ELPL

```
-----------------------------------

let x be 5
let y be 10

print "Initial values: " x " and " y

if x is less than y then
  print "x is less than y"
otherwise
  print "x is not less than y"

repeat 3 times {
  print "Repeating" x
}

while x is less than 10 {
  print "Incrementing x:" x
  let x be x add 1
}

function greet {
  print "Hello from function!"
}

call greet

let result be y multiply x
print "x * y = " result

let condition be true
if condition is equal to true then
  print "Condition is true"
otherwise
  print "Condition is false"

let a be not false
print "a (not false) = " a



-------

Array nums be [1, 2, 3, 4]

let sum be 0
for i be 0 to 3 {
    let sum be i add nums[i]
}

print sum  / prints: 10

print nums[1]  / prints: 2

/ this is a single line comment
>This is a multi line
comment<
-------------


for i be 0 to 5 {
  print "Loop at i ="
  print i
  if i is equal to 3 then {
    print "Stopping at i = 3"
    stop
  }
}
print "Loop ended"

---output below----
```

```
Loop at i =
0
otherwise case
Loop at i =
1
otherwise case
Loop at i =
2
otherwise case
Loop at i =
3
Stopping at i = 3
Loop ended
```

```
---------------

Array nums be [2,7,11,15]


let target be 9


for i be 0 to 3{

for j be 1 to 3{

if nums[i] add nums[j] is equal to target then{

print "Indices: "
print i

print j

stop

} otherwise {

print "no found"

}

}

}
```

RECURSION

```
function factorial(n) {
    if n is equal to 0 then {
        return 1
    }
    return n multiply call factorial(n subtract 1)
}

call factorial(5)      / prints: //  120
```

RECURSION USING ARRAYS

```
Array memo be [0,0,0,0,0,0,0]

function fibonacci(n) {
    if n is less than 2 then {
        return n
    }

    if not (memo[n] is equal to 0) then {
        return memo[n]
    }

    let a be call fibonacci(n subtract 1)
    let b be call fibonacci(n subtract 2)
    let memo[n] be a add b
    return memo[n]
}

print "Fibonacci(6) = " call fibonacci(6)    // Fibonacci(6) = 8
```

Solving twoSum

```
Array nums be [2,7,11,15]

let target be 9


for i be 0 to 3{
 for j be 1 to 3 {
if nums[i] add nums[j] is equal to target then {
 print “indices: ” i j
stop / will break the loop
} otherwise {
  print “no pair found”
}

}
stop

} / prints : indices : 0 1
```

```
Array nums be [1,1,1,1]


let sum be 0

for i be 0 to 3{
 let sum be sum add nums[i]
}

print sum  / output : 4
```

```
let flag be false

if not flag then {
    print "Flag is false"
} otherwise {
    print "Flag is true"
}

let a be true
let b be false

if a and not b then {
    print "Logical expression works!"
}
```

----

solving twoSum using Built-In function length

```
Array nums be [2, 7, 11, 15]
let target be 9
let found be false

for i be 0 to length(nums) subtract 1 {
    for j be i add 1 to length(nums) subtract 1 {
        let sums be nums[i] add nums[j]
        print "checking pair: " i j " sum: " sums
        if sums is equal to target then {
            print "indices: " i j
            let found be true
            stop
        } otherwise{
        print "pair no found"
        }
    }

        stop

}

----------------------------------------
Array nums be [2, 7, 11, 15]
let target be 9
let found be false
Array result be [-1, -1]

for i be 0 to length(nums) subtract 1 {
    for j be i add 1 to length(nums) subtract 1 {
        let sum be nums[i] add nums[j]
        print "checking pair:" i j "sum:" sum

        if sum is equal to target then {
            let found be true
            let result[0] be i
            let result[1] be j
            stop
        }
    }
    if found is equal to true then {
        stop
    }
}

if found is equal to true then {
    print "indices: " result[0] result[1]
} otherwise {
    print "no pair found"
}

-----------------------------------------
```

REVERSING AN ARRAYs

```
Array nums be [3, 9, 12,11]
Array reversed be []

for i be 0 to length(nums) subtract 1 {
    let reversed[i] be nums[length(nums) subtract 1 subtract i]
}

print reversed
```

IN place reversal

```
Array nums be [2, 7, 11, 15]
let n be length(nums)
let mid be n divide 2

for i be 0 to mid subtract 1 {
    let temp be nums[i]
    let nums[i] be nums[n subtract 1 subtract i]
    let nums[n subtract 1 subtract i] be temp
}

print nums
```

Recursion sum of Arrays

```
function sumArray(arr, i) {
    if i is equal to length(arr) then {
        return 0
    }
    return arr[i] add call sumArray(arr, i add 1)
}

Array nums be [1, 2, 3, 4, 5]
call sumArray(nums, 0)
```

More recursive function

```
function power(x, n) {
    if n is equal to 0 then {
        return 1
    }
    return x multiply call power(x, n subtract 1)
}

call power(2, 4) / should print 16
```