Rua Alberto Hoffmann, 285, Restinga
CEP 91791-508 – PORTO ALEGRE- RS
Criado pela Lei nº 11.892 de 29 de dezembro de 2008.

Implemente um programa em **Python 3** que calcula o valor total da conta de luz de um único cliente. Serão informados o identificador do cliente, a leitura anterior e a leitura atual do medidor em kWh, a tarifa da bandeira (R$/kWh), o custo de distribuição (R$/kWh) e o valor da iluminação pública (R$). O consumo deve ser calculado como a diferença entre a leitura atual e a leitura anterior. O valor da energia é dado pelo produto entre o consumo e a tarifa; o valor da distribuição é o produto entre o consumo e o custo de distribuição; o valor da iluminação pública é somado diretamente. O total da conta corresponde à soma do valor de energia, do valor de distribuição e do valor da iluminação pública. Considere que todos os valores fornecidos são válidos e coerentes, de modo que o programa não necessita realizar validações.

ENTRADA

A entrada é composta por **seis linhas**, nessa ordem: a primeira linha contém o **identificador do cliente**; a segunda linha contém **a leitura anterior do medidor em kWh (inteiro)**; a **terceira linha contém a leitura atual do medidor em kWh (inteiro)**; a quarta linha contém a **tarifa da bandeira em reais por kWh (número real)**; a quinta linha contém o **custo de distribuição em reais por kWh (número real)**; e a sexta linha contém o **valor da iluminação pública em reais (número real)**.

SAÍDA

A saída deve conter cinco linhas. A primeira linha deve apresentar o **identificador do cliente**, seguido do **consumo em kWh**, no formato "**CLIENTE <id>: CONSUMO <kwh> KWH**". A segunda linha deve apresentar "**ENERGIA: R$** " seguido do **valor da energia com duas casas decimais**. A terceira linha deve apresentar "**DISTRIBUICAO: R$** " seguido do **valor da distribuição com duas casas decimais**. A quarta linha deve apresentar "**ILUMINACAO: R$** " seguido do **valor da iluminação pública com duas casas decimais**. A quinta linha deve apresentar "**TOTAL: R$** " seguido do **valor total da conta com duas casas decimais**.

Exemplo de entrada	Exemplo de saída
cli01 1200 1350 0.92 0.48 14.50	CLIENTE cli01: CONSUMO 150 KWH ENERGIA: R$ 138.00 DISTRIBUICAO: R$ 72.00 ILUMINACAO: R$ 14.50 TOTAL: R$ 224.50
AA10 100 260 1.20 0.30 18.00	CLIENTE AA10: CONSUMO 160 KWH ENERGIA: R$ 192.00 DISTRIBUICAO: R$ 48.00 ILUMINACAO: R$ 18.00 TOTAL: R$ 258.00

Você deve implementar um programa em **Python 3** que recebe a permissão de um arquivo Linux em notação octal e imprime indicadores booleanos sobre quais permissões estão ativas para dono, grupo e outros, além de dois indicadores agregados. No Linux, as permissões básicas são representadas por três dígitos octais: o primeiro dígito é do dono, o segundo do grupo e o terceiro de outros usuários. Cada dígito é a soma de 4 para leitura (r), 2 para escrita (w) e 1 para execução (x). Por exemplo, 7 significa r+w+x (4+2+1), 5 significa r+x (4+1) e 4 significa apenas r. Na permissão 755, o dono tem 7 (rwx), o grupo tem 5 (r-x) e outros têm 5 (r-x). O programa deve apenas ler a permissão, decodificá-la com operações de bits, e imprimir os resultados como valores booleanos.

ENTRADA

A entrada consiste em uma única linha contendo **a permissão em octal como texto**, por exemplo 755 ou 775. O programa interpretará esse valor, extrairá os três dígitos de permissões (dono, grupo e outros) e calculará os indicadores solicitados exclusivamente com operadores de bits, relacionais e lógicos.

SAÍDA

A saída deve conter doze linhas, cada uma apresentando uma etiqueta e um valor booleano **True** ou **False**. As etiquetas, na ordem, são: **OWNER_READ**, **OWNER_WRITE**, **OWNER_EXEC**, **GROUP_READ**, **GROUP_WRITE**, **GROUP_EXEC**, **OTHER_READ**, **OTHER_WRITE**, **OTHER_EXEC**, **OWNER_ALL**, **WORLD_READ** e **NO_WORLD_WRITE**. Os três primeiros indicam se o dono tem leitura, escrita e execução; os três seguintes indicam as permissões do grupo; os três subsequentes indicam as permissões de outros; OWNER_ALL indica se o dono tem as três permissões (rwx); WORLD_READ indica se outros possuem leitura; NO_WORLD_WRITE indica se outros não possuem escrita.

Exemplo de entrada	Exemplo de saída
755	OWNER_READ: True OWNER_WRITE: True OWNER_EXEC: True GROUP_READ: True GROUP_WRITE: False GROUP_EXEC: True OTHER_READ: True OTHER_WRITE: False OTHER_EXEC: True OWNER_ALL: True WORLD_READ: True NO_WORLD_WRITE: True
644	OWNER_READ: True OWNER_WRITE: True OWNER_EXEC: False GROUP_READ: True GROUP_WRITE: False GROUP_EXEC: False OTHER_READ: True OTHER_WRITE: False OTHER_EXEC: False OWNER_ALL: False WORLD_READ: True NO_WORLD_WRITE: True

