7) Codificação de Church é uma forma de incorporar dados e operadores numéricos na notação lambda. Os numerais de Church são uma representação dos números naturais em cálculo-λ. São representados fomo uma função sucessor e podem ser chamados n vezes. O valor dos numerais de Church é definido pelo número de vezes que a função é chamada.

sucessor ≡ `λn.λf.λx. f (n f x)`
sucessor `de 0 =  λn.λf.λx. f (0 f x)`
sucessor `de 1 =  λn.λf.λx. f (1 f x)`
sucessor `de 2 =  λn.λf.λx. f (2 f x)`

Os numerais de Church também podem ser representados da seguinte maneira:

```
 0 = λf.λx.x
 1 = λf.λx. f x
 2 = λf.λx. f (f x)
 3 = λf.λx. f (f (f x))
 4 = λf.λx. f (f (f (f x)))
 5 = λf.λx. f (f (f (f (f x))))
 6 = λf.λx. f (f (f (f (f (f x)))))
 7 = λf.λx. f (f (f (f (f (f (f x))))))
 8 = λf.λx. f (f (f (f (f (f (f (f x)))))))
 9 = λf.λx. f (f (f (f (f (f (f (f (f x))))))))
10 = λf.λx. f (f (f (f (f (f (f (f (f (f x)))))))))
...
```
$n = \lambda f.\lambda x.\ f^{n}\ x$

Utilizando a notação de representação dos numerais, é possível definir uma função de adição para operar com dois valores (m + n)

plus ≡ `λm.λn.λf.λx. m f (n f x)`

Para realizar a soma (por exemplo, 0 + 1), é necessário aplicar à adição as funções sucessor dos numerais desejados. Assim, a operação 0 + 1 seria representada por:

`λm.λn.λf.λx. 1 f (0  f x)`	Aplicando a codificação de Church, temos
`(λf.λx. f x) f ((λf.λx. x)  f x)`	Que, ao passar por beta-redução, se torna:
`λm.λn.λf.λx.fx`	Que é equivalente a:
`λm.λn.λf.λx.fx`,	A notação para 1 nos numerais de Church

O mesmo pode ser feito para a operação 1 + 2, demonstrada a seguir:

`λm.λn.λf.λx. 2 f (1 f x)`	Substituindo nos numerais de Church:
`λm.λn.λf.λx. (λf.λx. f (f x)) f ((λf.λx. f x) f x`	Que, após beta-redução, se torna
`λm.λn.λf.λx.f(f(fx))`	Equivalente a:
`λf.λx.f(f(fx))`	Notação de Church para 3

A subtração pode ser definida utilizando a funcção predecessor **`pred`**, da forma

`sub``: λm.λn.(n `**`pred`**`) m;`

onde n é o numero de vezes que a função será aplicada.

Utilizando essas funções, podemos calcular, por exemplo, 1 - 0 da seguinte maneira:

`λm.λn.(0 `**`pred`**`) 1`	Substituindo com os numerais de Church:
`λm.λn.((λf.λx. x) `**`pred`**`) (λf.λx. f x)`	Aplicando beta-redução, temos:
`λm.λn.λf.λx.fx`	que é equivalente a:
`λf.λx.fx`	codificação de Church para 1

Um outro exemplo, agora realizando a operação 2 - 1:

`λm.λn.(1 `**`pred`**`) 2`	Aplicando os numerais na codificação de Church:
`λm.λn.((λf.λx. f x) `**`pred`**`) (λf.λx. f( f x))`	Após realizar beta-redução, obtemos:
`λm.λn.λf.λx.fx`	que equivale a
`λf.λx.fx`	notação para 1 em numerais de Church

8) Combinador y ou combinador ponto-a-ponto é uma função y que, quando aplicada a f, obtém o mesmo resultado que f aplicada ao resultado da aplicação de f a y. Ou seja:

`y f = f(y f)` *para todo* `f`

A importancia do combinador y é que ele permite que uma função chame a ela mesma possibilitando assim a implementação de recursividade em cálculo-λ para representar, por exemplo, a função fatorial (***fat****(n) = n ** ***fat****(n-1)*)