

---

## **TESTE DE PROGRAMAÇÃO**

Nome: Gabriel de Mores Lourenço Pereira Pinto

**Questão 1.** Escreva um algoritmo que receba uma certa quantidade de números de entrada, fornecida por um usuário e imprima o maior número dentre eles e quantas vezes este maior número foi repetido. Assuma que o usuário sempre fornecerá um número inteiro positivo. A quantidade de números a serem recebidos deve ser fornecida pelo usuário.

```
#include <iostream>
#include <cstdlib>
#include <vector>
```

```
int main()
```

```
{
    int qNumeros, n;
    std::vector<int> v = [0];

    std::cin >> qNumeros;

    for(int i = 0; i < qNumeros; ++i)
    {
        cin >> n;
        v.push_back(n);
    }

    int maiorValor = 0;
    int quantidade = 0;

    for(int i = 0; i < qNumeros ; ++i)
    {
        if(i == 0)
        {
            maiorValor = v[i];
        }
        if(maiorValor < v[i])
```

---

```
        {
            maiorValor = v[i];
        }
    }

    for(int i = 0; i < qNumeros ; ++i)
    {
        if(v[i] == maiorValor)
        {
            quantidade++;
        }
    }

    std::cout << "O maior valor e: " << maiorValor << std::endl;
    std::cout << "A quantidade que ele aparece e: " << quantidade << std::endl;

    return 0;
}
```

**Questão 2.** Qual será o resultado?

0xAC | 0x18

0x93 & 0xFE


0x3D << 2

**Questão 3.** Crie um algoritmo para verificar se um número é primo.

**Questão 4.** Dado um vetor de tamanho n de números inteiros, faça um algoritmo que ordene em ordem crescente este vetor.

```
std::vector ordenaVetor(std::vector v)
{
    v.sort();
    return v;
}
```

**Questão 5.** Dada a tabela de precedência de operadores, calcule:

Prioridade	Operador	Associatividade
<b>Maior</b>	() [] -> .	da esquerda para a direita
	! ~ ++ -- - (tipo) * & sizeof	da direita para a esquerda
	* / %	da esquerda para a direita
	+ -	da esquerda para a direita
	<< >>	da esquerda para a direita
	< <= > >=	da esquerda para a direita
	== !=	da esquerda para a direita
	&	da esquerda para a direita
	^	da esquerda para a direita
		da esquerda para a direita
	&&	da esquerda para a direita
		da esquerda para a direita
	?:	da direita para a esquerda
	= += -= etc.	da direita para a esquerda
<b>Menor</b>	,	da esquerda para a direita

**5.1.** Qual o valor de result?

Considere para cada expressão

int a=1;

int b=1;

int c=3;

int d=2;

int e=3;

int f=3;

**5.1.1** result= a+b\*c

result = 4

**5.1.2** result= a-d++

result = 0

**5.1.3** result= --e+f++

result = 6

**5.1.4** result= ++a;

result = 2

---

**5.1.5** result= c++;

result = 3

**5.1.6** result= b+=c++\*--d+!c+e<<f+2-d+a\*b;

result = 16

**5.1.7** result= d-- > --e\*d? f:--a;

result = 0

### Questão 6.

Dado um início do código a seguir :

```
#include <string.h>
```

```
#include <stdlib.h>
```

```
struct Nome {
```

```
    int contador;
```

```
    char *nome;
```

```
};
```

```
typedef struct Nome Nome_Def;
```

```
void funcao1 (void){
```

```
    Nome_Def matriz [5][10];
```

```
    for(int i = 0; i < 10; ++i)
```

```
    {
```

```
        for(int j=0; j < 5; ++j)
```

```
        {
```

```
            matriz[j][i].contador = j;
```

```
            matriz[j][i]->nome = "Nome" + (char)j + (char)i;
```

```
        }
```

```
    }
```

```
}
```

```
void funcao2(void){
```

```
    Nome_Def *matriz[5];
```

```
    for(int i =0; i < 5; i++)
```

```
    {
```

```
        matriz->[0].contador = i;
```

```
        matriz->[0]->nome = "Nome" + (char)i;
```

```
    }
```

```
}
```

```
void funcao3 (void){
```

```
    Nome_Def **matriz;
```

---

```
(&matriz).contador = 0;  
(&matriz)->nome = "Nome 0";  
}
```

Implemente as três funções acima, “carregando” todos os campos da variável “matriz” com os valores conforme esquematização:

(0,"Nome00")	(0,"Nome01")	...	...	...	...	...	...	...	(0,"Nome09")
(1,"Nome10")	(1,"Nome11")	...	...	...	...	...	...	...	(1,"Nome19")
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
(4,"Nome40")	(4,"Nome41")	...	...	...	...	...	...	...	(4,"Nome49")