Implemente um programa em **Python 3** que recebe o nome de um time do Campeonato Brasileiro e normaliza o texto do nome para apresentação. Ainda, deve ser possível a análise sem interferência de acentos e pontuações simples, e responder a perguntas como: se contém a marca "**FC**", se o nome inicia com a letra **S**, se termina com "**ENSE**", se a palavra "**CLUBE**" está presente, se o texto contém "**ATLÉTICO**" (com ou sem acento), se o nome é curto quando desconsiderados os espaços, se o nome é válido contendo apenas letras e espaços, e se é considerado "**DESTAQUE**" quando contém "**FC**", "**CLUBE**" ou "**GRÊMIO**" (com ou sem acento) e **não é curto**. As decisões devem ser feitas usando métodos de strings e operadores relacionais/lógicos, evitando estruturas que fujam do objetivo de manipular e consultar strings. O nome é considerado curto quando o número de caracteres, **desconsiderando espaços, é menor que 10**. O nome é considerado válido quando contém apenas **letras e espaços**. O rótulo "**DESTAQUE**" é verdadeiro quando o nome contém "**FC**", "**CLUBE**" ou "**GRÊMIO**" (com ou sem acento) e **não é curto.**

ENTRADA

A entrada é composta por uma única linha com o **nome do time**. O texto pode conter letras com acentos, misturar maiúsculas e minúsculas e conter espaços extras, pontos e hífens. O programa deve considerar uma versão em maiúsculas do texto, bem como uma versão sem acentos para certas análises, além de remover espaços duplicados para exibição.

SAÍDA

A saída é composta por **nove linhas**, cada uma apresentando uma etiqueta e o respectivo valor computado, no seguinte formato: **NOME_NORMALIZADO: <texto>; TEM_FC: <True|False>; COMECA_COM_S: <True|False>; TERMINA_COM_ENSE: <True|False>; TEM_CLUBE: <True|False>; TEM_ATLETICO: <True|False>; NOME_CURTO: <True|False>; NOME_VALIDO: <True|False>; DESTAQUE: <True|False>**. O nome normalizado deve usar as palavras com inicial maiúscula e espaçamento simples. A checagem de "FC" deve ignorar pontos e hífens. A checagem de "ATLÉTICO" deve funcionar mesmo sem acento (ex.: "Atletico"), assim como para "GRÊMIO".

Exemplo de entrada	**Exemplo de saída**
Grêmio Foot-Ball Porto Alegrense	NOME_NORMALIZADO: Grêmio Foot-Ball Porto Alegrense TEM_FC: False COMECA_COM_S: False TERMINA_COM_ENSE: True TEM_CLUBE: False TEM_ATLETICO: False NOME_CURTO: False NOME_VALIDO: False DESTAQUE: True
Santos FC	NOME_NORMALIZADO: Santos Fc TEM_FC: True COMECA_COM_S: True TERMINA_COM_ENSE: False TEM_CLUBE: False TEM_ATLETICO: False NOME_CURTO: True NOME_VALIDO: True DESTAQUE: False

Implemente um programa em **Python 3** que auxilia o professor Ricardo a formatar o relatório de correção de uma questão. O programa recebe o nome do aluno, o identificador numérico da questão, a pontuação máxima, a pontuação obtida e a nota de corte em percentual. O nome do aluno deve ser normalizado para maiúsculas com espaçamento simples (somente um espaço); o identificador da questão deve ser representado no padrão "Q" seguido de três dígitos (com zeros à esquerda quando necessário). O programa deve calcular o percentual alcançado dividindo a pontuação obtida pela pontuação máxima e multiplicando por 100. Além disso, deve imprimir um indicador booleano que informa se o percentual alcançado atinge a nota de corte.

ENTRADA

A entrada é composta por cinco linhas. A primeira linha contém o **nome do aluno** como texto livre. A segunda linha contém o **identificador numérico da questão**. A terceira linha contém **a pontuação máxima da questão**. A quarta linha contém a **pontuação obtida pelo aluno**. A quinta linha contém a **nota de corte em percentual**. Os valores numéricos devem ser informados em formato padrão (inteiros ou reais, conforme apropriado).

SAÍDA

A saída deve conter cinco linhas. A primeira linha deve apresentar "**ALUNO:** " seguido do nome do aluno em letras maiúsculas, com espaços normalizados. A segunda linha deve apresentar "**QUESTAO: Q**" seguido do identificador da questão com três dígitos. A terceira linha deve apresentar "**RESULTADO:** " seguido da pontuação obtida com duas casas decimais, uma barra, a pontuação máxima sem casas decimais e a palavra "**pontos**". A quarta linha deve apresentar "**PERCENTUAL:** " seguido do percentual alcançado com duas casas decimais e o símbolo de porcentagem. A quinta linha deve apresentar "**ATINIGIU_CORTE:** " seguida do valor booleano resultante da comparação do percentual com a nota de corte, sem utilizar estruturas de decisão.

Exemplo de entrada	Exemplo de saída
Ana Silva 7 10 8.5 60.0	ALUNO: ANA SILVA QUESTAO: Q007 RESULTADO: 8.50/10 pontos PERCENTUAL: 85.00% ATINIGIU_CORTE: True
felipe almeida 5 30 18.75 60.0	ALUNO: FELIPE ALMEIDA QUESTAO: Q005 RESULTADO: 18.75/30 pontos PERCENTUAL: 62.50% ATINIGIU_CORTE: True

Você foi contratado por uma empresa de Redes de Computadores da Restinga, para implementar um programa em **Python 3** que recebe, em uma única linha, os campos de um "pacote" de rede no formato **SRC_IP DST_IP PROTO TAMANHO_BYTES TTL FLAGS**. O protocolo informado deve ser normalizado para maiúsculas apenas para efeito de exibição e comparação. O resumo do tráfego deve ser apresentado como **<PROTO_UPPER> <SRC_IP> -> <DST_IP>**. Considere que a máscara de sub-rede para comparação é **/24**, então duas máquinas estão na mesma sub-rede quando os **três primeiros octetos dos IPs são idênticos**. O **endereço de broadcast** deve ser identificado como verdadeiro quando o destino for exatamente **255.255.255.255** ou quando o último octeto do IP de destino for **255**, pois isso caracteriza o **broadcast da sub-rede /24**. Um endereço de origem deve ser considerado privado quando estiver nas faixas **10.*.*.***, **192.168.*.*** ou **172.16.*.*** até **172.31.*.***. Um pacote é considerado grande quando seu tamanho em **bytes** for estritamente **maior que 1500**, e o **TTL** é considerado **baixo** quando for **menor ou igual a 10**. Um pacote TCP é considerado "**de controle**" quando, a **string de flags** contiver **SYN** ou **ACK**; para os demais protocolos, esse indicador deve ser **falso**.

