DOM | SEG | TER | QUA | QUI | SEX | SÁB

Simulado 1

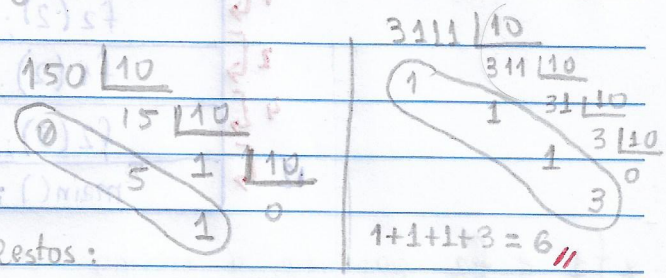
Discente - Paulo Henrique Diniz de Lima Alencar. • matrícula: 494837

1 - Resolução:

```
#include <stdio.h>
int irmaos (int x, int y){
    int soma_x = 0, soma_y = 0;
    while (x != 0 || y != 0){
        soma_x = soma_x + x % 10;
        soma_y = soma_y + y % 10;
        x = x / 10;
        y = y / 10;
    }
    if (soma_x == soma_y) return 1;
    return 0;
}

int main (void) {
    printf("retorno = %d", irmaos (150, 3111));
    return 0;
}
```

A ideia consiste em ir realizando divisões consecutivas de um determinado número por 10, na qual obteremos cada dígito do número por meio do seu resto, da seguinte forma:



Restos:

0 + 5 + 1 = 6 //. Assim, a soma dos restos é a soma de seus dígitos. Por isso se a soma dos dígitos de 2 determinados números são iguais eles são irmãos.

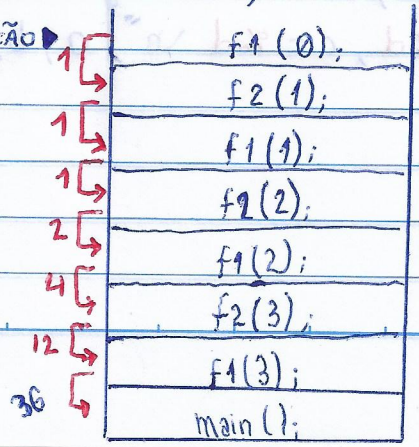
1 + 1 + 1 + 3 = 6 //

2 - Resolução:

* Se **f1** for chamada na main();

• PILHA DE EXECUÇÃO ▶

se n = 3



1 • f2(1) ⇒ 1 • 1 = 1

1 • f1(1) ⇒ 1 • 1 = 1

2 • f2(2) ⇒ 2 • 1 = 2

2 • f1(2) ⇒ 2 • 2 = 4

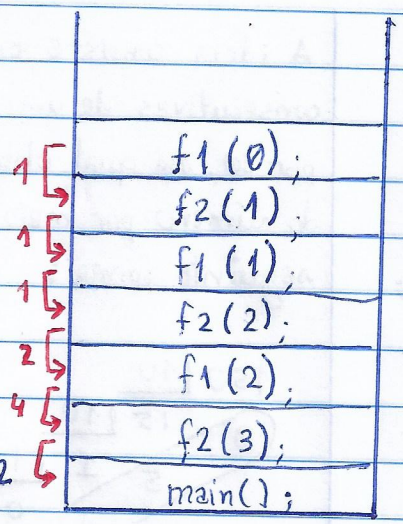
3 • f2(3) ⇒ 3 • 4 = 12

3 • f1(3) ⇒ 3 • 12 = 36 ///

* Se f2 for chamada na main();

• n = 3

PILHA DE EXECUÇÃO ▼



1 • f2(1) ⇒ 1 • 1 = 1

1 • f1(1) ⇒ 1 • 1 = 1

2 • f2(2) ⇒ 2 • 1 = 2

2 • f1(2) ⇒ 2 • 2 = 4

3 • f2(3) ⇒ 3 • 4 = 12 ///

* f1 e f2 possuem o mesmo caso base, isto é, uma condição de parada: n <= 0. Se cair no caso base, a função retorna 1 para a função que chamou ela, realizando assim o desempilhamento da pilha de execução. As duas ilustrações demonstram o desempilhamento e o retorno de cada função até o seu retorno final.

3 - Resolução:

```
# include <stdio.h>

void hipotenusas (int n) {
    int a=1, b=4, c=3, cont=2;
    While (a<=n){
        if(a*a == b*b + c*c){
            printf("a=%d, b=%d, c=%d \n", a, b, c);
            b= 4 * cont;
            c= 3 * cont;
            cont++;
        }
        a++;
    }
}
```

```c
int main (void) {
    int n;

    While (1) {
        printf ("Entre com um n, onde n>=1 :");
        scanf ("%d", &n);
        if (n>=1) break;
    }

    hipotenusas (n);
    return 0;
}
```

4 - Resolução:

a) Resolução:

```c
#include <stdio.h>

int a (int n) {
    int pot = 2;

    While (pot - 1 <= n) {
        if (pot - 1 == n) return 1;
        pot = pot * 2;
    }
    return 0;
}
```

```c
int main (void) {
    int n;
    scanf ("%d", &n);
    printf ("%d\n", a(n));
    return 0;
}
```

b) Resolução:

```
void b (int n) {
    int pot = 2, cont = 0, j = 1;

    While (cont < n) {
        pot = 2;
        While (pot - 1 <= j) {
            if (pot - 1 == j) {
                int i;
                for (i = 2; i <= j / 2; i++) {
                    if (j % i == 0) break;
                }
                if (i == j / 2 + 1) {
                    printf("%d é um número de mersenne e tbm é primo \n", j);
                    cont++;
                }
            }
            Pot = pot * 2;
        }
        j++;
    }
}
```

* verifica se é nº de mersenne!

* verifica se é primo!