**Exercício 1 – Soma dos elementos**

Álvaro está economizando para comprar uma uma passagem para a europa. Por isso diariamente ele coloca um valor no cofre. Faça um programa que calcule o total acumulado no cofre até o momento.

**Input Format**

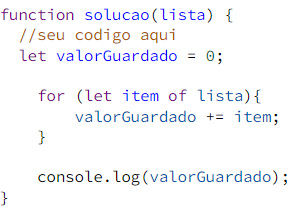
A entrada será sempre uma lista de inteiros positivos.

***Passos para resolver***

1. **Pegar o valor da primeira entrada**
2. **Guardar**
3. **Pegar o valor da segunda entrada**
4. **Somar o valor da primeira e segunda entrada**
5. **Guardar**

*Usar laço de repetição*

***Resolução entregue:***



**Exercício 2 – Média aritmética**

Álvaro está economizando para viajar para a europa. Todo dia ele coloca mais dinheiro no seu cofre. A quantidade de dinheiro que ele coloca no cofre a cada dia está registrada no array **lista**. Faça um programa que calcule quanto Álvaro vem acumulando, em média, por dia.

**Input Format**

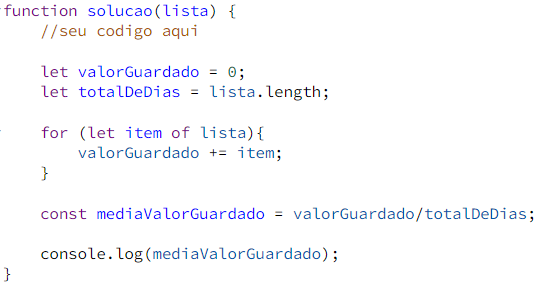
A entrada é uma lista que contém quanto Álvaro guarda no cofre a cada dia.

***Passos para resolver***

1. Pegar o valor da primeira entrada
2. Guardar
3. Pegar o valor da segunda entrada
4. Somar o valor da primeira e segunda entrada
5. Guardar
6. Repetir até somar todos os itens
7. Pegar o número de dias total (extenção do array)
8. Guardar a extenção do array
9. Dividir soma total das economias pelo total de dias

*Usar laço de repetição*

***Resolução entregue:***



**Exercício 3 – Números dentro de um intervalo**

Jacqueline e Emanuel acabaram de sair da aulas desesperados pela quantidade de exercícios de matemática que a professora Raissa passou como dever de casa. Para a sorte deles, você sabe programação e vai criar um programa pra ajudá-los a resolver todos os problemas do assunto *intervalos* entre dois números.

Eles pediram, então, para que você crie um programa que consiga fazer a prova-real de todos os exercícios da professora Raissa.

Seu objetivo é:

* Implementar uma função que receba três argumentos: numero, limiteInferior e limiteSuperior;
* Essa função deve retornar se esse número pertence ao conjunto que é limitado pelo **limiteInferior** e **limiteSuperior**

Pertencer ao conjunto significa que dado o **numero**, ele deve ser maior ou igual que o **limiteInferior** e menor ou igual do que o **limiteSuperior**.

**Input Format**

A entrada consista de três parâmetros: numero, limiteInferior e limiteSuperior.

**numero** refere-se ao número desejado para saber se ele está dentro ou não do limite. **limiteInferior** refere-se ao menor valor do limite de um dado intervalo; **limiteSuperior** refere-se ao maior valor do limite de um dado intervalo;

**Constraints**

* Quaisquer números inteiros;
* **limiteInferior** menor ou igual ao **limiteSuperior**;

**Output Format**

Imprima uma das duas opções abaixo:

* **PERTENCE** - para quando um número pertence ao limite delimitado;
* **NAO PERTENCE** - para quando um número **não** pertence ao limite delimitado;

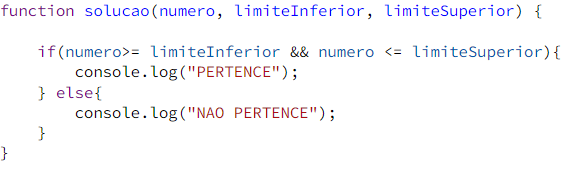
***Passos para resolver***

Pertencer ao conjunto significa que dado o **numero**, ele deve ser maior ou igual que o **limiteInferior** e menor ou igual do que o **limiteSuperior**.

