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

Escreva um programa em **Python 3** que leia um número inteiro que representa um código de DDD para discagem interurbana. Em seguida, informe à qual cidade o DDD pertence, considerando a tabela abaixo:

DDD	Destino
51	Porto Alegre
61	Salvador
11	São Paulo
21	Rio de Janeiro
31	Belo Horizonte
54	Caxias do Sul
55	Santa Maria
53	Pelotas

OBS: Se a entrada for qualquer outro DDD que não esteja presente na tabela acima, o programa deverá informar: DDD não cadastrado

ENTRADA

A entrada consiste de um único valor inteiro.

SAÍDA

Imprima o nome da cidade correspondente ao DDD existente na entrada. Imprima *DDD não cadastrado* caso não existir DDD correspondente ao número digitado.

Exemplo de entrada	Exemplo de saída
11	São Paulo
51	Porto Alegre
55	Santa Maria
99	DDD não cadastrado

O posto de combustíveis “Postinga” está vendendo combustíveis com a seguinte tabela de descontos:

- Álcool
  - até 20 litros (inclusive), desconto de 5,5% por litro
  - acima de 20 litros, desconto de 8% por litro
- Gasolina
  - até 20 litros (inclusive), desconto de 3,5% por litro
  - acima de 20 litros, desconto de 10% por litro

Faça um programa em **Python 3** que leia o tipo de combustível (codificado da seguinte forma: A - Álcool, G - Gasolina) e o número de litros vendidos, calcule e mostre o valor a ser pago pelo cliente sabendo-se que o preço do litro da gasolina é R$ 6,10 o preço do litro do álcool é R$ 5,75.

Importante: caso seja informado um combustível inválido deve ser informada a frase “CÓDIGO INVÁLIDO”. Além disso, caso seja informada uma quantidade de combustível inferior ou igual a 0, deverá ser informada a frase “QUANTIDADE INVÁLIDA”. Caso, ambos os valores estejam errados exibir as duas mensagens.

ENTRADA

A primeira linha da entrada contém um caractere que identificará o tipo de combustível. A segunda linha contém a quantidade de litros vendidos.

SAÍDA

A saída deverá conter uma frase informando o valor a ser pago pelo cliente precedido pela seguinte expressão “Valor a pagar: R$” (sem aspas)

Exemplo de entrada	Exemplo de saída
G 100.11	Valor a pagar: R$ 549.60
A 10	Valor a pagar: R$ 54.34
F 1	CÓDIGO INVÁLIDO
G -100	QUANTIDADE INVÁLIDA
F -1	CÓDIGO INVÁLIDO QUANTIDADE INVÁLIDA

Leia 5 valores Inteiros. A seguir mostre quantos valores digitados foram pares, quantos valores digitados foram ímpares, quantos valores digitados foram positivos e quantos valores digitados foram negativos.

ENTRADA

O arquivo de entrada contém 5 valores inteiros quaisquer.

SAÍDA

Imprima a mensagem conforme o exemplo fornecido, uma mensagem por linha, não esquecendo o final de linha após cada uma.

Exemplo de entrada	**Exemplo de saída**
-5 0 -3 -4 12	3 valor(es) par(es) 2 valor(es) ímpar(es) 1 valor(es) positivo(s) 3 valor(es) negativo(s)

Leia 5 valores Inteiros. Ordene os valores e exiba os tanto na ordem crescente quanto na ordem decrescente.

ENTRADA

O arquivo de entrada contém 5 valores inteiros quaisquer.

SAÍDA

Imprima os valores de forma ordenada conforme o exemplo fornecido (separados por vírgula), uma mensagem por linha, não esquecendo o final de linha após cada uma. Na primeira linha, a ordem apresentada deve ser a crescente, enquanto que na segunda linha a ordem de apresentação deve ser a decrescente.

Exemplo de entrada	**Exemplo de saída**
-5 0 -3 -4 12	-5, -4, -3, 0, 12 12, 0, -3, -4, -5
89 9 -1 0 0	-1, 0, 0, 9, 89 89, 9, 0, 0, -1

Paulo e Pedro fizeram uma longa jornada desde que partiram do Brasil para competir na Final Mundial da Maratona, em Phuket, Tailândia. Notaram que a cada escala que faziam, tinham que ajustar seus relógios por causa do fuso horário.