ENTRADA

A entrada é composta por uma única linha contendo exatamente **seis valores** separados por espaço, na ordem: **IP de origem**, **IP de destino**, **protocolo**, **tamanho em bytes**, **TTL** e **flags**. O IP deve estar no formato decimal pontuado, o protocolo pode estar em qualquer combinação de maiúsculas e minúsculas, o tamanho e o TTL são números inteiros, e as flags são uma string sem espaços (por exemplo, SYN,ACK, NONE, PSH,ACK).

SAÍDA

A saída deve apresentar oito linhas. A primeira linha é um resumo no formato "**RESUMO: <PROTO> <SRC_IP> -> <DST_IP>**". A segunda linha deve exibir "**PROTO_NORMALIZADO: <PROTO>**". As linhas seguintes devem exibir, respectivamente, os valores booleanos dos indicadores "**MESMA_SUBREDE_24**", "**BROADCAST_DST_24**", "**SRC_PRIVADA**", "**PACOTE_GRANDE**", "**TTL_BAIXO**" e "**TCP_CONTROLE**".

Exemplo de entrada	Exemplo de saída
192.168.1.10 192.168.1.255 TCP 1514 64 SYN,ACK	RESUMO: TCP 192.168.1.10 -> 192.168.1.255 PROTO_NORMALIZADO: TCP MESMA_SUBREDE_24: True BROADCAST_DST_24: True SRC_PRIVADA: True PACOTE_GRANDE: True TTL_BAIXO: False TCP_CONTROLE: True
10.0.0.5 8.8.8.8 UDP 64 64 NONE	RESUMO: UDP 10.0.0.5 -> 8.8.8.8 PROTO_NORMALIZADO: UDP MESMA_SUBREDE_24: False BROADCAST_DST_24: False SRC_PRIVADA: True PACOTE_GRANDE: False TTL_BAIXO: False TCP_CONTROLE: False

Você foi contratado pela empresa de software IFCODE para auxiliar no desenvolvimento um programa para auxiliar os outros desenvolvedores na rápida verificação de elementos de uma mensagem de commit no padrão convencional. O programa em **Python 3** deve receber uma linha única contendo quatro campos separados por ponto e vírgula no formato **tipo;escopo;mensagem;versao**. O **tipo** representa a categoria do *commit* (por exemplo, feat, fix, docs, chore, test, refactor); o **escopo** é um identificador curto do módulo afetado; a **mensagem** é o título do *commit* e a **versão** é um texto no padrão semântico "**MAJOR.MINOR.PATCH**". O programa deve apresentar um resumo e uma série de indicadores booleanos calculados apenas com expressões lógicas/relacionais e métodos de *string*. O **tipo** é considerado válido quando corresponde a um dos valores mencionados; o **escopo** é considerado válido quando não está vazio, não possui espaços e contém apenas letras, dígitos, hífen ou sublinhado; o **tamanho da mensagem** é considerado adequado quando possui entre **10 e 72 caracteres**; a **versão** está em formato adequado quando contém exatamente dois pontos e as três partes são exclusivamente numéricas; um **identificador de issue** é considerado presente quando a mensagem contém **"#"** seguido de pelo menos um dígito; o **commit** é considerado "**breaking**" quando o tipo contém **"!"** ou a mensagem contém a expressão "**BREAKING CHANGE**" em caixa alta.

ENTRADA

A entrada consiste de uma única linha com **quatro** campos separados por **ponto e vírgula** no formato **tipo;escopo;mensagem;versao**. O **tipo** e o **escopo** são textos sem ponto e vírgula; a **mensagem** pode conter espaços; a **versão** é um texto com dígitos e pontos.

SAÍDA

A saída deve apresentar sete linhas, nesta ordem: uma linha de resumo no formato **RESUMO: <TIPO_EM_MAIÚSCULAS> <escopo> - <mensagem_original>**, seguida das linhas **TIPO_VALIDO: <True|False>**, **ESCOPO_VALIDO: <True|False>**, **MSG_TAMANHO_OK: <True|False>**, **SEMVER_FORMATO_OK: <True|False>**, **TEM_ISSUE_ID: <True|False>** e **BREAKING: <True|False>**.

Exemplo de entrada	Exemplo de saída
feat;auth;adiciona login social #123;1.2.0	RESUMO: FEAT auth - adiciona login social #123 TIPO_VALIDO: True ESCOPO_VALIDO: True MSG_TAMANHO_OK: True SEMVER_FORMATO_OK: True TEM_ISSUE_ID: True BREAKING: False
docs;readme;atualiza instruções de build;0.1.0	RESUMO: DOCS readme - atualiza instruções de build TIPO_VALIDO: True ESCOPO_VALIDO: True MSG_TAMANHO_OK: True SEMVER_FORMATO_OK: True TEM_ISSUE_ID: False BREAKING: False

Você foi contratado pela empresa de desenvolvimento Projetos de Software TingaWare, para implementar um programa para apoiar um desenvolvedor na elaboração rápida de um relatório de custos de *sprint*s. O programa em **Python 3** deve receber, em uma única linha e separados por ponto e vírgula, os campos **projeto;horas_planejadas;horas_executadas;valor_hora;orcamento**. Com esses dados, deve calcular o **custo estimado**, **o custo real**, o **desvio de horas** e o **desvio percentual**. O programa deve apresentar um **indicador booleano** informando se o custo real excede o orçamento e **outro indicador booleano** informando se o desvio percentual está no intervalo de **−10% a 10%**.