1. Pegar o número
2. Verificar se é maior ou igual limiteInterior
3. && verificar se é menor ou igual limiteSuperior
4. Se pertencer ao limite retornar **PERTENCE**
5. Se não pertencer ao limite retornar **NAO PERTENCE**

*Usar if else*

***Resolução entregue:***

****

**Exercício 4 – Mesa de Poker**

Numa mesa de poker existe um valor mínimo de dinheiro que você precisa ter para poder jogar naquela mesa. Contudo, há também um limite máximo, pois jogadores com muito mais dinheiro na mesa levam vantagem. Faça um programa que selecione dentre um lista de valores, apenas aqueles que são permitidos para se jogar numa determiada mesa de poker.

**Input Format**

A entrada é costituida de 3 variáveis:

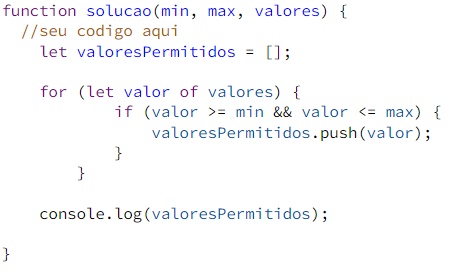
* **min** é o mínimo necessário para se poder jogar nesta mesa. É necessário ter o mínimo ou mais.
* **max** é o máximo permitido para se poder jogar nesta mesa. É necessário ter o máximo, ou menos.
* **valores** é um array que contém os valores com os quais o jogadores estão tentando sentar na mesa para jogar

***Passos para resolver***

1. Pegar a lista(array de valores)
2. Verificar se o item do array é maior ou igual ao minimo necessário
3. Verificar se o item do array é menor ou igual ao máximo permitido
4. Caso seja, colocar em lista apartada

*Usar for para repetição e if else para verificar o valor*

***Resolução entregue:***



**Exercício 5 – Truco**

O jogo de truco é muito popular pelo Brasil. Numa de suas versões, ele é jogado apenas com as cartas Q J K A 2 3. Elas tem essa ordem de "força" nesse jogo, sendo 3 a mais valiosa e Q a menos valiosa. Contudo, a cada partida é virada uma carta com a face pra cima na mesa. Esta carta serve para indicar que a próxima carta é a manilha, ou seja, a carta mais poderosa para essa partida. Fizemos uma tabela resumo para explicar essa mecânica:

| **Carta virada pra cima** | **Manilha** |
| --- | --- |
| Q | J |
| J | K |
| K | A |
| A | 2 |
| 2 | 3 |
| 3 | Q |

Supondo que a carta virada pra cima dessa rodada seja um ás (A), a manilha será 2, isso significa que 2 é a carta mais forte dessa rodada.

Faça um programa que diga qual é a manilha, dada uma carta virada para cima.

**Input Format**

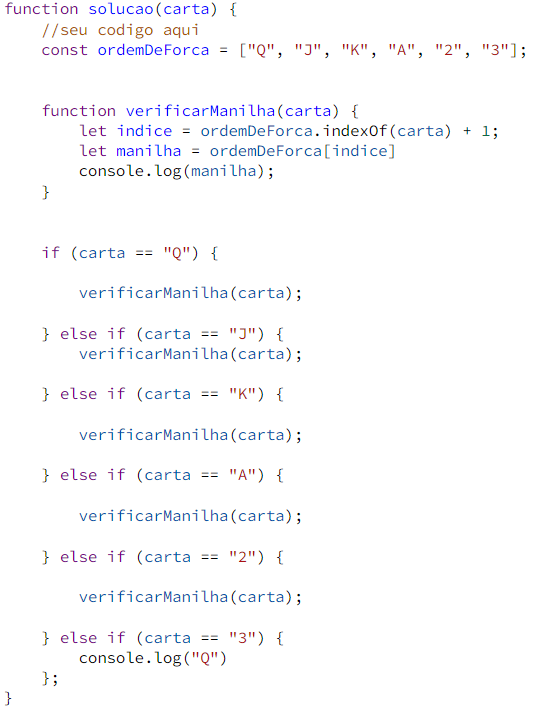
A entrada é um string armazenado na variável cartaParaCima que indica qual carta ficou virada para cima. Será sempre uma das opções: Q, J, K, A, 2, 3.

