1. Bindings and Nested Subprograms
    (1) In this case, I always need memory and do not want to deallocate it, I prefer to adopting static, because the object returned from a subprogram is always nonlocal to the subprogram.
    Of course, I can adopt heap to free store and allocate the memory, but it is inefficient and unreliable (memory leak).
    (2) In this case, I need to use the memory dynamically, or explicitly manage the memory, I prefer to adopting heap, and variable created on the heap are accessible anywhere in the program.
    (3) Yes, it will change the answer. I change to use stack. In stack, Arguments to be passed to subsequent routines lie at the top of the frame, where the callee can easily find them.
    (4) Yes, it may be parallel. While some variables update in reference globally in heap, some variables in stack update locally in upper frame.
    (5) Selected language: Python.
    In python, if we want to return dynamic object from subprograms, we can define a function first, and then pass the parameter when calling the function.
    If we want to return subprogram from subprograms, we can define a class to let subprogram return from other subprograms.

2. Nested Procedures and Runtime Bindings

a.

innermost(0)
innermost(1)
innermost(2)
innermost(3)
inner(3)
outer()
main

b.

b and f locate at innermost(0)
c and d locate at inner(3)

3. Parameter Passing
Assumption: Array indexing is zero-based.

(a) Call-by-Name

Initialization: a1 = 0 , a2 =3 , tmp =3

Iteration 1: tmp = 3 + A[1] = 3 + 3 = 6

Iteration 2: tmp = 6 + A[2] = 6 + 2 = 8

Iteration 3: tmp = 8 + A[3] = 8 + 7 = 15

(b) Call-by-Value

Initialization: a1 = 0 , a2 =3 , tmp =3

Iteration 1: tmp = 6

Iteration 2: tmp = 9

Iteration 3: tmp = 12

4. Lambda Calculus

(a)

$$\left(\lambda\mathrm{xyz.}\left(xz(yz)\right)\right)(\lambda\mathrm{xy.x})(\lambda\mathrm{xy.x})$$

$$\Rightarrow\ \beta[\lambda\mathrm{x}\to(\lambda\mathrm{xy.x})]\left(\lambda\mathrm{xyz.}\left(\mathrm{xz(yz)}\right)\right)$$

$$\Rightarrow\left(\lambda\mathrm{yz.}\left((\lambda\mathrm{xy.x})\mathrm{z(yz)}\right)\right)(\lambda\mathrm{xy.x})$$

$$\Rightarrow\beta[\lambda\mathrm{y}\to(\lambda\mathrm{xy.x})]\left(\lambda\mathrm{yz.}\left((\lambda\mathrm{xy.x})\mathrm{z(yz)}\right)\right)$$

$$\Rightarrow\left(\lambda\mathrm{z.}\left((\lambda\mathrm{xy.x})\mathrm{z(yz)}\right)\right)$$

$$\Rightarrow\beta[\lambda\mathrm{x}\to\mathrm{z}]\left(\lambda\mathrm{z.}\left((\lambda\mathrm{xy.x})\mathrm{z(yz)}\right)\right)$$

$$\Rightarrow\left(\lambda\mathrm{z.}\left((\lambda\mathrm{y.z})(\mathrm{yz})\right)\right)$$

$$\Rightarrow\beta[\lambda\mathrm{y}\to(\mathrm{yz})]\left(\lambda\mathrm{z.}\left((\lambda\mathrm{y.z})(\mathrm{yz})\right)\right)$$

$$\Rightarrow\lambda\mathrm{z.z}$$

(b) $(\lambda\mathrm{x} * \mathrm{x\,x})(+2\ 3)$

<u>Normal Order:</u>

$(\lambda\mathrm{x} * \mathrm{x\,x})5$

$(* \ 5\ 5)$

25

<u>Applicative Order:</u>

(*(+ 2 3)(+ 2 3))

(*5 (+ 2 3))

(*5 5)

25

(c)

i. $(\lambda xy.\, yx)(\lambda x.\, xy)$

Alpha-conversion is required, because y is free in the right hand side but bound on the left hand side.

Correct way:

$(\lambda xy.\, yx)(\lambda x.\, xy)$

$\Rightarrow \alpha[\lambda y \rightarrow \lambda z](\lambda xz.\, zx)(\lambda x.\, xy)$

$\Rightarrow (\lambda z.\, z(\lambda x.\, xy))$

Incorrect way:

$(\lambda xy.\, yx)(\lambda x.\, xy)$

$\Rightarrow (\lambda y.\, y(\lambda x.\, xy))$

Therefore, the two expression is not identical.

ii. $(\lambda x.\, xz)(\lambda xz.\, xy)$

Alpha-conversion is not required in the beginning, because y is free on the right hand side and not bound in the left hand side. Nevertheless, when we proceed to reduce one more step, alpha-conversion is required.

Correct way:

$(\lambda x.\, xz)(\lambda xz.\, xy)$

$\Rightarrow \alpha[\lambda z \rightarrow \lambda w](\lambda xz.\, xy)z$

$\Rightarrow (\lambda xw.\, xy)z$

$\Rightarrow \beta(\lambda w.\, zy)$

Incorrect way:

$(\lambda x.\, xz)(\lambda xz.\, xy)$

$\Rightarrow \beta(\lambda xz.\, xy)z$

$\Longrightarrow (\lambda z.\, zy)$

Thus, these two expressions are not identical.

iii. $(\lambda x.\, xy)(\lambda x.\, x)$

Since there is no free variable on the right hand side, alpha-conversion is not required.

$(\lambda x.\, xy)(\lambda x.\, x)$

$\Rightarrow \beta(\lambda x.\, x)y$

$\Rightarrow \beta y$

(d) (+ 1 1) = 2

$(\text{PLUS } 1\ 1) = \big(\lambda mn\ fx. mf(n\ f\ x)\big)(\lambda fx. fx)(\lambda fx. fx)$

$= \big(\lambda m. \lambda n. \lambda f. \lambda x. mf(n\ f\ x)\big)(\lambda fx. fx)(\lambda fx. fx)$

$\Rightarrow \beta(\lambda n. \lambda f. \lambda x. (\lambda fx. fx)f(n\ f\ x))(\lambda fx. fx)$

$= (\lambda n. \lambda f. \lambda x. (\lambda f. \lambda x. \lambda x)f(n\ f\ x))$

$\Rightarrow \beta(\lambda n. \lambda f. \lambda x. (f(n\ f\ x))(\lambda fx. fx)$

$\Rightarrow \beta\left(\lambda f. \lambda x\left(f\big((\lambda fx. fx)fx\big)\right)\right)$

$= \left(\lambda f. \lambda x. \left(f\big((\lambda f. \lambda x. fx)fx\big)\right)\right)$

$\Rightarrow \beta\left(\lambda f. \lambda x. \big(f(fx)\big)\right)$

$= \big(\lambda fx. f(fx)\big)$

$= 2$