Sartre Anthonin Programmation fonctionelle TP1

Question 1 :

```
λ > (lambdaf.(f(lambdax.(lambdax.(lambday.y)))(lambdap.(lambdaq.p))))(lambdaa.(lambdab.b))
ω > wrap abstraction and application with brackets:
     (lambdaf.(f(lambdax.(lambdax.(lambday.y)))(lambdap.(lambdaq.p))))(lambdaa.(lambdab.b)) ->
((lambdaf.(f(lambdax.(lambdax.(lambday.y)))(lambdap.(lambdaq.p))))(lambdaa.(lambdab.b)))
α > ((λf.((f(λx.(λx0.(λy.y))))(λp.(λq.p))))(λa.(λb.b)))
β > (((λa.(λb.b))(λx.(λx0.(λy.y))))(λp.(λq.p)))
β > ((λb.b)(λp.(λq.p)))
β > (λp.(λq.p))
λ > (lambdaf.(f(lambdax.(lambdax.(lambday.y)))(lambdap.(lambdaq.p))))(lambdaa.(lambdab.a b))
ω > wrap abstraction and application with brackets:
     (lambdaf.(f(lambdax.(lambdax.(lambday.y)))(lambdap.(lambdaq.p))))(lambdaa.(lambdab.a b)) ->
((lambdaf.(f(lambdax.(lambdax.(lambday.y)))(lambdap.(lambdaq.p))))(lambdaa.(lambdab.a b)))
α > ((λf.((f(λx.(λx0.(λy.y))))(λp.(λq.p))))(λa.((λb.a)b)))
ε >      unbound variable: b rename as x'0
((λf.((f(λx.(λx0.(λy.y))))(λp.(λq.p))))(λa.((λb.a) [x'0] )))
β > (((λa.((λb.a)x'0))(λx.(λx0.(λy.y))))(λp.(λq.p)))
β > (((λb.(λx.(λx0.(λy.y))))x'0)(λp.(λq.p)))
β > ((λx.(λx0.(λy.y)))(λp.(λq.p)))
β > (λx0.(λy.y))
```

Question 2 :

```
ISZERO(ZERO) == TRUE : true
ISZERO(ONE)  == FALSE : true
undefined
console.log(ISZERO)
f=>(f(x=>FALSE))(TRUE)
```

Question 3 :

```
ZERO = \x.\y.y
ONE = \x.\y.x y
TWO = \x.\y.x(x y)

SUCC = (\f.(\x.\y.f x(x y)))

SUCC(SUCC(ZERO))
```

Run

```
λx y.x(x y)
```

Question 4 :

```
SUCC(ZERO) == 1 : true
SUCC(ONE) == 2 : true
```

Question 5 :

```
> var PRED = k => (k(p=>u=>(u)(SUCC(p (TRUE)))(p (TRUE)))(u=>u (ZERO) (ZERO)))(FALSE)
< undefined
> var convert_to_int = (numeral) => {
    if (numeral === ZERO) {
  return 0; }
    else {
       return 1 + convert_to_int(PRED(numeral));
  } }
  console.log('PRED(ONE) == 0 :', (PRED(ONE)(x => x + 1)(0) == 0));
  console.log('PRED(TWO) == 1 :', (PRED(TWO)(x => x + 1)(0) == 1));
PRED(ONE) == 0 : – true
PRED(TWO) == 1 : – true
< undefined
```

Question 6 :

```
(\n.\p.(\times.\funct.\base.times funct base)(n)(\f.(\x.(\y.(f x(x y)))))(p))(\x.\y.x(x
y))(\x.\y.x(x y))
```

Run

```
λx y.x(x(x(x y)))
```

Question 7 :

```
(\m.\q.(\times.\funct.\base.times funct base)(m)(( \n.\p. (\times.\funct.\base.times funct
base) (n) (\f.(\x.(\y.(f x(x y))))) (p) ) q) (\x.\y.y)) (\x.\y.x(x y)) (\x.\y.x(x(x y)))
```

Run

```
λx y.x(x(x(x(x y))))
```

Question 8 :

```
> var MUL = m=>q=>ITER (m) (ADD (q)) (ZERO)
< undefined
> console.log('MUL(TWO)(THREE) == 6 :', (MUL(TWO)(THREE)(x => x + 1)
  (0) == 6));
MUL(TWO)(THREE) == 6 : – true
< undefined
```