**Output Format**

Imprima na tela qual é a manilha desta partida. Sua resposta deve ser sempre uma das opções: Q, J, K, A, 2, 3.

***Passos para resolver***

1. Guardar uma lista array com a ordem de força das cartas.
2. Verificar o indice da carta retirada dentro do array
3. Somar mais 1 ao indice
4. Retornar a string da carta manilha



*Como verificar a manilha é um processo que se repete em todos os elementos a verificar foi possível incluir uma função para execução dessa tarefa*

**Exercício 6 – Caçula dos Adultos**

Um determinado evento é permitido apenas para maiores de idade. Através de um lista com as idades dos potenciais participantes, seu objetivo é fazer um programa que determine qual a idade da pessoa mais nova, dentre os que podem participar (maiores de idade).

**Input Format**

A entrada é um array com as idades das pessoas que tentam participar.

**Output Format**

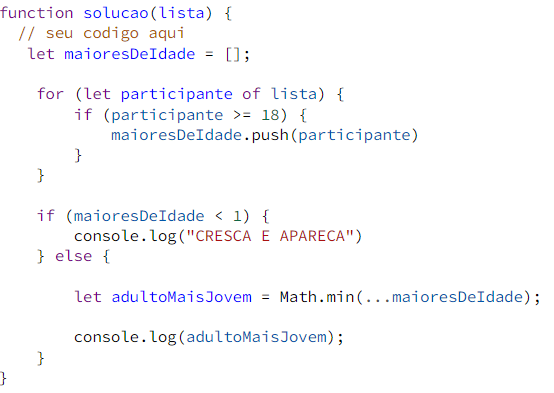
Imprima na tela a idade da pessoa mais jovem que pode participar. Se ninguém puder participar, imprima na tela a mensagem **CRESCA E APARECA**.

***Passos para resolver***

1. Pegar a lista de participantes
2. Verificar se o participante é maior de idade
3. Salvar em uma lista de maioresDeIdade
4. Verificar qual o menor numero da lista
5. Caso a lista maioresDeIdade esteja zerada imprimir **CRESCA E APARECA**.

*Usar for para repetição e if else para verificar o valor*

***Resolução entregue:***



**Exercício 7 – Contador de palavras**

Todo bom editor de texto informa no rodapé do programa quantas palavras tem no texto. Você está trabalhando numa empresa que está desenvolvendo um editor de texto e ficou com a tarefa de desenvolver esta funcionalidade. Para o texto dado na entrada, imprima na tela quantas palavras existem neste texto.

**Input Format**

A entrada será sempre um texto qualquer

**Constraints**

A entrada será sempre um texto qualquer com no máximo 5000 caracteres.

**Output Format**

Imprima na tela a quantidade de palavras contidas no texto.

***Passos para resolver***

1. Verificar a palavra recebida
2. Contar o número de palavras com base em um separador (“espaço”, “, virgula”, “.ponto”, “:”)

**Exercício 8 – Americano**

Num jogo de futebol entre amigos é muito comum que ninguém queira ser o goleiro. Para resolver esse impasse, um time decidiu utilizar o jogo "Americano".

Neste jogo, o time faz uma roda e cada um dos jogadores "joga" um número X qualquer. Após isso, o capitão do time soma todos os números jogados e obtém o resultado S. Depois, começa a contar de 1 até S apontando inicialmente para si mesmo (e falando alto e claramente "um"), depois para o jogador imediatamente a sua direita (e falando "dois") e assim por diante. O goleiro será aquele que estiver sendo apontado quando o capitão chegar a S.

Para fins de facilitar a resposta do problema, vamos considerar que o capitão está na posição 1, o jogador a sua direita está na posição 2, o jogador a direita deste está na posição 3, e assim por diante.