Assim, para melhor se organizarem para as próximas viagens, eles pediram que você faça um aplicativo para celular que, dada a hora de saída, tempo de viagem e o fuso do destino com relação à origem, você informe a hora de chegada de cada vôo no destino.

Por exemplo, se eles partiram às 10 horas da manhã para uma viagem de 4 horas rumo a um destino que fica à leste, em um fuso horário com uma hora a mais com relação ao fuso horário do ponto de partida, a hora de chegada terá que ser: 10 horas + 4 horas de viagem + 1 hora de deslocamento pelo fuso, ou seja, chegarão às 15 horas. Note que se a hora calculada for igual a 24, seu programa deverá imprimir 0 (zero).

ENTRADA

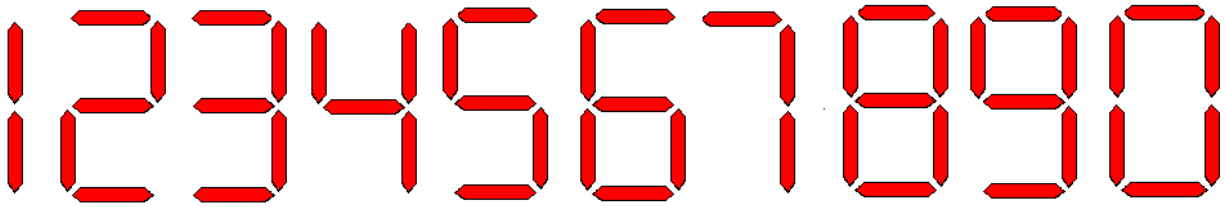
A entrada contém 3 inteiros: **S** ($0 \leq$ **S** ≤ 23), **T** ($1 \leq$ **T** ≤ 12) e **F** ($-5 \leq$ **F** ≤ 5), um por linha, indicando respectivamente a hora da saída, o tempo de viagem e o fuso horário do destino com relação à origem.

SAÍDA

Imprima um inteiro que indica a hora local prevista no destino, conforme os exemplos abaixo.

Exemplo de entrada	**Exemplo de saída**
10 7 3	20
22 6 -2	2
0 3 -4	23

Roben quer montar um painel de leds contendo diversos números para o ROBOLAB. Ele não possui muitos leds, e não tem certeza se conseguirá montar o número desejado. Considerando a configuração dos leds dos números abaixo, faça um algoritmo que ajude Roben a descobrir a quantidade de leds necessário para montar o valor.



Entrada

A entrada contém um número **de dois dígitos [00, 99]** correspondente ao valor que Roben quer montar com os leds.

Saída

Imprima uma linha contendo o número de leds que João precisa para montar o valor desejado, seguido da palavra "leds".

Exemplo de entrada	**Exemplo de saída**
14	6 leds
28	12 leds
09	12 leds

Para participar da categoria OURO do 1º Campeonato Mundial de bolinha de Gude, o jogador deve pesar entre 70 Kg (inclusive) e 80 Kg (inclusive) e medir de 1,75 m (inclusive) a 1,90 m (inclusive). Faça um programa para ler a altura e o peso de um jogador e determine se o jogador está apto a participar do campeonato escrevendo uma das seguintes mensagens conforme cada situação.

- RECUSADO POR ALTURA - se somente a altura do jogador for inválida;
- RECUSADO POR PESO - se somente o peso do jogador for inválido;
- TOTALMENTE RECUSADO - se a altura e o peso do jogador forem inválidos;
- ACEITO - se a altura e o peso do jogador estiverem dentro da faixa especificada.

Entrada

A entrada consiste de duas linhas. A primeira apresentará o peso do jogador. A segunda linha apresentará a altura do jogador. Ambos os valores informados serão de ponto flutuante.

Saída

Imprima a mensagem de acordo com a especificação acima. Caso o peso ou a altura forem inválidos, informar “PESO INVÁLIDO” (caso o valor esteja fora do intervalo [20, 200]) ou “ALTURA INVÁLIDA” (caso o valor esteja fora do intervalo [0.35, 3.00]). Se ambos os valores forem inválidos informar “VALORES INVÁLIDOS”.

