ano resolveram organizar uma quermesse para arrecadar fundos para a festa de formatura. A festa prometia ser um sucesso, pois o pai de um dos formandos, Teófilo, dono de uma loja de informática, decidiu doar um computador para ser sorteado entre os que comparecessem. Os alunos prepararam barracas de quentão, pipoca, doces, ensaiaram a quadrilha e colocaram à venda ingressos numerados sequencialmente a partir de 1. O número do ingresso serviria para o sorteio do computador. Ficou acertado que Teófilo decidiria o método de sorteio; em princípio o sorteio seria, claro, computadorizado.

O local escolhido para a festa foi o ginásio da escola. A entrada dos participantes foi pela porta principal, que possui uma roleta, onde passa uma pessoa por vez. Na entrada, um funcionário inseriu, em uma lista no computador da escola, o número do ingresso, na ordem de chegada dos participantes. Depois da entrada de todos os participantes, Teófilo começou a trabalhar no computador para preparar o sorteio. Verificando a lista de presentes, notou uma característica notável: havia apenas um caso, em toda a lista, em que o participante que possuia o ingresso numerado com i, havia sido a i-ésima pessoa a entrar no ginásio. Teófilo ficou tão encantado com a coincidência que decidiu que o sorteio não seria necessário: esta pessoa seria o ganhador do computador.

Tarefa

Conhecendo a lista de participantes, por ordem de chegada, sua tarefa é determinar o número do ingresso premiado, sabendo que o ganhador é o único participante que tem o número do ingresso igual à sua posição de entrada na festa. **Você deve escrever uma função do tipo int que retorne o índice da pessoa sorteada.**

Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém um número inteiro positivo N, $N \leq 200$, que indica o número de participantes da festa. A linha seguinte contém a sequência, em ordem de entrada, dos N ingressos das pessoas que participaram da festa. O final da entrada é indicado quando $N = 0$. Para cada conjunto de teste da entrada haverá um único ganhador.

Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha identifica o conjunto de teste, no formato "Teste n", onde n é numerado a partir de 1. A segunda linha deve conter o número do ingresso do ganhador, conforme determinado pelo seu programa. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo

Entrada:
4 4 5 3 1 10 9 8 7 6 1 4 3 2 12 10 0

Saída:
Teste 1 3 Teste 2 10

4 Fatorial (++)

 (++)

Dado um número inteiro n, calcule seu fatorial $n!$. O fatorial de um número é dado pela equação: $n! = n(n-1)(n-2)\ldots 1$. Por definição, $0! = 1$.

Você deve implementar a função:

```
/**
 * Funcao que calcula o fatorial de um numero n
 * @param n um numero inteiro positivo
 * @return o fatorial de n
 */
unsigned long int fat( unsigned int n);
```

Entrada

O programa deve ler um número inteiro n.

Saída

O programa deve apresentar uma linha com a mensagem: "n! = f", onde n é o número lido e f o seu fatorial.

Observações

O fatorial de um número é resultado de uma operação de produtório que pode levar a valores incrivelmente grandes. Lembre-se de usar tipos de dados apropriados ao problema proposto.

Exemplo

Entrada
2
Saída
2! = 2

Entrada
4
Saída
4! = 24

5 Fibonacci (++)

 (++)

Faça um programa que, dados os termos iniciais da sequência de Fibonacci, calcule o n-ésimo número da sequência. Uma sequência é denominada sequência de Fibonacci segue todos os seus elementos são calculados pela soma de seus dois elementos antecessores. Exemplo:

$$F_n = F_{n-1} + F_{n-2} \tag{1}$$

Para $t_1 = 1$ e $t_2 = 1$, temos:

$$F(t_1, t_2) = F(1,1) = 1,1,2,3,5,8,13,21,... \tag{2}$$

Por exemplo: o quarto e o sétimo termos da sequência $F(1,1)$ são 3 e 13 respectivamente. Você deve implementar a função:

```c
/**
 * Retorna o n-ésimo termo da sequência de Fibonacci
 * @param t1 primeiro termo da sequência
 * @param t2 segundo termo da sequência
 * @param n a posição do termo desejado da sequência
 * @return o valor do n-ésimo termo da sequência
 */
int fibonacci( int t1, int t2, int n);
```

Entrada

O programa deve ler os dois termos iniciais t_1 e t_2 e a posição n do termo a ser retornado pela função.