ENTRADA

A entrada consiste de uma única linha contendo todas as informações no formato **projeto; horas_planejadas;horas_executadas;valor_hora;orcamento**, onde **projeto** é um texto sem ponto e vírgula e os demais campos são valores reais.

SAÍDA

A saída deve apresentar sete linhas. A primeira linha é um resumo no formato **RESUMO: <PROJETO_EM_MAIÚSCULAS> H:<executadas>/<planejadas> C:R$ <custo_real> (R$ <valor_hora>/h)**. Em seguida, exiba as linhas **CUSTO_ESTIMADO: R$ <valor>, CUSTO_REAL: R$ <valor>, DESVIO_HORAS: <valor>, DESVIO_PERC: <valor>%, EXCEDEU_ORCAMENTO: <True|False>** e **TOLERANCIA_10: <True|False>**.

Exemplo de entrada	Exemplo de saída
Projeto A;40;38;120;5000	RESUMO: PROJETO A H:38.00/40.00 C:R$ 4560.00 (R$ 120.00/h) CUSTO_ESTIMADO: R$ 4800.00 CUSTO_REAL: R$ 4560.00 DESVIO_HORAS: -2.00 DESVIO_PERC: -5.00% EXCEDEU_ORCAMENTO: False TOLERANCIA_10: True
API V2;100;120;90;9500	RESUMO: API V2 H:120.00/100.00 C:R$ 10800.00 (R$ 90.00/h) CUSTO_ESTIMADO: R$ 9000.00 CUSTO_REAL: R$ 10800.00 DESVIO_HORAS: 20.00 DESVIO_PERC: 20.00% EXCEDEU_ORCAMENTO: True TOLERANCIA_10: False

Implemente um programa em **Python 3** para a empresa de software TINGA Security que valida uma senha criada por um usuário e gera um relatório de conformidade. O programa deve ler uma única linha com duas informações separadas por ponto e vírgula no formato **usuario;senha**. A validação considera os seguintes critérios: o **comprimento** deve estar entre **12 e 64 caracteres**; deve haver pelo menos **uma letra maiúscula** e pelo menos **uma letra minúscula**; deve haver pelo menos **um dígito**; deve haver pelo menos **um caractere especial** entre **! @ # $ % ^ & * ( ) - _ + = . ?** (conforme exatamente esse conjunto); **não deve haver espaços nem tabulações**; a senha **não deve conter o nome de usuário** (ignorando diferenças de maiúsculas e minúsculas); a senha **não deve conter a palavra tinga** (também ignorando caixa). O relatório deve apresentar, além de um resumo com o usuário e o comprimento da senha, um indicador booleano para **cada critério**, a conjunção booleana de todos os critérios ("**POLITICA_ATENDIDA**") e uma **pontuação de 0 a 8** somando os verdadeiros. Todos os cálculos devem ser expressões booleanas com métodos de string (por exemplo, upper, lower, replace, find, in, split) e operadores lógicos/relacionais.

ENTRADA

A entrada consiste de uma única linha no formato **usuario;senha**, onde usuario é um texto livre sem ponto e vírgula e senha é um texto arbitrário. O programa não realiza tratamento de exceções nem sanitização adicional além das regras descritas.

SAÍDA

A primeira deve ser **RESUMO: TINGA Security — Usuário: <USUARIO_EM_MAIUSCULAS> — Tamanho: <N>**. As linhas seguintes devem ser, exatamente nesta ordem: **COMPRIMENTO_OK: <True|False>, MAIUSCULA: <True|False>, MINUSCULA: <True|False>, DIGITO: <True|False>, ESPECIAL: <True|False>, SEM_ESPACOS: <True|False>, NAO_CONTEM_USUARIO: <True|False>, NAO_CONTEM_TINGA: <True|False>, POLITICA_ATENDIDA: <True|False>, PONTUACAO: <K>/8**, onde **<K>** é a soma dos oito indicadores de critérios.

Exemplo de entrada	Exemplo de saída
alice;Str0ng!Pass2025	RESUMO: TINGA Security — Usuário: ALICE — Tamanho: 15 COMPRIMENTO_OK: True MAIUSCULA: True MINUSCULA: True DIGITO: True ESPECIAL: True SEM_ESPACOS: True NAO_CONTEM_USUARIO: True NAO_CONTEM_TINGA: True POLITICA_ATENDIDA: True PONTUACAO: 8/8
bob;bob-1234-BOB-!@#	RESUMO: TINGA Security — Usuário: BOB — Tamanho: 16 COMPRIMENTO_OK: True MAIUSCULA: True MINUSCULA: True DIGITO: True ESPECIAL: True SEM_ESPACOS: True NAO_CONTEM_USUARIO: False NAO_CONTEM_TINGA: True POLITICA_ATENDIDA: False PONTUACAO: 7/8

Implemente um programa em Python 3 para a equipe TINGA Racing que recebe, em uma única linha, dados básicos de telemetria e logística de combustível de um carro e gera um relatório consolidado. O programa deve calcular, a partir da telemetria, a **coerência entre velocidade média** informada e a velocidade média derivada da distância da volta e do tempo de volta, considerando uma **tolerância de 5%.** Também deve estimar **o consumo por volta** (a partir do consumo em L/100 km), **a distância total do GP**, o **combustível total necessário** (com margem de segurança), o **déficit em relação ao combustível atual** e o **número de abastecimentos estimados** durante a prova, supondo reabastecimentos com um volume fixo por *pit stop*. Todas as saídas resultam de expressões com operadores lógicos/relacionais e operações de strings (por exemplo, split, upper, find, concatenação). Considere que a fórmula da velocidade média derivada é **distância_da_volta / (tempo_da_volta_em_horas)**, e que o consumo por volta é **(consumo_100km / 100) * distância_da_volta**. O combustível total necessário é o consumo por volta multiplicado pelo número de voltas do GP, acrescido da margem percentual. O déficit é tomado como zero quando o combustível atual excede a necessidade. O número de abastecimentos é o **teto de déficit / volume_por_pit**. Por fim, o programa também calcula **quantas voltas são possíveis com o combustível atual**, como a parte inteira de **combustível_atual / consumo_por_volta**.

