Exercícios: analise os trechos de código abaixo e estime o tempo de execução para o melhor e o pior caso de cada um.

a) Busca de maior valor em vetor

		Operação	Melhor Caso	Pior Caso
1.	função MaiorValor(vetor[ ]:int)			
2.	maior = vetor[0]	op1	1	1
3.	para i = 1 até tamanho(vetor)	op2	n	n
4.	se (maior < vetor[i])	op3	n-1	n-1
5.	maior = vetor [i]	op1	0	n-1
6.	retorna(maior)	op4	1	1

O melhor caso ocorre com o vetor ordenado de forma decrescente, pois assim o primeiro valor é o maior e não ocorre trocas. O pior caso ocorre com o vetor ordenado de forma crescente, pois todos os valores testados serão trocados.

Melhor caso: $T(n) = op1 + n\,op2 + (n-1)\,op3 + op4$
$= op1 + n\,op2 + n\,op3 - op3 + op4$
$= n\,op2 + n\,op3 + op1 - op3 + op4$
$= n(op2 + op3) + op1 - op3 + op4$

Pior caso: $T(n) = op1 + n\,op2 + (n-1)\,op3 + (n-1)\,op1 + op4$
$= op1 + n\,op2 + n\,op3 - op3 + n\,op1 - op1 + op4$
$= n\,op2 + n\,op3 + n\,op1 + op1 - op3 - op1 + op4$
$= n(op2 + op3 + op1) - op3 + op4$

Apesar de haver diferença no comportamento do melhor e do pior caso, os dois se apresentam com uma estimativa de tempo de execução linear ($t(n) = n$).

b) Trecho de cálculo

		Operação	Melhor Caso	Pior Caso
1.	int aux1=1;	op1	1	1
2.	int aux2=1;	op1	1	1
3.	aux1 = aux1 * aux2;	op1+op2	1	1
4.	aux2 = aux2 + 1;	op1+op3	1	1
5.	aux2 = aux1 + 1;	op1=op3	1	1
6.	aux2 = aux2 + aux1;	op1+op3	1	1

Não existe diferença (melhor/pior caso). O tempo de execução é constante.

c) Busca sequencial em vetor

		Operação	Melhor	Pior
1.	função BuscaSequencial(vetor[ ]:int; valor:int)			
2.	achou=falso	op1	1	1
3.	i = 0	op1	1	1
4.	posição=-1	op1	1	1
5.	enquanto i < tamanho(vetor) && achou=falso	op2	2	n+1
6.	se (valor = vetor [i])	op3	1	n
7.	achou=true	op1	1	1
8.	posição=i	op1	1	1
9.	retorna(posição)	op4	1	1

O melhor caso ocorre com o valor procurado na primeira posição e o pior caso ocorre com o valor procurado na última posição.

Melhor caso: $T(n) = 3\ op1 + 2\ op2 + op3 + 2\ op1 + op4$
$= 5\ op1 + 2\ op2 + op43 + op4$

Pior caso: $T(n) = 3\ op1 + (n + 1)\ op2 + n\ op3 + 2\ op1 + op4$
$= 3\ op1 + n\ op2 + op2 + n\ op3 + 2\ op1 + op4$
$= n\ (op2 + op3) + 5\ op1 + op2 + op4$

No melhor caso o tempo de execução é constante e o pior caso se apresenta com estimativa de tempo de execução linear ($t(n) = n$).

d) Somatório de valores em vetor

		Operação	Melhor	Pior
1.	função SomatórioVetor(vetor[ ]:int)			
2.	soma=0	op1	1	1
3.	i = 0	op1	1	1
4.	enquanto i < tamanho(vetor)	op2	n+1	n+1
5.	soma = soma + vetor[i]	op1+op3	n	n
6.	retorna(soma)	op4	1	1

$T(n) = 2\ op1 + (n+1)\ op2 + n\ op1 + n\ op3 + op4$
$= 2\ op1 + n\ op2 + op2 + n\ op1 + n\ op3 + op4$
$= n\ (op2 + op1 + op3) + 2\ op1 + op4$

Não existe distinção entre casos e o tempo de execução é linear ($t(n) = n$).

e) Considerar neste caso o somatório de uma matriz de dimensões nxn

		Operação	Melhor	Pior
1.	função SomatórioMatriz(matriz[][]:int)	Operação	Melhor	Pior
2.	soma=0	op1	1	1
3.	i = 0	op1	1	1
4.	enquanto i < tamanho_linha(matriz[][])	op2	n+1	n+1
5.	j=0	op1	n	n
6.	enquanto j < tamanho_coluna(matriz[][])	op2	n*(n+1)	n*(n+1)
7.	soma = soma + matriz[][]	op1+op3	n*n	n*n
8.	j= j + 1	op1+op3	n*n	n*n
9.	i = i + 1	op1+op3	n	n
10.	retorna(soma)	op4		

$T(n) = 2\ op1 + (n+1)\ op2 + n\ op1 + n(n+1)\ op2 + 2*n*n\ op1 + 2*n*n\ op3 + n\ op1 + n\ op3 + op4$
$= 2\ op1 + n\ op2 + op\ 2 + n\ op1 + n^2\ op2 + n\ op2 + 2n^2 op1 + 2n^2 op3 + n\ op1 + n\ op3 + op4$
$= n^2(op2 + 2op1 + 2op3) + n(2op2 + 2op1 + op3) + 2\ op1 + op\ 2 + op4$

A estimativa do tempo de execução é quadrática ($T(n) = n^2$)