Saída

O programa deve apresentar uma linha com o valor do n-ésimo termo da sequência.

Exemplo

Entrada	Saída
3 8 5	30

Entrada	Saída
10 10 5	50

Entrada	Saída
2 2 19	8362

6 Interceptos em x e em y de uma Reta (++)

 (++)

Dada a equação de uma reta: $y = ax + bx$, o *intercepto* em x corresponde ao ponto em que a reta toca o eixo x. O intercepto em y é o ponto em que a reta cruza o eixo y. Escreva um programa que leia uma quantidade n de coeficientes de equações de reta e imprima para cada um os interceptos em x e em y da reta.

Escreva duas funções intercetoEmX e interceptoEmY. Ambas possuem dois parâmetros de entrada do tipo float que correspondem aos coeficientes a e b da equação de uma reta e ambas possuem dois parâmetros de saída do tipo float correspondentes às coordenadas (x,y) do ponto intercepto correspondente a cada uma das funções. Seu programa deve usar essas funções para computar os interceptos em x e em y. Considere que a é sempre diferente de zero.

Entrada

A primeira linha da entrada contém um inteiro $n > 0$ que corresponde ao número de casos de teste. Em seguida há n linhas, cada uma correspondente a um caso de teste. Cada linha contém dois valores do tipo float, correspondentes aos coeficientes de uma reta.

Saída

Para cada caso de teste o programa deve imprimir duas linhas: "Intercepto em X: (x,0)", e "Intercepto em Y: (0.00,y)", onde (x,0.00) corresponde ao intercepto em x e (0,y) corresponde ao intercepto em y. Os valores de x e de y devem ser impressos com duas casas decimais.

Exemplo

Entrada
2 -9 4 -9 0
Saída
Intercepto em x: (0.44, 0.00) Intercepto em y: (0.00, 4.00) Intercepto em x: (0.00, 0.00) Intercepto em y: (0.00, 0.00)

7 Número de dígitos (++)

 (++)

Escreva um programa que leia um número *n* e apresente a quantidade de dígitos que ele possui.
Você deve implementar a função:

```
/**
 * Funcao que calcula a quantidade de digitos de um numero inteiro
 * @param n um numero inteiro
 * @return quantidade de digitos de n
 */
int digit_count(long int n);
```

Entrada

O programa deve ler um número inteiro *n*.

Saída

O programa deve apresentar uma linha com a mensagem: "Numero de digitos: *c*", onde *c* é a quantidade de dígitos de *n*.

Exemplo

Entrada	Saída
123	Numero de digitos: 3

8 Número Invertido (++)

 (++)

Escreva um programa para ler um número de três dígitos e imprimir o número invertido. Seu programa deve ter uma função: separaDigitos que possui quatro parâmetros. O primeiro, é um parâmetro de entrada e corresponde a um número inteiro (com três dígitos), os demais três parâmetros são de saída, e correspondem, respectivamente ao primeiro, ao segundo e ao terceiro dígitos do número correspondente ao primeiro parâmetro. A função recebe um valor inteiro no primeiro parâmetro e devolve os dígitos que o formam nos três parâmetros seguintes.

Entrada

A entrada contém apenas um número com três dígitos. Esse número é diferente de zero e não é múltiplo de 10 ou 100.

Saída

A saída deve conter apenas uma linha com o número correspondente ao valor da entrada, com seus dígitos invertidos. Logo após o número, deve ser impresso o caractere de quebra de linha: '\n'.

Exemplos

Entrada
123
Saída
321

Entrada
987
Saída
789

9 Número perfeito (++)

 (++)

Dado um número n inteiro e positivo, dizemos que n é perfeito se n for igual à soma de seus divisores positivos diferentes de n. Construa um programa que leia um número inteiro n, apresenta a soma dos divisores de n e verifica se o número informado é perfeito ou não.

Escreva uma função somaDivisores que receba como parâmetro um número inteiro e retorne a soma dos divisores desse número excluindo o próprio número como divisor de si mesmo. Seu programa deve chamar a função somaDivisores para resolver o problema.

Entrada

O programa deve ler um número inteiro n.

