```c
/*Lista de exercícios 10 - Recursividade/Ponteiro/Alocação de memória
Guilherme de Abreu Lima Buitrago Miranda
Matrícula: 2018054788*/

#include <stdio.h>
#include <math.h>

int fatorial (int numero, int resultado){
	if (numero == 1)
		return resultado;
	else{
		resultado = resultado * numero;
		numero--;
		return(fatorial(numero, resultado));
	}
}

int mdc (int numeroA, int numeroB){
	//algoritmo de Euclides (recursivo)
	if (numeroB == 0)
		return numeroA;
	else
		return mdc(numeroB, numeroA % numeroB);
}

int mdc3 (int numeroA, int numeroB, int numeroC){
	//mdc3 = mdc(c(mdc(a,b)))
	int resultAB = mdc(numeroA, numeroB);

	if (resultAB == 0)
		return numeroC;
	else
		return mdc(resultAB, numeroC % resultAB);

}

int fib (int numero){
	//Dois casos de base:
	if (numero == 0)
		return (0);
	else if (numero == 1)
		return(1);
	//Numero da sequencia = soma dos dois anteriores
	else{
		return(fib(numero-1) + fib(numero-2));
	}
```

```
}

int primo (int numero){ //crivo de erastostenes não otimizado.
	int raiz = floor(sqrt(numero));

	int primoRec(int numero, int raiz){
		if (raiz <= 1 || numero == raiz)
			return 1;
		else if (numero % raiz == 0)
			return 0;
		else
			return(primoRec(numero, raiz-1));
	}

	primoRec(numero, raiz);
}

void decrescente (int numero){
	if (numero == 0)
		return;
	else{
		printf("%d\n", numero);
		return decrescente(numero-1);
	}
}

int res(int a, int b){
   if(a>=b){
      a-=b;
      res(a,b);
   }else
      return a;
}

int form (int numero){
	if (numero == 1)
		return numero;
	else
		return numero * numero + form(numero-1);
}

int mmc (int a, int b){
	return ((a*b)/mdc(a, b));
}

int div (int a, int b){
```

```
        if(a>b)
       return ( 1 + div(a-b, b) );
    else if (a - b == 0)
        return 1;
    else
        return 0;
}

float sqrt_(int numero, float xn, float x0){
        //metodo babilonico
        //https://en.wikipedia.org/wiki/Methods_of_computing_square_roo
ts#Babylonian_method
        float precisao = 0.001;
        if (x0 - xn > precisao){
                float aux = (x0+xn)/2;
                sqrt_(numero, (numero/aux), (x0+xn)/2);
        }else
                return x0;
}

int dig(int numero){
        if (numero >= 10)
                return (numero % 10 + dig(numero/10));
        else
                return numero;
}

int exp_ (int numK, int numN){
        if (numN != 1)
                return (numK * exp_(numK, numN-1));
        else
                return numK;
}

void crescente (int numero){
        if (numero == 0)
                return;
        else{
                crescente(numero-1);
                printf("%d\n", numero);
        }
}
int main(){
        crescente(10);
        return 0;
}
```