Exemplo de entrada	**Exemplo de saída**
50.2 1.90	RECUSADO POR PESO
70.0 1.75	ACEITO
76.5 1.11	RECUSADO POR ALTURA
45.40 -1.1	ALTURA INVÁLIDA
-9.43 110.4	VALORES INVÁLIDOS

O professor Rafael é um grande admirador de calculadoras. No entanto, ele esqueceu de todas as suas 10 calculadoras em casa. Assim, ele solicitou para que você desenvolva uma calculadora com as operações abaixo relacionadas. Como o teclado de Rafael está com um pequeno problema e algumas teclas não funcionam, você deve considerar dois símbolos diferentes para cada operação:

- Adição: + a
- Subtração: - s
- Multiplicação: * . m
- Divisão: / d
- Potência: ** p
- Módulo: % r

Note que seu programa não deve diferenciar letras maiúsculas, pois se tratam do mesmo símbolo.

Entrada

A entrada consiste de três linhas. A primeira apresentará o primeiro termo da operação. A segunda linha apresentará o símbolo da operação. A terceira linha conterá o segundo termo da operação.

Saída

Você deverá exibir o resultado da operação com no máximo duas casas decimais. Note que no caso da operação causar um erro você deve imprimir a mensagem ERROR.

Exemplo de entrada	**Exemplo de saída**
1 A 1	2.00
48.13 s 2.00	46.13
4 R 2	0
4 / 0	ERROR
2 P 2	4

A ECR (Editora de Crônicas da Restinga) é muito tradicional quando se trata de numerar as páginas de seus livros. Ela sempre usa a numeração romana para isso. E seus livros nunca ultrapassam as 100 páginas pois, quando necessário, dividem o livro em volumes.

Você deve escrever um programa que, dado um número arábico, mostra seu equivalente na numeração romana.

Lembre que I representa 1, V é 5, X é 10, L é 50, por fim, C representa o valor 100.

Entrada

A entrada é um número inteiro positivo **N** (0 < **N** < 101), podendo ser expresso no formato decimal ou romano

Saída

A saída é o número **N** escrito na numeração inversa do dado de entrada. Se o valor de entrada for informado em decimal você deverá informar no formato romano. Caso o valor de entrada for informado em romano, você deverá informar em decimal. Use sempre letras maiúsculas.

Exemplo de entrada	Exemplo de saída
66	LXVI
83	LXXXIII
C	100
XCIX	99
1	I

Faça um programa em Python 3 que receba o nome de usuário e uma possível senha. O retorno deverá conter a informação se o usuário e senha são válidos, exibindo-os como no exemplo de saída. Considere, usuários válidos como sendo aqueles diferente da senha (não diferenciar maiúsculas e minúsculas). Além disso, você deverá verificar se a senha é válida, considerando os seguintes critérios:

- Exatamente 6 caracteres
- Pelo menos um número
- Não conter espaços
- Possuir pelo menos um símbolo (,), {,}, ., ou _

ENTRADA

Haverá uma única linha na entrada que conterá o nome do usuário e a senha (separados por um espaço simples).

SAÍDA

A saída deverá conter uma frase informando se o usuário é válido seguido da senha conforme o formato abaixo.

Exemplo de entrada	Exemplo de saída
aluno (abc1)	Usuário válido. USER: aluno PASSWORD: (abc1)
boss senha	Usuário válido. Senha inválida
Aluno aluno	Usuário inválido. Senha inválida
Aluno alun1.	Usuário válido. USER: Aluno PASSWORD: alun1.
241a_. senha_	Usuário válido. Senha inválida
admin _4dmin	Usuário válido. USER: admin PASS: _4dmin
root {senhaforte42}	Usuário válido. Senha inválida

Leia um valor de ponto flutuante com duas casas decimais. Este valor representa um valor monetário. A seguir, calcule o menor número de notas e moedas possíveis no qual o valor pode ser decomposto. As notas consideradas são de 100, 50, 20, 10, 5, 2. As moedas possíveis são de 1, 0.50, 0.25, 0.10, 0.05 e 0.01. A seguir mostre a relação de notas necessárias.

ENTRADA

O arquivo de entrada contém um valor de ponto flutuante **N** ($0 \leq N \leq 1000000.00$).

SAÍDA

Imprima a quantidade mínima de notas e moedas necessárias para trocar o valor inicial, conforme exemplo fornecido.

Obs: Utilize ponto (.) para separar a parte decimal.

