**R :**

$$12 * x^2 + 12 * x - 3$$

je pense que

code :

```
# Classes S3 pour les expressions
Constante <- function(value) {
  structure(list(value = value), class = "Constante")
}

Variable <- function(name = "x") {
  structure(list(name = name), class = "Variable")
}

Addition <- function(left, right) {
  structure(list(left = left, right = right), class = "Addition")
}

Multiplication <- function(left, right) {
  structure(list(left = left, right = right), class = "Multiplication")
}

Puissance <- function(base, exponent) {
  structure(list(base = base, exponent = exponent), class = "Puissance")
}

Fonction <- function(func, arg) {
  structure(list(func = func, arg = arg), class = "Fonction")
}

# Fonction générique derivative
derivative <- function(expr, var = "x") {
  UseMethod("derivative")
}

# Méthodes derivative pour chaque classe
derivative.Constante <- function(expr, var = "x") {
  Constante(0)
}
```

```r
derivative.Variable <- function(expr, var = "x") {
  if (expr$name == var) {
    Constante(1)
  } else {
    Constante(0)
  }
}

derivative.Addition <- function(expr, var = "x") {
  Addition(derivative(expr$left, var), derivative(expr$right, var))
}

derivative.Multiplication <- function(expr, var = "x") {
  Addition(
    Multiplication(derivative(expr$left, var), expr$right),
    Multiplication(expr$left, derivative(expr$right, var))
  )
}

derivative.Puissance <- function(expr, var = "x") {
  n <- expr$exponent
  u <- expr$base
  Multiplication(
    Multiplication(Constante(n), Puissance(u, n - 1)),
    derivative(u, var)
  )
}

derivative.Fonction <- function(expr, var = "x") {
  f <- expr$func
  u <- expr$arg
  u_prime <- derivative(u, var)

  switch(f,
         sin = Multiplication(Fonction("cos", u), u_prime),
         cos = Multiplication(Constante(-1), Multiplication(Fonction("sin", u), u_prime)),
         exp = Multiplication(Fonction("exp", u), u_prime),
         log = Multiplication(Multiplication(Constante(1), Puissance(u, -1)), u_prime),
         stop(paste("Fonction non prise en charge:", f))
  )
}
```

```r
# Fonction générique simplify
simplify <- function(expr) {
  UseMethod("simplify")
}

# Méthodes simplify pour chaque classe
simplify.Constante <- function(expr) {
  expr
}

simplify.Variable <- function(expr) {
  expr
}

simplify.Addition <- function(expr) {
  left <- simplify(expr$left)
  right <- simplify(expr$right)

  if (inherits(left, "Constante") && left$value == 0) {
    return(right)
  }
  if (inherits(right, "Constante") && right$value == 0) {
    return(left)
  }
  if (inherits(left, "Constante") && inherits(right, "Constante")) {
    return(Constante(left$value + right$value))
  }
  Addition(left, right)
}

simplify.Multiplication <- function(expr) {
  left <- simplify(expr$left)
  right <- simplify(expr$right)

  if ((inherits(left, "Constante") && left$value == 0) ||
      (inherits(right, "Constante") && right$value == 0)) {
    return(Constante(0))
  }
  if (inherits(left, "Constante") && left$value == 1) {
    return(right)
  }
  if (inherits(right, "Constante") && right$value == 1) {
```

```
    return(left)
  }
  if (inherits(left, "Constante") && inherits(right, "Constante")) {
    return(Constante(left$value * right$value))
  }
  Multiplication(left, right)
}

simplify.Puissance <- function(expr) {
  base <- simplify(expr$base)
  exponent <- expr$exponent

  if (exponent == 0) {
    return(Constante(1))
  }
  if (exponent == 1) {
    return(base)
  }
  if (inherits(base, "Constante")) {
    return(Constante(base$value ^ exponent))
  }
  Puissance(base, exponent)
}

simplify.Fonction <- function(expr) {
  arg <- simplify(expr$arg)

  if (inherits(arg, "Constante")) {
    value <- switch(expr$func,
                    sin = sin(arg$value),
                    cos = cos(arg$value),
                    exp = exp(arg$value),
                    log = log(arg$value),
                    stop(paste("Fonction non prise en charge:", expr$func))
    )
    return(Constante(value))
  }
  Fonction(expr$func, arg)
}

# Fonction générique print
print <- function(expr, ...) {
```

```
  UseMethod("print")
}

# Méthodes print pour chaque classe
print.Constante <- function(expr, ...) {
  cat(expr$value)
}

print.Variable <- function(expr, ...) {
  cat(expr$name)
}

print.Addition <- function(expr, ...) {
  print(expr$left)
  cat(" + ")
  print(expr$right)
}

print.Multiplication <- function(expr, ...) {
  left_needs_parens <- inherits(expr$left, c("Addition"))
  right_needs_parens <- inherits(expr$right, c("Addition"))

  if (left_needs_parens) {
    cat("(")
    print(expr$left)
    cat(")")
  } else {
    print(expr$left)
  }
  cat(" * ")
  if (right_needs_parens) {
    cat("(")
    print(expr$right)
    cat(")")
  } else {
    print(expr$right)
  }
}

print.Puissance <- function(expr, ...) {
  base_needs_parens <- inherits(expr$base, c("Addition", "Multiplication"))
```

```
  if (base_needs_parens) {
    cat("(")
    print(expr$base)
    cat(")")
  } else {
    print(expr$base)
  }
  cat("^")
  cat(expr$exponent)
}

print.Fonction <- function(expr, ...) {
  cat(expr$func, "(")
  print(expr$arg)
  cat(")")
}
```

test 1 :

```
# Exemple 1

x <- Variable("x")

# f(x) = x^2 + 3x + 5
expr <- Addition(
  Addition(
    Puissance(x, 2),
    Multiplication(Constante(3), x)
  ),
  Constante(5)
)

# Calcul de la dérivée

d_expr <- derivative(expr, "x")
d_expr
```

```
2 * x^1 * 1 + 0 * x + 3 * 1 + 0
```

test 2 :

```r
# Exemple 2

expr2 <- Fonction("sin", Puissance(x, 2))
d_expr2 <- derivative(expr2, "x")
d_expr2
```

```
cos (x^2) * 2 * x^1 * 1
```

Dans le fichier derivative_r.R, la fonction generique derivative() utilise UseMethod("derivative") pour déléguer l'appel à une methode spécifique en fonction de la classe de l'argument expr. C'est un mecanisme de simple_dispathcing qui redirige vers la methode appropriée en fonction de la classe de l'expression donnée.