Saída

O programa deve apresentar uma linha contendo o texto: "n = d1 + d2 + d3 + ... + dk = x (MENSAGEM)", onde n é o número lido, d_i são os divisores de n em ordem crescente, x é a soma dos divisores e MENSAGEM é a mensagem "NUMERO PERFEITO"ou "NUMERO NAO E PERFEITO".

Observações

Suponha que o usuário sempre fornecerá um número maior que 1.

Exemplo

Entrada
`6`
Saída
`6 = 1 + 2 + 3 = 6 (NUMERO PERFEITO)`

Entrada
`12`
Saída
`12 = 1 + 2 + 3 + 4 + 6 = 16 (NUMERO NAO E PERFEITO)`

10 Raízes de equações de grau 2 (++)

 (++)

Desenvolver um programa que leia os coeficientes (a, b e c) de uma equação de segundo grau e calcule as raízes da equação. O programa deve mostrar a classificação das raízes, e, quando possível, o valor das raízes calculadas.

Seu programa deve criar uma função raizesEq2Grau que tenta computar as raízes de uma equação do segundo grau. A função deve retornar 2 se existir duas raízes reais distintas entre si, ou 1 se existir uma única raiz real, ou ainda, zero se as raízes são imaginárias. A função deve ter como parâmetros de entrada os coeficientes a, b, c de uma equação de segundo grau e deve ter dois parâmetros de saída, correspondendo às raízes da equação. No caso em que a função retorna 0 (raízes imaginarias) os parâmetros de saída não são utilizados pelo seu programa.

Entrada

O programa deve ler três valores reais na entrada. O primeiro valor corresponde ao valor do coeficiente a, o segundo, do coeficiente b e o terceiro, do coeficiente c, de uma equação de segundo grau. Os três valores ocorrem em uma única linha na entrada, separados entre si por um espaço.

Saída

O programa deve imprimir uma linha contendo uma das seguintes frases, conforme for o resultado do cálculo das raízes da equação: RAIZES DISTINTAS, ou RAIZ UNICA, ou RAIZES IMAGINARIAS. No primeiro caso o programa deve imprimir uma outra linha contendo a frase X1 = x_1, onde x_1 é o valor da menor raiz encontrada para a equação. Ainda no primeiro caso, o programa deve imprimir uma terceira linha com a frase X2 = x_2, onde x_2 corresponde ao valor da segunda raiz. No segundo caso, o programa deve imprimir uma frase X1 = x_1, onde x_1 é o valor da única raiz da equação. O terceiro caso não há o que imprimir pois as raízes são imaginárias.

Observações

Dada uma equação do segundo grau do tipo $ax^2 + bx + c$, Δ (delta) $= b^2 - 4ac$. Se $\Delta = 0$, a raiz da equação é ÚNICA. Se $\Delta < 0$. As raízes da equação são IMAGINÁRIAS. Se $\Delta > 0$, então há duas RAÍZES DISTINTAS para a equação. A fórmula geral para computar as raízes de uma equação do segundo grau é a fórmula de Báskara, dada por:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

Exemplo

A seguir são mostrados três exemplos distintos de entrada, e suas correspondentes saídas, entretanto, existe apenas uma linha de entrada para esse problema.

Entrada
`2 12 10`
Saída

`RAIZES DISTINTAS`
`X1 = -1.00`
`X2 = -5.00`

Entrada
2 12 18
Saída
RAIZ UNICA X1 = -3.00

Entrada
15 17 89
Saída
RAIZES IMAGINARIAS

11 Cálculo da raiz quadrada (+++)

 (+++)

Os Babilônios utilizavam um algoritmo para aproximar uma raiz quadrada de um número qualquer, da seguinte maneira:

Dado um número n, para calcular $r = \sqrt{n}$ assume-se uma aproximação inicial $r_0 = 1$ e calcula-se r_k para $k = 1, \ldots, \infty$ até que $r_k^2 \approx n$. O algoritmo deve realizar a aproximação enquanto $|n - r_k^2| > e$. O método babilônico é dado pela seguinte equação:

$$r_k = \frac{r_{k-1} + \frac{n}{r_{k-1}}}{2} \tag{3}$$

Funções a serem implementadas:

```
1
2  /**
3   * Função que calcula a raiz quadrada de n.
4   * @param n um numero real qualquer
5   * @return a raiz quadrada de n
6   */
7  double raiz( double n );
8
9  /**
10  * Valor absoluto de um numero qualquer
11  * @param n um número real qualquer
12  * @return o valor absoluto de n
13  */
14 double absoluto( double n );
```

Entrada

O programa deve ler um número **double** n, cuja raiz quadrada deseja-se obter, e o erro e que deverá ser considerado pelo algoritmo.