Você deve fazer um programa que determina qual a posição do jogador que deve ser o goleiro.

**Input Format**

A entrada será um array com N números, que corresempondem ao número jogado por cada um dos jogadores do time. Logo, se o array tiver tamanho 11, significa que este time tem 11 jogadores, por exemplo.

**Constraints**

0 < X < 11

N < 12

Passos para resolver

1. Verificar a soma dos números
2. Enquanto

**Exercício 9 – Promoção**

Para o dia dos namorados, um loja de presentes que, sabiamente, investiu em tecnologia e fazia uma profunda análise de dados percebeu que mais de 80% dos seus clientes compravam 2 itens nessa época. Com o objetivo de tentar aumentar o faturamento, essa loja lançou uma promoção, na qual o cliente que comprasse pelo menos 3 itens, teria um desconto de 50% no item mais barato.

Contudo, juntando a alta demanda desse período com o fato de o caixa ter que calcular esse desconto manualmente está causando filas demais. Seu papel, como desenvolvedor(a) dessa empresa é fazer um algoritmo que calcule automaticamente o valor devido pelo cliente ao se aplicar essa promoção quando necessário.

**Input Format**

A entrada será sempre um vetor de inteiros positivos. Cada inteiro desse representa o valor de cada produto comprado por um dado cliente, em centavos.

**Constraints**

O vetor terá sempre no máximo 100 itens.

Passos para resolver

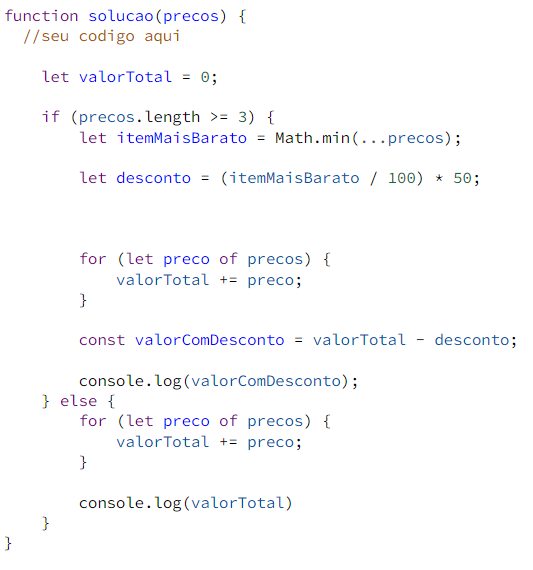
1. Verificar quantos itens tem na lista
2. Se tiver menos de 3, informar a soma
3. Se tiver 3 ou mais:

3.1 – Verificar qual é o item mais barato

3.2 - Aplicar o desconto no item mais barato

3.3 –Somar com os demais itens

3.4 – Informar o valor com desconto



**Exercício 10 – Zerinho ou Um**

Quando precisa-se escolher apenas uma pessoa aleatoriamente dentro de um grupo, é comum jogar "zerinho ou um" para sortear alguém. O jogo é muito simples: cada jogador joga 0 ou 1 com as mão. Aquele que for o único(a) a jogar zero ou um é o sorteado. Faça um programa que imprima o nome da pessoa que foi sorteada, ou NINGUEM, caso ninguém tenha sido sorteado(a).

**Input Format**

A entrada será sempre um vetor de objetos chamado jogadores, em que cada objeto é uma pessoa, com o seguinte formato:

{

nome: "Herbert",

jogada: 0 //será sempre 0 ou 1

}

**Constraints**

Pode-se assumir que sempre haverá pelo menos 3 pessoas jogando, ou seja, a entrada será sempre um vetor com pelo menos 3 itens. Você não precisa fazer nenhum código para checar isso.