Exemplo de entrada	Exemplo de saída
576.73	NOTAS: 5 nota(s) de R$ 100.00 1 nota(s) de R$ 50.00 1 nota(s) de R$ 20.00 0 nota(s) de R$ 10.00 1 nota(s) de R$ 5.00 0 nota(s) de R$ 2.00 MOEDAS: 1 moeda(s) de R$ 1.00 1 moeda(s) de R$ 0.50 0 moeda(s) de R$ 0.25 2 moeda(s) de R$ 0.10 0 moeda(s) de R$ 0.05 3 moeda(s) de R$ 0.01
91.01	NOTAS: 0 nota(s) de R$ 100.00 1 nota(s) de R$ 50.00 2 nota(s) de R$ 20.00 0 nota(s) de R$ 10.00 0 nota(s) de R$ 5.00 0 nota(s) de R$ 2.00 MOEDAS: 1 moeda(s) de R$ 1.00 0 moeda(s) de R$ 0.50 0 moeda(s) de R$ 0.25 0 moeda(s) de R$ 0.10 0 moeda(s) de R$ 0.05 1 moeda(s) de R$ 0.01

Implemente um programa em Python 3 que, a partir de uma única linha de dados sobre um processo do sistema operacional, calcule um índice de carga e classifique a prioridade e a ação sugerida. O índice de carga deve ser calculado como uma média ponderada das utilizações de CPU e memória, na forma **LOAD_INDEX = 0,6×CPU(%) + 0,4×MEM(%)**. Traduza o estado do processo: R para "Executando", S para "Dormindo", D para "I/O", Z para "Zumbi", T para "Parado" e I para "Inativo"; qualquer outro valor pode ser tratado como "Desconhecido". Com base no valor de **nice** e do **LOAD_INDEX**, classifique a prioridade do processo conforme as seguintes regras: prioridade **ALTA** se **nice ≤ -5 ou LOAD_INDEX ≥ 80**; prioridade **MEDIA** se **nice ≤ 5 ou LOAD_INDEX ≥ 50** (e não for ALTA); caso contrário, **BAIXA**. Em seguida, defina a ação sugerida: se o estado for Z, a ação é **FINALIZAR**; senão, se o estado for D **e** LOAD_INDEX ≥ 70, a ação é **INVESTIGAR_DISCO**; senão, se o estado for R **e** a prioridade for **ALTA**, a ação é **MONITORAR**; nos demais casos, a ação é **MANTER**. O relatório deve exibir resumo, métricas, estado, prioridade, sugestão de ação e duas flags booleanas: CRITICO (verdadeiro se LOAD_INDEX ≥ 90) e ZUMBI (verdadeiro se estado for Z).

ENTRADA

A entrada é uma única linha no formato **usuario;processo;nice;cpu_pct;mem_pct;estado**, onde usuario e processo são textos, nice é inteiro, cpu_pct e mem_pct são números (que podem conter decimais) e estado é um dos códigos citados.

SAÍDA

A saída deve conter seis linhas: um resumo com **usuário** e processo; a linha de métricas com **nice**, **CPU**, **memória** e **LOAD_INDEX** (com duas casas decimais); a linha de estado com a **descrição** do estado; a **linha de prioridade**; a **linha de sugestão**; e a linha de **flags** com **CRITICO** e **ZUMBI**.

Exemplo de entrada	Exemplo de saída
ricardo;nginx;-8;70;30;R	RESUMO: USUARIO=RICARDO PROCESSO=nginx METRICAS: NICE=-8 CPU=70.00% MEM=30.00% LOAD_INDEX=54.00 ESTADO: Executando PRIORIDADE: ALTA SUGESTAO: MONITORAR FLAGS: CRITICO=False ZUMBI=False
maria;db;-3;85;70;D	RESUMO: USUARIO=MARIA PROCESSO=db METRICAS: NICE=-3 CPU=85.00% MEM=70.00% LOAD_INDEX=79.00 ESTADO: I/O PRIORIDADE: MEDIA SUGESTAO: INVESTIGAR_DISCO FLAGS: CRITICO=False ZUMBI=False