ENTRADA

A entrada é uma única linha com onze campos separados por ponto e vírgula no formato: **carro;equipe;tempo_volta_s;vel_media_kmh;dist_volta_km;voltas_gp;tanque_cap_l;combust_atual_l;consumo_100km;abastecimento_por_pit_l;reserva_pct**. O campo **carro** é um identificador textual do carro, **equipe** é o nome da equipe, **tempo_volta_s** é um número real (segundos), **vel_media_kmh** é a velocidade média informada (km/h), **dist_volta_km** é a distância de uma volta (km), **voltas_gp** é inteiro, **tanque_cap_l** e **combust_atual_l** são volumes em litros, **consumo_100km** é o consumo em **L/100 km**, **abastecimento_por_pit_l** é o volume fixo reabastecido por *pit stop* e **reserva_pct** é a margem de segurança percentual adicionada ao total necessário.

SAÍDA

A saída consiste de dez linhas com os seguintes itens, exatamente nesta ordem: **RESUMO: <CARRO> — EQUIPE: <EQUIPE_EM_MAIUSCULAS>, LAP_TIME_S: <tempo_volta_formatado>, VEL_MEDIA_KMH: <vel_media_informada>, DISTANCIA_GP_KM: <dist_total>, CONSUMO_POR_VOLTA_L: <cons_por_volta>, COMBUSTIVEL_NECESSARIO_L: <necessario_total>, DEFICIT_L: <deficit_nao_negativo>, PITS_ESTIMADOS: <inteiro>, VOLTA_POSSIVEIS_TANQUE_ATUAL: <inteiro>, SPEED_COERENTE_5PCT: <True|False>**. Valores numéricos devem ser exibidos com até três casas decimais quando fizer sentido.

Exemplo de entrada	Exemplo de saída
R-77;DevRacing;102.3;215;6.0;52;120;120;50;35;5	RESUMO: R-77 — EQUIPE: DEVRACING LAP_TIME_S: 102.300 VEL_MEDIA_KMH: 215.000 DISTANCIA_GP_KM: 312.000 CONSUMO_POR_VOLTA_L: 3.000 COMBUSTIVEL_NECESSARIO_L: 163.800 DEFICIT_L: 43.800 PITS_ESTIMADOS: 2 VOLTA_POSSIVEIS_TANQUE_ATUAL: 40 SPEED_COERENTE_5PCT: True

Implemente um programa em **Python 3** para a empresa SofaScore que atribui, de forma automática, uma nota a um jogador de futebol a partir de uma linha com dados de telemetria da partida. A nota é construída a partir de uma **base fixa (valor de 3)** e de componentes normalizados entre 0 e 1, somados com pesos, além de uma penalização por erros. Considere as seguintes definições e normalizações: a **precisão de passes** é calculada como passes_certos / passes_totais e tem **peso 3**; a **taxa de duelos vencidos** é duelos_ganhos / duelos_totais e tem **peso 1**. Para esforço físico, normalize **sprints** por 30 (ou seja, sprints/30 limitado ao intervalo [0,1] com **peso 1**) e a **distância percorrida** dist_km por 12 km (dist_km/12 limitado a [0,1] com **peso 1**); para **finalizações**, normalize **chutes_no_alvo** por 5 (chutes_no_alvo/5 limitado a [0,1] com **peso 0.8**); para **ações defensivas**, normalize a soma **interceptacoes + desarmes** por 10 ((interceptacoes + desarmes)/10 limitado a [0,1] com **peso 1**). Além disso, aplique um pequeno bônus de **velocidade máxima** quando (vel_max_kmh − 28)/10 limitado a [0,1] com **peso 0.5** (não há bônus se a velocidade máxima for até 28 km/h)). Para **erros decisivos**, erros_decisivos limitado ao máximo de 3 (**valores acima de 3 não aumentam a penalidade**) com **peso -1.5**. De forma geral a nota bruta é então calculada por:

nota_bruta = 3.0 + 2.0*gols + 1.2*assistências + 3.0*precisão de passes + 1.0*taxa duelos vencidos + 1.0*taxa de sprints + 1.0*distância percorrida + 0.8*taxa de chutes no alvo + 1.0*taxas de ações defensivas + 0.5*bônus de velocidade − 1.5*erros decisivos

Por fim, aplique o corte para manter a nota no intervalo [0,10] (nota máxima deve ser 10). No relatório, exiba o **nome** e a **posição** (em maiúsculas), os **minutos**, **distância, sprints**, **velocidade máxima**, os agregados de **passe** e **duelos** com seus **percentuais**, os números **ofensivos** e **defensivos**, os **erros decisivos** e a **NOTA final formatada com duas casas decimais**.

ENTRADA

A entrada é composta por uma única linha no formato **nome;posicao;min_jogados;dist_km;sprints;vel_max_kmh;gols;assist;passes_certos;passes_totais;chutes_no_alvo;duelos_ganhos;duelos_totais;interceptacoes;desarmes;erros_decisivos**. Os campos numéricos podem ser reais e representam valores da partida corrente.

SAÍDA

A saída deve conter nove linhas de estatísticas e a linha final com a nota. A primeira linha deve ser **RESUMO: <NOME> — POS: <POS_EM_MAIUSCULAS>**. Em seguida, exiba **MINUTOS: <min>**, depois **DIST_KM: <dist> — SPRINTS: <sprints> — VEL_MAX_KMH: <vmax>**. Em seguida, mostre **PASSES: <certos>/<totais> (<precisao>%)** e **DUELOS: <ganhos>/<totais> (<taxa>%)**. Depois mostre **OFENSIVO: GOLS=<g> ASSIST=<a> CHUTES_NO_ALVO=<cna>**, seguido de **DEFENSIVO: INTERCEPTACOES=<i> DESARMES=<d>**, depois **ERROS_DECISIVOS: <e>**. Por fim, mostre **NOTA: <valor>/10**, com duas casas decimais.