Saída

A saída deve apresentar cada iteração do algoritmo, sendo cada linha composta pelo valor aproximado da raiz quadrada de n com 9 casas decimais, seguido do erro, também com 9 casas decimais.

Exemplo

Entrada
2 0.00001
Saída
r: 1.500000000, err: 0.250000000 r: 1.416666667, err: 0.006944444 r: 1.414215686, err: 0.000006007

12 Escovando *bits* (+++)

 (+++)

Faça um programa que leia um número real (double), o converta para variáveis dos seguintes tipos de dados: `unsigned char`, `unsigned short`, `unsigned int`, `float`, `double` e apresentes os *bits* de cada *byte* de cada variável na mesma sequência da lista. Você deve implementar a função:

```
/**
 * Imprime os bits dos n bytes endereçados por end_byte.
 * @param end_byte endereço do primeiro byte a ser impresso
 * @param quantidade de bytes a serem impressos
 */
void print_bytes( const void * end_byte, int n );
```

Entrada

Um número real com dupla precisão.

Saída

Cinco linhas contendo os *bits* dos *bytes* de cada variável, separados por espaços.

Exemplo

Entrada	Saída
127	01111111 01111111 00000000 01111111 00000000 00000000 00000000 00000000 00000000 11111110 01000010 00000000 00000000 00000000 00000000 00000000 11000000 01011111 01000000

Entrada	Saída
256	00000000 00000000 00000001 00000000 00000001 00000000 00000000 00000000 00000000 10000000 01000011 00000000 00000000 00000000 00000000 00000000 00000000 01110000 01000000

Entrada	Saída
0.3	00000000 00000000 00000000 00000000 00000000 00000000 00000000 10011010 10011001 10011001 00111110 00110011 00110011 00110011 00110011 00110011 00110011 11010011 00111111

13 Escovando *bytes* (+++)

 (+++)

Faça um programa que leia um número real (double), o converta para variáveis dos seguintes tipos de dados: `unsigned char`, `unsigned short`, `unsigned int`, `float`, `double` e apresentes o conteúdo de cada *byte* de cada variável na mesma sequência da lista.

Entrada

Um número real com dupla precisão.

Saída

Cinco linhas contendo o valores dos *bytes* de cada variável, impressos como "%u"e separados por ’,’.

Exemplo

Entrada	Saída
127	127, 127,0, 127,0,0,0, 0,0,254,66, 0,0,0,0,0,192,95,64,

Entrada	Saída
256	0, 0,1, 0,1,0,0, 0,0,128,67, 0,0,0,0,0,0,112,64,

Entrada	Saída
0.3	0, 0,0, 0,0,0,0, 154,153,153,62, 51,51,51,51,51,51,211,63,

14 José (+++)

 (+++)

João tem um irmão mais novo, José, que começou a ir à escola e já está tendo problemas com números. Para ajudá-lo a pegar o jeito com a escala numérica, sua professora escreve dois números de três dígitos e pede a José para comparar esses números. Mas em vez de interpretá-los com o dígito mais significativo à esquerda, ele deve interpretá-lo com o dígito mais significativo à direita. Ele tem que dizer à professora qual o maior dos dois números. Escreva um programa que irá verificar as respostas de José.

Escreva uma função inverte que receba como parâmetro um número inteiro (de três dígitos) e retorne o número invertido. Sugestão: aprenda a aproveitar código! Copie a função SeparaDigitos que você escreveu para o problema "Número Invertido" e cole no seu programa. Faça a função inverte chamar a função SeparaDigitos para te auxiliar a computar o número invertido. O seu programa deve chamar a função inverte quantas vezes for necessário para resolver o problema.

Entrada

A entrada conterá um inteiro T , o número de casos de testes, e, para cada caso de teste, uma única linha com dois números de três dígitos, A e B, os quais não serão iguais e não conterão zeros.

Saída

A saída deve conter, numa linha para cada caso de teste, com o maior dos números na entrada, comparados como descrito no enunciado da tarefa. O número deve ser escrito invertido, para mostrar a José como ele deve lê-lo.