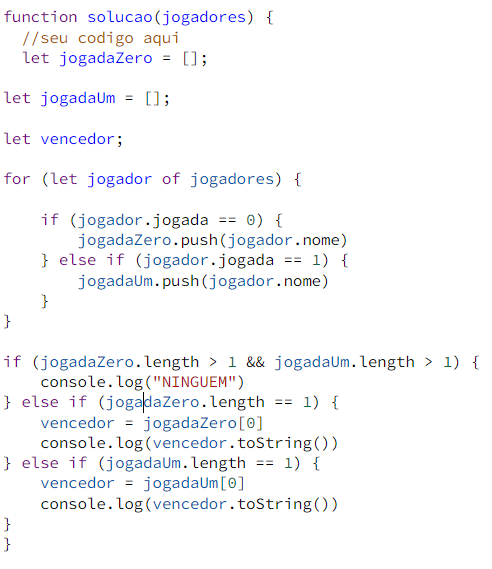
Passos para resolver

1. Separar em um novo array de objetos todos os jogadores que jogaram 1
2. Separar em um novo array de objetos todos os jogadores que jogaram 0
3. Verificar o tamando de cada array

3.1 Se os dois arrays forem maiores que um ninguém ganha

3.2 Se algum dos array for apenas 1, verificar o nome do jodador

4. Imprimir o nome do jogador ganhador



**Exercício 11 – Taximetro**

Você é o programador responsável por programar o "taxímetro" do novo aplicativo de mobilidade que a empresa onde você trabalha está lançando. Nessa tarefa, dada uma distância percorrida e um tempo de viagem, você tem que fazer um programa que calcula o preço da viagem. Sabe-se que o app deve cobrar, ***inicialmente***, **50 centavos por minuto** de viagem e ***mais*** **70 centavos por cada quilômtro** de viagem realizado. Caso a **viagem** tenha **mais de 10km**, cada **km adicional** (acima de 10) fica mais barato, passando a custar apenas **50 centavos por km**. Caso a **viagem** dure **mais de 20min**, cada **min adicional** (acima de 20) fica mais barato, passando a custar apenas **30 cetavos** por minuto. Seu trabalho é fazer a parte do programa que calcula, em centavos, o valor a ser pago pelo cliente (quanta responsabilidade, hein?). Lembre-se de arredondar para baixo o valor final a ser pago.

**Input Format**

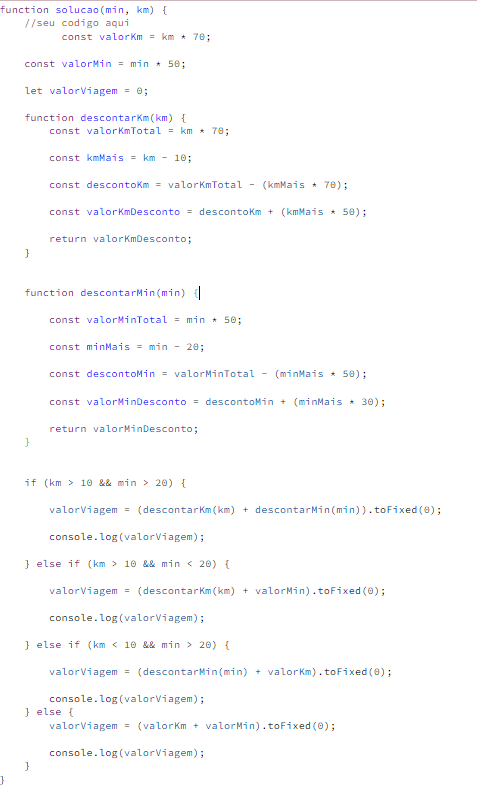
A entrada é composta por duas variáveis: - min representa quantos minutos de duração a viagem teve. É sempre um número inteiro. - km representa quantos quilômetros foram percorridos na viagem. Pode ser um número com casas decimais.

**Constraints**

-

Passos para resolver

1. Multiplicar os km por 70
2. Multiplicar os minutos por 50
3. Somar o total da viagem
4. Verificar os minutos a mais e km a mais
5. Se tiver percorrido km a mais e minutos a mais aplicar função desconto
6. Se tiver apenas minutos a mais, aplicar desconto de minutos
7. Se tiver apenas km a mais, aplica desconto de km
8. Informa os valor total com desconto