Exemplo de entrada	Exemplo de saída
Arthur;MEI;90;11.2;28;32.5;1;1;65;80;3;8;12;3;5;0	RESUMO: Arthur - POS: MEI MINUTOS: 90 DIST_KM: 11.200 - SPRINTS: 28 - VEL_MAX_KMH: 32.5 PASSES: 65/80 (81.2%) DUELOS: 8/12 (66.7%) OFENSIVO: GOLS=1 ASSIST=1 CHU- TES_NO_ALVO=3 DEFENSIVO: INTERCEPTACOES=3 DESARMES=5 ERROS_DECISIVOS: 0 NOTA: 9.04/10

Implemente um programa em **Python 3** que recebe um único valor representando uma quantidade de bits e apresenta a conversão desse valor para diferentes unidades padrão de medição de dados. O programa deve ler uma linha com o número de bits (permitindo valores reais), tratar esse valor como base comum e exibir a conversão para as unidades em sistema decimal (SI) e binário (IEC). No sistema decimal (SI), apresente os equivalentes em bits e em bytes: para bits, utilize as escalas b, kb, mb, gb e tb, em que cada ordem de grandeza usa potências de 10 (por exemplo, 1 kb = 10^3 b); para bytes, apresente B, KB, MB, GB e TB, lembrando que 1 B = 8 b. No sistema binário (IEC), apresente também as grandezas em bytes KiB, MiB, GiB e TiB, onde cada passo utiliza potências de 1024 (por exemplo, 1 KiB = 1024 B). Todos os resultados devem ser calculados diretamente por expressões aritméticas, e os valores devem ser mostrados com até seis casas decimais, removendo zeros à direita quando não forem necessários.

ENTRADA

A entrada consiste de uma única linha contendo o número de bits, que pode ser informado como inteiro ou número real. O texto da unidade não é informado na entrada; considera-se, por definição, que o valor lido está em bits.

SAÍDA

A saída deve começar com a linha **ENTRADA: <valor> b**, mostrando o valor lido em bits formatado com até seis casas decimais, sem zeros finais desnecessários. Em seguida, devem ser apresentadas 14 linhas, cada uma com a etiqueta da unidade e o valor convertido, exatamente nesta ordem: **b, kb, mb, gb, tb, B, KB, MB, GB, TB, KiB, MiB, GiB, TiB**. Em todas as linhas, o número deve ter no máximo seis casas decimais e não exibir zeros desnecessários no final.

Exemplo de entrada	Exemplo de saída
8388608	ENTRADA: 8388608 b b: 8388608 kb: 8388.608 mb: 8.388608 gb: 0.008389 tb: 0.000008 B: 1048576 kB: 1048.576 MB: 1.048576 GB: 0.001049 TB: 0.000001 KiB: 1024 MiB: 1 GiB: 0.000977 TiB: 0.000001

Implemente um programa em **Python 3** para a Tinga Veículos que calcula o **salário de um vendedor com base em dados consolidados de suas vendas**. Considere que todos os veículos são **vendidos por 50% acima do preço de custo**. A **comissão do vendedor** é de **15% sobre o lucro total**. O programa deve ler uma única linha com as informações do vendedor e calcular: a **receita total**, o **lucro total**, a **comissão**, um **bônus de meta** (pago integralmente quando a quantidade vendida é maior ou igual à meta), um **bônus por satisfação do cliente** (CSAT) quando o **índice for maior ou igual a 90%,** o **salário bruto** e o **salário líquido**. Além disso, apresente o **ticket médio de venda**. Para tratar condições (como divisão por zero ou elegibilidade de bônus), utilize aritmética booleana, como por exemplo, multiplicar o valor por int(condicao) ou por (variavel != 0) * (1.0/variavel).

ENTRADA

A entrada consiste em uma única linha com nove campos separados por ponto e vírgula, no formato **vendedor;base_sal;unidades_vendidas;total_custo;meta_unidades;bonus_meta;csat_pct;bonus_csat;descontos**. O campo vendedor é um texto; os demais campos são numéricos. A **comissão** é sempre **15% sobre o lucro**, o **preço de venda total é sempre 50% acima do custo**, o bônus de meta é **aplicado integralmente quando unidades_vendidas >= meta_unidades**, e o **bônus de CSAT é aplicado integralmente quando csat_pct >= 90**.

SAÍDA

A saída deve conter um relatório com 14 linhas: resumo com o **nome do vendedor** em maiúsculas, **base salarial**, **unidades**, **meta** e o **indicador booleano de atingimento de meta**, **a receita total** (com a observação “50% sobre custo”), o **custo total**, o **lucro total**, a **comissão de 15% sobre o lucro**, os **bônus de meta e CSAT calculados**, os **descontos**, o **salário bruto** e o **líquido**, o **ticket médio** e o **indicador booleano de CSAT a partir de 90%**. Valores monetários devem ser apresentados com duas casas decimais.

