2) macros: (i) (incr++! var) (ii) (incr++! var expr)

```
(define-syntax (incr++!)
 (syntax-rules ()
  ((incr++! var)
   (incr++! var 1))
  ((incr++! var expr)
   (if (number? var)
    (begin
     (set! var (+ var expr))
     var)
    #f))))
```

- loop / fixed

```
(define-syntax (loop/fixed)
 (syntax-rules ()
  ((loop/fixed expr expr-rest ...)
   (let loop ((i 0))
    (if (> i expr)
     (begin
      expr-rest ...
      (loop (+ i 1)))
     #f)))))
```

(3) - playing with call/cc:

```
(define (reverse-by-2/erase lst)
 (call/cc
  (lambda (exit)
   (if (even? (length lst))
(begin (define i 0) (let loop ((i 0))
 (if (< i (length lst))
  (begin t= (lst i) (lst-i) = (lst (+i 1)) (lst(+i 1)) = t
   (loop (+i 2)))) lst)))
   (exit #f)))))
```

3)- Plugin with call/cc

```
(define (reverse-by-2 lerase lst)
  (call/cc
    (lambda (exit)
      (cond
        ((null? lst) '())
        ((null? (cdr lst)) (exit #f))
        (else (cons (cadr lst) (cons (car lst)
                                     (reverse-by-2 lerase (cddr lst))))))))
```

4)- interprète lexical et interprète dynamique :

1)- calcul la différence entre deux ~~valeurs~~ Valeurs

2)- (let ((x 2023)) (mystery 2024)) => 1

(let ((x 2025)) (mystery 2026)) => 1

les deux appelle domes = 1

3)- les causes d'erreurs possibles sont :

Si x ou y ne sont pas des ~~nombres~~

---

5)- Clôture lexicale :

~~(define (created-cave-wins init name~~

(define (created-cave-wins name Value)

(let loop ((init name)

(number Value))

(cond

(

1)- Prelude :

- Reverse2 Rune :

```
(define (reverse-by-2 Rune lst)
  (cond
    ((null? lst) '())
    ((null? (cdr lst)) '())
    (else (cons (cadr lst) (cons (car lst)
                                 (reverse-by-2 Rune (cddr lst)))))))
```

- Reverse - by 2 / last :

```
(define (reverse-by-2 last lst)
  (cond
    ((null? lst) '())
    ((null? (cdr lst)) lst)
    (else (cons (cadr lst) (cons (car lst)
                                 (reverse-by-2 last lst (cddr lst)))))))
```

- Reverse - by - 2 / fixed e n6 :

```
(define (reverse-by-2 fixed lst n6)
(if (< n6 (length lst))
    (cond
      ((null? lst) '())
      ((null? (cadr lst)) '())
      (else (cons (cadr lst) (cons (car lst)
                                   (reverse-by-2 fixed lst n6)))))
    (cond
      ((null? lst) '())
      ((null? (cdr lst)) '())
      (else (append (cons (cadr lst) (cons (car lst)
                                           (reverse-by-2 fixed lst n6)))
                    (make-list
                      (- n6 (length lst))
                    )))))
```

5)- Postludio :

1)- la fonction Strange-Function ~~[illegible]~~ : ~~retourne~~ elle fait l'inverse ~~des deux premiers éléments de l puis des~~ des éléments deux par deux et si la liste est impaire en laisse le dernier élément comme il-est

2)- la variable a-continuation représente : ~~[illegible]~~ la fin de call/cc, c'est à dire valeur de sortie du programe (exit)

3)- elle est modifié dans le cas que si la liste est ~~impaire~~ le nombre impaire

4)- (Strange-Function '(tuesday 21 1 2025 today)) =>
(21 tuesday 2025 1 today)

~~[illegible]~~

(a-continuation '(bye bye)) => (21 tuesday 2025 1 '(bye bye))

6)- Postludio :

1)- la function Strange-Function ~~[illegible]~~ ; ~~renverse~~ elle fait l'iverse ~~des deux premiers elements de l puis des~~ des éléments deux par deux et si la liste est impaire de nombre en laisse le dernier élément comme il-est

2)- la variable a-continuation représente : ~~[illegible]~~ la fin de call/cc, c'est à dire valeur de sortie du programe (exit)

3)- elle est modifié dans le cas que si la liste est ~~de nombre~~ ~~impaire~~ de nombre impaire

4)- (Strange-Function '(tuesday 21 1 2025 today)) =>
(21 tuesday 2025 1 today)

(a-continuation '(bye bye)) => (21 tuesday 2025 1 '(bye bye))