**Exercício 12 – CapsLock**

vOCÊ ESTÁ DESENVOLVENDO UM FORMULÁRIO DE CADASTRO E NOS PRIMEIROS TESTES DE USABILIDADE COM USUÁRIOS REAIS NOTOU ALGO PECULIAR: mUITOS USUÁRIOS PREENCHEM O FORMULÁRIO TODO COM A TECLA cAPS lOCK ATIVA, DEIXANDO TUDO BEM MENOS AGRADÁVEL DE LER. a SOLUÇÃO ESCOLHIDA NO dAILY DO DIA SEGUINTE FOI DE DETECTAR QUANDO O USUÁRIO ESTÁ ESCREVENDO DESSA FORMA E CORRIGIR AUTOMATICAMENTE. vOCÊ DEVE AGORA ESCREVER ESSE ALGORITMO.

cONSIDERAMOS QUE UMA PALAVRA FOI ESCRITA COM cAPS lOCK SE OU TODAS AS SUAS LETRAS FOREM MAIÚSCULAS OU SE A PRIMEIRA FOR MINÚSCULA E TODAS AS OUTRAS MAIÚSCULAS. a CORREÇÃO NESSES CASOS É INVERTER TODAS AS LETRAS.

**Input Format**

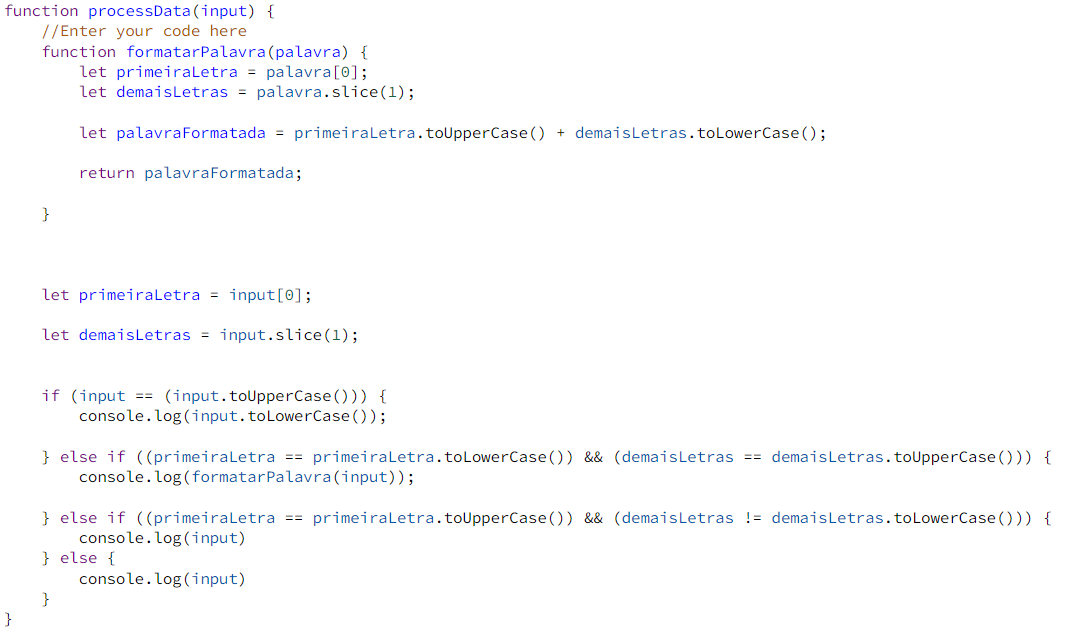
a ENTRADA CONSISTE DE UMA ÚNICA PALAVRA CONTENDO APENAS LETRAS LATINAS (A-Z), MINÚSCULAS OU MAIÚSCULAS.

**Constraints**

uMA PALAVRA POSSUI NO MÁXIMO 100 LETRAS.

***Passos para resolver***

1. Verifica se a palavra é toda em letra maiuscula
2. Caso não verificar se a palavra tem a primeira letra minuscula e o restante em maiuscula
3. Se não, não está com capsLock ativo. Apenas imprimir a palavra digitada



**Oque foi feito:?**

* Se a palavra está toda em letra maiúscula, é transformada inteira em minúscula.
* Se a palavra em a primeira letra minúscula e as demais letras maiúscula, a primeira fica maiúscula e as demais minúsculas.
* Para todos os outro casos é mantida como está!