Exemplo de entrada	Exemplo de saída
Ana Silva;3000;8;160000;6;500;92;400;200	RESUMO: VENDEDOR: ANA SILVA BASE: R$ 3000.00 UNIDADES: 8 META: 6 ATINGIU_META: True RECEITA_TOTAL: R$ 240000.00 (50% sobre custo) CUSTO_TOTAL: R$ 160000.00 LUCRO_TOTAL: R$ 80000.00 COMISSAO_15_SOBRE_LUCRO: R$ 12000.00 BONUS_META: R$ 500.00 BONUS_CSAT: R$ 400.00 DESCONTOS: R$ 200.00 SALARIO_BRUTO: R$ 45900.00 SALARIO_LIQUIDO: R$ 45700.00 TICKET_MEDIO: R$ 30000.00 CSAT_OK_90PCT: True

Implemente um programa em **Python 3** de apuração para uma urna eletrônica que recebe, em uma única linha, **os totais de votos de três candidatos**, além de **brancos** e **nulos**, e gera um **relatório da seção eleitoral**. O programa deve calcular o **total de votos**, a **participação** (em percentual sobre o número de eleitores), os **votos válidos**, a **distribuição percentual de cada candidato em relação aos votos válidos**, além dos **percentuais de brancos e nulos** em relação ao **total de votos**. O programa também deve informar a **votação de cada um dos cadidatos** (quantidade de votos) e o **percentual de cada candidato em relação aos votos válidos**, **maioria absoluta** (mais de 50% dos votos válidos), exibir o **nome do vencedor** e um indicador de consistência das contagens (**todos os números não negativos** e **a soma de válidos+brancos+nulos menor ou igual ao número de eleitores**). Qualquer condição (por exemplo, divisão por zero, ou verificação de empate) deve ser implementada por aritmética booleana e expressões (ex.: multiplicar por int(condicao) ou dividir por denominador + (denominador == 0)).

ENTRADA

A entrada é uma única linha com onze campos separados por **;** no formato: **zona;secao;eleitores; c1_nome;c1_votos;c2_nome;c2_votos;c3_nome;c3_votos;brancos;nulos**. Os campos **zona** e **secao** são textuais, **c1_nome**, **c2_nome** e **c3_nome** são os nomes dos candidatos, e os demais campos são numéricos.

SAÍDA

A saída consiste um resumo com **zona** e **seção**; o **número de eleitores**; **o total de votos e o turnout (%)**; os **válidos**, **brancos (%), nulos (%); três linhas com <nome>: <votos> votos (<percentual_de_válidos>%) para cada candidato**; a linha **MAIORIA_ABSOLUTA: <True|False>** (só é True quando há vencedor único com mais de 50% dos válidos); a linha **VENCEDOR_UNICO: <nome_ou_vazio>**; e a linha **CONSISTENTE: <True|False>**.

Exemplo de entrada	Exemplo de saída
101;A;1000;ALFA;420;BETA;380;GAMA;150;30;20	RESUMO: ZONA 101 — SECAO A ELEITORES: 1000 TOTAL_VOTOS: 1000 — TURNOUT_PCT: 100.00% VALIDOS: 950 — BRANCOS: 30 (3.00%) — NULOS: 20 (2.00%) ALFA: 420 votos (44.21%) BETA: 380 votos (40.00%) GAMA: 150 votos (15.79%) MAIORIA_ABSOLUTA: False VENCEDOR_UNICO: ALFA CONSISTENTE: True

Você foi contratado pela INVESTinga para desenvolver um programa em **Python 3** que projeta o **valor final de um investimento** com base em **aportes inicial e mensais**, **prazo em meses** e uma **taxa anual informada**, descontando uma **taxa anual de administração simples**. Para simplificar, considere a **taxa mensal líquida** como a diferença entre **taxa anual** e **taxa de administração dividida por 12**. O **valor futuro bruto** é calculado por capitalização composta clássica: soma do capital inicial capitalizado e da série de pagamentos mensais (**FV = Aporte_inicial * fator + aporte_mensal * serie,** sabendo que **serie = ((fator - 1) / taxa_mensal_líquida)** e que **fator = (1 + taxa_mensal_líquida)^{prazo}**). Some ao final um **cashback fixo**, que é um percentual do total aportado (aporte inicial mais todos os mensais). Exiba também o **total aportado**, os **juros ganhos** (**valor final menos total aportado**) e a **rentabilidade acumulada percentual** sobre o **total aportado**.

ENTRADA

A entrada é uma única linha com oito campos separados por **;** no formato **investidor;produto;aporte_inicial;aporte_mensal;meses;taxa_anual_pct;taxa_adm_anual_pct; cashback_pct**. Os dois primeiros campos são textos; os demais são numéricos. A taxa mensal líquida utilizada é calculada por **(taxa_anual_pct - taxa_adm_anual_pct) / 12 / 100**.

SAÍDA

O programa imprime nove linhas: um resumo com **nome do investidor** e **produto**; a **linha de aportes com os valores inicial**, **mensal** e **meses**; a **linha de taxas com anual**, **administração anual** e a **taxa mensal líquida calculada**; o **total aportado**; o **valor futuro bruto**; o **cashback**; o **valor final (bruto + cashback)**; os **juros ganhos** e a **rentabilidade acumulada sobre o total aportado (em %)**.

Exemplo de entrada	Exemplo de saída
Ana;CDB;10000;500;24;12;1.5;0.5	RESUMO: ANA — PRODUTO: CDB APORTES: INICIAL=R$ 10000.00 MENSAL=R$ 500.00 MESES=24 TAXAS: ANUAL=12.00% ADM_ANUAL=1.50% MEN- SAL_LIQ=0.8750% TOTAL_APORTADO: R$ 22000.00 FV_BRUTO: R$ 25614.19 CASHBACK: R$ 110.00 VALOR_FINAL: R$ 25724.19 JUROS_GANHOS: R$ 3724.19 RENT_ACUM_PCT: 16.93%
Carla;LCI;0;800;18;10;1.2;0.3	RESUMO: CARLA — PRODUTO: LCI APORTES: INICIAL=R$ 0.00 MENSAL=R$ 800.00 MESES=18 TAXAS: ANUAL=10.00% ADM_ANUAL=1.20% MEN- SAL_LIQ=0.7333% TOTAL_APORTADO: R$ 14400.00 FV_BRUTO: R$ 15333.69 CASHBACK: R$ 43.20 VALOR_FINAL: R$ 15376.89 JUROS_GANHOS: R$ 976.89 RENT_ACUM_PCT: 6.78%