Implemente um programa em Python 3 que avalia o uso de disco do usuário e calcula uma sugestão de limpeza, sem varrer pastas nem listar arquivos. A entrada informa o **espaço total** e **usado**, o **tipo predominante de dados** e a **política de limpeza**. Calcule o percentual de utilização e o espaço livre**.** Obtenha um **peso** (fator base) por tipo de dados: **logs → 0,15**, **temp → 0,20**, **media → 0,10**, **docs → 0,05**, **vm → 0,08**. Além disso, seu programa deve definir um **multiplicador** por política: **STRICT → 1,5, NORMAL → 1,0** e **FLEX → 0,5**. A sugestão de limpeza em gigabytes é **SUGESTAO = usado × peso × multiplicador**. Classifique o status do disco por: **CRITICO** se USO_% ≥ 90, **ALTO** se USO_% ≥ 75 (e < 90), **MODERADO** se USO_% ≥ 50 (e < 75) e **OK** caso contrário. A flag NECESSITA_ACAO deve ser verdadeira se **USO_% ≥ 75** ou se **LIVRE < 10%** do total. Imprima um relatório com **resumo, disco, tipo/política, sugestão, status e flags**, formatando números com duas casas decimais.

ENTRADA

A entrada é uma linha no formato **usuario;total_gb;usado_gb;tipo;politica**, onde usuario é texto, total_gb e usado_gb são números em gigabytes (e do tipo com ponto flutuante), tipo é um dos valores indicados e politica é **STRICT**, **NORMAL** ou **FLEX**.

SAÍDA

A saída deve conter **seis linhas**: um resumo com o **usuário**; os **dados de disco** com **total**, **usado**, **livre** e **percentual de uso**; a linha com **tipo de dados** e **política**, mostrando **peso** e **multiplicador**; a **sugestão de limpeza em GB**; o **status**; e as **flags** com **NECESSITA_ACAO**.

Exemplo de entrada	Exemplo de saída
alice;500;420;logs;STRICT	RESUMO: USUARIO=ALICE DISCO: TOTAL=500.00GB USADO=420.00GB LIVRE=80.00GB USO=84.00% TIPO_DADOS: logs PESO=0.15 POLITICA=STRICT MULT=1.50 SUGESTAO_LIMPEZA_GB: 94.50 STATUS: ALTO FLAGS: NECESSITA_ACAO=True
bob;256;100;media;FLEX	RESUMO: USUARIO=BOB DISCO: TOTAL=256.00GB USADO=100.00GB LIVRE=156.00GB USO=39.06% TIPO_DADOS: media PESO=0.10 POLITICA=FLEX MULT=0.50 SUGESTAO_LIMPEZA_GB: 5.00 STATUS: OK FLAGS: NECESSITA_ACAO=False

Implemente um programa em Python 3 que calcula o preço final de um jogo vendido pela TingaGames a partir de uma única linha de dados. A linha contém o **nome do jogo** (com sublinhados no lugar dos espaços), o **usuário comprador** e quatro números: **preço base**, **taxa de plataforma (%)**, **imposto (%)**, e **desconto promocional (%)**. Todos os números chegam no formato brasileiro com vírgula como separador decimal. Além disso, apresente o nome do jogo com espaços no lugar dos sublinhados e em maiúsculas. O cálculo deve ocorrer somente se os dados forem válidos, obedecendo às regras: o **preço base** deve ser **maior ou igual a zero**; **taxa de plataforma** e **imposto** devem ser **maiores ou iguais a zero**; o **desconto** deve estar no intervalo **[0, 100]**. Caso **qualquer** dessas regras seja violada, o relatório deve indicar validação **OK=False** e um **MOTIVO** específico, e os campos monetários calculados devem sair como 0.00, a **categoria** deve ser **N/A** e as flags devem ser **False**. Se os dados forem válidos, calcule o valor buto (incluindo os impostos sobre os valor base) e o valor final (com desconto sobre o bruto). Classifique a **CATEGORIA** do jogo por decisão: **PREMIUM** se FINAL ≥ 300.00, **PADRAO** se FINAL ≥ 100.00 (e < 300.00), e **ECONOMICO** caso contrário. Exiba ainda duas flags booleanas: **DESCONTO_APLICADO** (verdadeiro se desconto > 0) e **ALERTA_ALTA_TAXA** (verdadeiro se **qualquer** entre taxa de plataforma, imposto ou desconto **exceder 50%**).

