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

int contadigito(int x)
{
   if (x == 0)
   {
      return 0;
   }
   else
   {
      return 1 + contadigito(x / 10);
   }
}

int main(void)
{
   int n, ndigit;
   scanf("%d", &n);
   ndigit = contadigito(n);
   printf("%d", ndigit);
}
```

//

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

int divisao(int x)
{

   if (x == 0)
   {
      return 0;
   }
   else
   {
      return (x % 10) + somadigito(x / 10);
   }
}

int main(void)
{
   int n, sdigit;
   scanf("%d", &n);
   sdigit = somadigito(n);
   printf("%d", sdigit);
}
```

//

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

int divisao(int x, int y)
{
   if (x < y)
   {
      return 0;
   }
   else
   {
      return 1 + divisao(x - y, y);
   }
}

int main(void)
{
   int dividendo, divisor, res;
   scanf("%d", &dividendo);
   scanf("%d", &divisor);
   res = divisao(dividendo, divisor);
   printf("%d", res);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <stdlib.h>

int divisao(int x, int y)
{
   if (x < y)
   {
      return 0;
   }
   else
   {
      return 1 + divisao(x - y, y);
   }
}

int main(void)
{
   int dividendo, divisor, res;
   scanf("%d", &dividendo);
   scanf("%d", &divisor);
   res = divisao(dividendo, divisor);
   printf("%d", res);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

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

float fat(int x)
{
   if (x == 0 || x == 1)
   {
      return 1;
   }
   else
   {
      return x * fat(x - 1);
   }
}

float calculae(float y)
{
   if (y == 0)
   {
      return 0;
   }
   else
   {
      return 1 / fat(y) + calculae(y - 1);
   }
}

int main(void)
{
   int n1;
   float valorE;
   scanf("%d", &n1);
   valorE = calculae(n1);
   printf("%.2f", valorE);
}
```

//

```c
#include <stdio.h>

int main(void)
{
   int x = 0;
   int *p = &x;

   p++; // o valor adicionado mudará o local de P na memória (adicionando 4 bytes)
   (*p)++; // o valor adicionado mudará o valor atrelado a variável
   *(p++); // acessará o valor da variável atrelado ao endereço de X + 4 bytes
}
```

//

// somente a C será verdadeira, pois esta está atribuindo o endereço da variável (&valor) ao ponteiro ValorPtr

```
#include <stdio.h>

int main(void)
{
   int valor;
   int *valorptr;
   valorptr = &valor;
   printf("%p", valorptr);
}
```

//

```
#include <stdio.h>

int main()
{
   int x, *p, **q;
   p = &x;
   q = &p;
   x = 10;
   printf("\n%d\n", **q); // para printar o valor 10 na tela, foi necessário adicionar mais um * no q do código original
   return 0;
}
```

//

```
#include <stdio.h>

void converteHora(int totalSeg, int *h, int *m, int *s)
{
   *h = totalSeg / 3600;
   *m = (totalSeg % 3600) / 60;
   *s = totalSeg % 60;
}

int main()
{
   int segtotal, hora, min, seg;
   scanf("%d", &segtotal);
   converteHora(segtotal, &hora, &min, &seg);
   printf("%02d:%02d:%02d\n", hora, min, seg);
   return 0;
}
```

//

// A diferença entre os dois códigos é que, no primeiro, o uso do *ptr indica que o criador está usando o valor atrelado ao endereço o qual o ponteiro ptr está apontando. Já no segundo código, o uso de ptr sem * indica a manipulação do local onde a variável está alocada na memória, por isso o resultado impresso é diferente nos dois códigos.