Exemplo

Entrada
3 734 893 221 231 839 237
Saída
437 132 938

15 Próxima potência (+++)

 (+++)

Escreva um programa que leia dois números inteiros, n e p, e calcule a número mais próximo de n que seja uma potência inteira de p. Por exemplo, para o número $n = 5$ e $p = 2$ o número mais próximo de 5 que é uma potência inteira de 2 é $4 = 2^2$. Para $n = 10$ e $p = 3$, o número mais próximo é $2^3 = 8$.

Você deve implementar a função:

```
/**
 * Função que calcula a potencia de p mais próximo a n.
 * @param n valor inteiro
 * @param p valor da potencia
 * @return retorna o valor da potencia mais proxima.
 */
int next_power( int n, int p );
```

Entrada

O programa deve ler os números inteiros n e p.

Saída

O programa deve apresentar uma linha com o valor da potência mais próxima a n no formato: "n -> k ^$p = x$", onde k é a base da potência e x o valor mais próximo de n.

Exemplo

Entrada	Saída
5 2	5 -> 2^2 = 4

Entrada	Saída
10 3	10 -> 2^3 = 8

Entrada	Saída
40 4	40 -> 2^4 = 16

16 *string to int* (+++)

 (+++)

Faça um programa que leia um número inteiro fornecido como uma *string* e o converta para um `long int`. Você deve implementar a função:

```
/**
 * Converte a string str para o valor inteiro correspondente.
 * @param str string contendo um número inteiro
 * @return o número inteiro correspondente
 */
long int string2int( const char * str );
```

Entrada

O programa deve ler uma sequência de *strings* contento um número inteiro, de no máximo 128 caracteres, usando o comando: `scanf("%s", str);`, até atingir o final do arquivo, ou seja, enquanto `( scanf("%s", str) != EOF )`.

Saída

A saída é composta por linhas linha contendo o número inteiro e o seu dobro impressos usando o comando `printf("%ld %ld\n", n, n*n);`, onde *n* é o número convertido.

Exemplo

Entrada	Saída
1 -2 3 -4	1 2 -2 -4 3 6 -4 -8

Entrada	Saída
15	15 30

Entrada	Saída
-1234	-1234 -2468

17 Triângulo ou trapézio? (+++)

 (+++)

Leia três valores reais (A, B e C) e verifique se eles formam ou não um triângulo. Em caso positivo, calcule o perímetro do triângulo e imprima a mensagem:

Perimetro = XX.X

Caso os valores não formem um triângulo, calcule a área do trapézio que tem A e B como base e C como altura, mostrando a mensagem:

Area = XX.X

Escreva as seguintes funções para resolver o problema:

- Uma função E_Triangulo que receba como parâmetros três valores do tipo float correspondendo aos três lados de um possível triângulo e que retorne um número inteiro contendo um dos possíveis valores: 1 - se os três números formam um triângulo, ou 0 - se os três números não formam os lados de um triângulo.
- Uma função Perimetro que recebe como parâmetros os valores de três lados de um triângulo e retorne um valor float correspondente ao perímetro do triângulo correspondente.
- Uma função areaTrapezio que receba como parâmetros três valores float correspondendo, respectivamente, à base inferior, à base superior e à altura de um trapézio e retorne um valor float correspondendo à area do trapézio.

Seu programa deve ler três valores float, verificar se formam um triângulo através do uso da função E_Triangulo. Se formarem triângulo, seu programa deve chamar a função Perimetro para computar o perímetro do triângulo formado pelos três valores e imprimir o perímetro. Se não formarem um triângulo, seu programa deve chamar a função areaTrapezio e deve imprimir a área do trapézio formado pelos valores.

Entrada

A entrada é formada por uma linha contendo três valores decimais separados um do outro por um espaço em branco.

Saída

A saída deve conter em uma única linha a frase apropriada. Observe nos exemplos acima que a saída deve conter apenas uma casa decimal. Os valores "X" que aparecem nos formatos são substituídos por dígitos que formam o valor de saída. Depois desses valores o programa deve imprimir o caractere de quebra de linha: '\n'.

Observações

Para que os três valores: A, B e C formem um triângulo as três condições abaixo devem ser satisfeitas:

- $|b-c| < a < b+c$;
- $|a-c| < b < a+c$;
- $|a-b| < c < a+b$;

A área de um trapézio é computada como Área$= \frac{(A+B)*C}{2}$.