ENTRADA

A entrada é uma única linha com **campos separados por espaço** no formato: **jogo_com_sublinhados usuario preco_base taxa_plataforma_pct imposto_pct desconto_pct**.

SAÍDA

O programa imprime **nove linhas**: um **resumo** com jogo (em maiúsculas e com espaços) e **usuário**; a linha da **base**; a linha das **taxas**; a linha do **desconto**; a linha de **validação com OK e MOTIVO**; o **preço bruto**; o **preço final**; a **categoria**; e as **flags DESCONTO_APLICADO** e **ALERTA_ALTA_TAXA**. Em entradas inválidas conforme as regras, **OK=False**, **MOTIVO específico**, **BRUTO=0.00**, **FINAL=0.00**, **CATEGORIA=N/A**, **DESCONTO_APLICADO=False** e **ALERTA_ALTA_TAXA=False**.

Exemplo de entrada	Exemplo de saída
Diablo_IV ricardo 59,90 12,5 7,5 5,0	RESUMO: JOGO=DIABLO IV USUARIO=ricardo BASE: R$ 59.90 TAXAS: PLATAFORMA=12.50% IMPOSTO=7.50% DESCONTO: 5.00% VALIDACAO: OK=True MOTIVO=- BRUTO: R$ 71.88 FINAL: R$ 68.29 CATEGORIA: ECONOMICO FLAGS: DESCONTO_APLICADO=True ALERTA_ALTA_TAXA=False
Battle_Arena bruno 149,99 10,0 0,0 15,0	RESUMO: JOGO=BATTLE ARENA USUARIO=bruno BASE: R$ 149.99 TAXAS: PLATAFORMA=10.00% IMPOSTO=0.00% DESCONTO: 15.00% VALIDACAO: OK=True MOTIVO=- BRUTO: R$ 164.99 FINAL: R$ 140.24 CATEGORIA: PADRAO FLAGS: DESCONTO_APLICADO=True ALERTA_ALTA_TAXA=False

Implemente um programa em Python 3 que, a partir de uma única linha descrevendo um evento de segurança capturado pela equipe TingaSec, classifique a **AÇÃO** a ser tomada e gere um pequeno relatório. O evento contém as seguintes informações em sequência: o **nome de usuário**, o **e-mail corporativo (ou externo)**, o **endereço IP de origem**, o **país de origem** (código ISO-2 como "BR", "US", "PT"), o **sistema do dispositivo** (WIN, LIN, MAC), o **tipo de evento** (LOGIN, DOWNLOAD ou UPLOAD) e o **recurso alvo** (por exemplo, um nome de arquivo ou uma URL simplificada). O programa deve avaliar condições típicas de segurança como **origem de rede interna** (endereços iniciados por "10." ou "192.168."), **domínio do e-mail corporativo** (endereços terminados em "@tingasec.com.br"), e extensões de arquivos potencialmente perigosas (como ".exe", ".js", ".bat"). Ao final, a saída deve apresentar um resumo normalizado do evento (nome do usuário e e-mail em minúsculas e sem espaços supérfluos), a origem e o tipo, o alvo, e a **AÇÃO** entre: **QUARENTENA**, **BLOQUEAR**, **MONITORAR** ou **PERMITIR**, de acordo com as regras de decisão.

ENTRADA

A entrada é composta por **uma única linha** com **campos separados por espaço** no formato: **usuario email ip pais sistema evento recurso**. Todos os campos não devem conter espaços internos.

SAÍDA

O programa deve imprimir **cinco linhas**: a primeira contendo um **RESUMO** com **usuario** e **email** em minúsculas; a segunda descrevendo a **ORIGEM** (IP e país) e o **SISTEMA**; a terceira descrevendo o **TIPO** do evento; a quarta descrevendo o **RECURSO** (normalizado em minúsculas); e a quinta contendo **ACAO: …** com a decisão final.