Implemente um programa em **Python 3** que gere um relatório mensal de um hotel a partir de uma linha de dados. O relatório deve calcular a **taxa de ocupação do mês**, a **tarifa promocional** e a **tarifa final com ajustes automáticos**, a **receita mensal**, o **custo total**, o **RevPAR** (**receita por quarto disponível**), e o **resultado do mês** (**lucro ou prejuízo**). A **taxa de ocupação em percentual** é a razão entre diárias vendidas no mês e o número total de diárias disponíveis (quartos × dias), multiplicada por 100. Considere que a **tarifa promocional** aplica um desconto fixo informado, e que há **ajustes automáticos** sobre a tarifa final: aplica-se **um desconto extra de 10% quando a ocupação mensal for inferior a 40% e um acréscimo de 5% quando a ocupação mensal for superior a 85%**. A **receita** é o produto das diárias vendidas pela tarifa final. O **custo total do mês** é a soma do custo fixo mensal com o custo variável por diária multiplicado pelas diárias vendidas. O **RevPAR** é a receita dividida pelas diárias disponíveis. O **resultado do mês** é dado pela diferença entre a receita e o custo total.

ENTRADA

A entrada é **uma única linha** com campos separados por ; no formato **hotel;quartos_totais;dias; diarias_vendidas;tarifa_base;promocao_pct;custo_fixo_mensal;custo_var_por_diaria**. O campo hotel é textual; os demais são numéricos. A **tarifa promocional** é calculada como **tarifa_base × (1 − promocao_pct/100)**. Os ajustes automáticos são aplicados multiplicando a **tarifa promocional por (1 − 0,10)** quando a **ocupação for menor que 40%** e **por (1 + 0,05)** quando a ocupação for **maior que 85%.**

SAÍDA

O programa deve imprimir dez linhas: **um resumo com o nome do hotel**; o **estoque do mês** (**quartos, dias e diárias disponíveis**); a **ocupação em percentual**; as **três tarifas (base, promocional e final)**; os indicadores booleanos de aplicação de **desconto extra por baixa ocupação** e de **acréscimo por alta ocupação**; a **receita do mês**; o **custo total**; o **RevPAR**; o **tipo de resultado** (“LUCRO” ou “PREJUIZO”) e o **valor numérico do resultado**.

Exemplo de entrada	Exemplo de saída
Tinga Plaza;100;30;1800;300; 10;200000;50	RESUMO: HOTEL TINGA PLAZA ESTOQUE: QUARTOS=100 DIAS=30 DIARIAS_DISPONIVEIS=3000 OCUPACAO_PCT: 60.00% TARIFAS: BASE=R$ 300.00 PROMO=R$ 270.00 FINAL=R$ 270.00 AJUSTES_AUTO: LOW_OCC(-10%)=False HIGH_OCC(+5%)=False RECEITA: R$ 486000.00 CUSTO_TOTAL: R$ 290000.00 REVPAR: R$ 162.00 RESULTADO_TIPO: LUCRO LUCRO_PREJUIZO: R$ 196000.00

Implemente um programa em **Python 3** para o Tinga Bank que valide os dígitos verificadores (DV) de uma conta bancária no formato **NNNNNN-DD**, onde **NNNNNN** representa os **seis dígitos da conta** e **DD** os **dois dígitos verificadores**. O cálculo do **primeiro dígito (D1)** é feito pela **soma ponderada** dos **seis dígitos** com os **pesos 7, 6, 5, 4, 3 e 2**, nessa ordem, tomando o resultado **módulo 11**; **quando o resultado for 10, o dígito deve ser 0**. O **segundo dígito (D2)** é calculado pela **soma ponderada dos seis dígitos originais** e do **dígito D1** com os **pesos 8, 7, 6, 5, 4, 3 e 2**, nesta ordem, tomando o **resultado módulo 11**; quando **o resultado for 10, o dígito deve ser 0**. O programa deve imprimir um relatório que inclui o **nome do banco** e a **agência**, a **conta informada**, os detalhes do **cálculo** de **D1** e **D2** (**soma, módulo e dígito**), o **DV esperado**, o **DV informado**, se o **DV confere** e se a **conta é considerada válida**.

ENTRADA

A entrada é uma única linha contendo três campos separados por ponto e vírgula no formato **banco;agencia;conta**, onde **banco** é um texto livre, **agencia** é um texto curto e **conta** segue o padrão **NNNNNN-DD**. O programa deve usar exatamente esses valores para produzir o relatório, calculando os dígitos esperados a partir dos seis dígitos da conta e comparando com os dois dígitos informados após o **hífen**.

SAÍDA

A saída consiste em oito linhas: um resumo com o **nome do banco** em maiúsculas e a **agência**; a **conta** no formato **NNNNNN-DD**; uma linha com **os pesos, a soma, o módulo e o dígito D1**; uma linha com os **pesos, a soma, o módulo e o dígito D2**; a linha com o **DV esperado**; a linha com o **DV informado**; a linha **booleana** indicando se o **DV está correto**; e a linha **booleana** indicando se a conta é **considerada válida**.

Exemplo de entrada	Exemplo de saída
Tinga Bank;0001;123456-00	RESUMO: BANCO TINGA BANK — AGENCIA 0001 CONTA: 123456-00 PESOS_D1: 7,6,5,4,3,2 SOMA=77 MOD11=0 D1=0 PESOS_D2: 8,7,6,5,4,3,2(D1) SOMA=98 MOD11=10 D2=0 DV_ESPERADO: 00 DV_INFORMADO: 00 DV_OK: True CONTA_VALIDA: True
Tinga Bank;1020;654321-00	RESUMO: BANCO TINGA BANK — AGENCIA 1020 CONTA: 654321-00 PESOS_D1: 7,6,5,4,3,2 SOMA=112 MOD11=2 D1=2 PESOS_D2: 8,7,6,5,4,3,2(D1) SOMA=137 MOD11=5 D2=5 DV_ESPERADO: 25 DV_INFORMADO: 00 DV_OK: False CONTA_VALIDA: False