Para imprimir um valor float com apenas uma casa decimal você deve usar a função **printf** com o código de formato "%.1f".

Exemplo

Entrada
`6.0 4.0 2.0`
Saída
`Area = 10.0`

Entrada
`6.0 4.0 2.1`
Saída
`Perimetro = 12.1`

18 Valor em Notas e Moedas (+++)

 (+++)

Escreva um algoritmo par ler um valor em reais e calcular qual o menor número possível de notas de $R 100, $R 50, $R 10 e moedas de $R 1 em que o valor lido pode ser decomposto. O programa deve escrever a quantidade de cada nota e moeda a ser utilizada.

Você deve escrever uma função converteEmNotasMoedas que possui 5 parâmetros. O primeiro parâmetro corresponde ao valor inteiro a ser convertido em notas e moedas, o segundo parâmetro corresponde ao número de notas de 100, o terceiro, corresponde ao número de notas de 50, o quarto, corresponde ao número de notas de dez e o quinto parâmetro corresponde ao número de moedas de 1 Real. Seu programa deve chamar essa função para resolver o problema proposto.

Entrada

O programa deve ler uma única linha na entrada, contendo um valor em Reais. Considere que somente um número inteiro seja fornecido como entrada.

Saída

O programa deve imprimir quatro frases, uma em cada linha: NOTAS DE 100 = X, NOTAS DE 50 = Y, NOTAS DE 10 = Z, MOEDAS DE 1 = W, onde X, Y, Z e W correspondem às quantidades de cada nota ou moeda necessárias para corresponder ao valor em Reais dado como entrada. Após cada quantidade, o programa deve imprimir um caractere de quebra de linha: '\n'.

Exemplo

Entrada
`46395`
Saída

`NOTAS DE 100 = 463`
`NOTAS DE 50 = 1`
`NOTAS DE 10 = 4`
`MOEDAS DE 1 = 5`

19 Cálculo de PI (++++)

 (++++)

Na história da ciência, muitas foram as tentativas de se encontrar o número π com a maior precisão possível. Uma dessas tentativas foi a do matemático John Wallis, que desenvolveu a série infinita da Equação 4 em 1655.

$$\frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \frac{8}{9} \cdots = \frac{\pi}{2} \tag{4}$$

Faça um programa que calcula o valor de *pi* usando a série proposta, permitindo o usuário definir a quantidade de termos da série.

Você deve implementar a função:

```c
/**
 * Função que calcula o valor de pi usando a série proposta por John Wallis
 * @param n quantidade de termos da série
 * @return o valor aproximado da constante pi
 */
double compute_pi( int n );
```

Entrada

O programa deve ler a quantidade de termos *n*.

Saída

O programa deve apresentar uma linha com o valor de π com 12 casas decimais.

Observações

Use precisão dupla (`double`) para os números reais.

Exemplo

Entrada	Saída
30	3.091336888596

Entrada	Saída
500	3.138458897672

Entrada	Saída
10000	3.141435593590

20 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:

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

/**
 * Troca o conteúdo 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

21 *string to double* (++++)

 (++++)

Faça um programa que leia um número real fornecido como uma *string* e o converta para um **`double`**. Um número real pode ter ou não o caracter '.' para separar a parte inteira da fracionária. Você deve implementar a função:

```
/**
 * Converte a string str para o valor real correspondente.
 * @param str string contendo um número real
 * @return o número inteiro correspondente
 */
double string2double( const char * str );
```

Entrada

O programa deve ler uma sequência de *strings* contento um número real, de no máximo 128 caracteres, usando o comando: `scanf("%s", str);`, até atingir o final do arquivo, ou seja, enquanto `( scanf("%s", str) != EOF )`.

Saída

A saída é composta por linhas linha contendo o número real e o seu dobro impressos usando o comando `printf("%.3lf %.3lf\n", n, n*n);`, onde *n* é o número convertido.

Exemplo

Entrada	Saída
1.01 -2.2 3.5 -4.0	1.010 2.020 -2.200 -4.400 3.500 7.000 -4.000 -8.000

Entrada	Saída
15	15.000 30.000

Entrada	Saída
0.5	0.500 0.250

Entrada	Saída
10.02	10.020 20.040

Entrada	Saída
-1234	-1234.000 -2468.000