Exemplo de entrada	Exemplo de saída
maria MARIA@TINGASEC.COM.BR 10.0.0.12 BR WIN LOGIN portal.tingasec.com.br	RESUMO: usuario=maria email=maria@tingasec.com.br ORIGEM: ip=10.0.0.12 pais=BR sistema=WIN TIPO: LOGIN RECURSO: portal.tingaesc.com.br ACAO: PERMITIR
joao joao@gmail.com 203.0.113.5 US LIN DOWNLOAD setup.EXE	RESUMO: usuario=joao email=joao@gmail.com ORIGEM: ip=203.0.113.5 pais=US sistema=LIN TIPO: DOWNLOAD RECURSO: setup.exe ACAO: BLOQUEAR

Você foi contrado pela TingaChain, empresa de criptomoedas da Restinga para implementar um programa em Python 3 que, a partir de uma única linha com dados de um envio de, valide o endereço de destino conforme a rede escolhida, verifique a consistência numérica dos parâmetros, calcule a taxa total e a taxa efetiva, classifique a prioridade de envio e produza uma recomendação operacional. A linha de entrada deve conter, separados por espaço, os campos: **identificador da carteira, endereço de destino, valor em BTC, taxa por vbyte em sat/vB, tamanho da transação em vbytes, carga atual do mempool em porcentagem e a rede** (MAINNET ou TESTNET). Os números podem vir com vírgula como separador decimal e devem ser interpretados como valores reais. Para o relatório, normalize a carteira para maiúsculas e o endereço para minúsculas.

A validação numérica exige: **valor em BTC estritamente maior que 0**; **taxa por vbyte em sat/vB maior ou igual a 1**; **tamanho da transação em vbytes estritamente maior que 0**; **carga do mempool entre 0 e 100 inclusive**. A validação do endereço depende da rede: para **MAINNET**, o endereço é considerado compatível quando **começa com bc1, 1 ou 3**; para **TESTNET**, quando **começa com tb1, m, n ou 2**. Se a rede não for uma dessas ou o prefixo não corresponder, o endereço é inválido. A taxa total em BTC é calculada como **taxa_total_btc** = (taxa sat/vB × tamanho em vbytes) / 100_000_000, e a taxa efetiva percentual é **taxa_efetiva_pct** = (taxa_total_btc / valor_btc) × 100.

A classificação de prioridade usa a **carga do mempool** e/ou a **taxa por vbyte**, seguindo as regras:

- URGENTE - (mempool_pct ≥ 80) ou (taxa sat/vB ≥ 50)
- ALTA - (não for URGENTE) e [(mempool_pct ≥ 60) ou (taxa sat/vB ≥ 30)]
- MEDIA - (não for ALTA) e [(mempool_pct ≥ 30) ou (taxa sat/vB ≥ 15)]
- BAIXA - demais casos

A recomendação final segue estas regras: se qualquer validação falhar (números ou endereço/rede incompatíveis), a recomendação é **REVISAR**. Caso seja válida, se a prioridade for **URGENTE** e o valor for **maior ou igual a 1.0 BTC**, recomenda-se **ENVIAR_AGORA**; se a prioridade for **BAIXA** e o valor for **menor que 0.001 BTC**, recomenda-se **AGRUPAR_UTXOS** (adiar e consolidar entradas menores); nos demais casos, recomenda-se **AGUARDAR_JANELA** (aguardar momento de taxa mais favorável).

ENTRADA

A entrada é **uma única linha** com **campos separados por espaço** no formato:
carteira endereco valor_btc fee_rate_sat_vb tamanho_vb mempool_pct rede
Os números devem usar vírgula como separador decimal.

SAÍDA

A saída deve conter sete linhas: um **RESUMO** com carteira e rede; a **DESTINACAO** com o endereço normalizado; uma linha **VALORES** com o valor em **BTC** e a **taxa por vbyte**; uma linha **TAXA** com o **tamanho**, a **taxa total em BTC** e a **taxa efetiva sobre o valor**; uma linha **PRIORIDADE**; uma linha **VALIDACAO** indicando se o endereço é compatível com a rede e se os números são válidos; e uma linha **RECOMENDACAO** com a ação sugerida.

Exemplo de entrada	Exemplo de saída
wallet01 bc1qwxyz... 0,015 35 200 82 MAINNET	RESUMO: carteira=WALLET01 rede=MAINNET DESTINACAO: endereco=bc1qwxyz... VALORES: valor_btc=0.01500000 fee_rate_sat_vb=35.00 TAXA: tamanho_vb=200 taxa_total_btc=0.00007000 taxa_efetiva_pct=0.47 PRIORIDADE: URGENTE VALIDACAO: endereco_ok=True numeros_ok=True RECOMENDACAO: ENVIAR_AGORA