60 câu:

* 6 câu midterm
* 10 câu btl3
* 10 câu btl4 (2 câu dc gợi ý)
* 34 câu lý thuyết
* 11 câu Naming
* 8 câu Static (5)/Dynamic (3) Scope | Referencing env (o1 o2 o3 o nào có hiệu lực) Code dài, gộp luôn control program hỏi chung

**Shallow binding / Deep binding, param passing**

* 3 câu Error (Alias, Garbage, Dangling Reference)
* Scope (in Python, hên đéo có Pascal, Scala :)))
* 8 câu Type
  + 3 câu Inference
  + Học kĩ int (2 complement, decimal -> binary), float (IEEE-754), string (3 string implementations, “Toi di hoc”), array (tính address), struct (alignment), set (set bits)
* 5 câu JVM (học thuộc mấy cái lệnh jasmine)

Control:

* Expression
* 1 prefix postfix (Polish, Cambridge…)
* Short circuit evaluation (and, or, \*) có áp dụng vô JVM

ex:

foo1(&a, &b) {a++, b++, return false}

foo2(&a, &b) {c++, d++, return j đó)

foo1(a,b) && foo2(c,d)

=> Chỉ có a, b bị thay đổi

* Stmt (TNBD kêu câu stmt dễ vãi lồn, nhảm nhí, đéo cần ôn)
* Program (Các cơ chế truyền tham số)

Python case study:

* Coi kĩ scope, global, local, non-local (Coi kĩ ví dụ trong slide)

Đề nay ko có pass by name

# Ôn Duy

## question 1

# 

**\*Huge note: “Pascal-like”, not Pascal -> foo can still call foo1.**

Tựa ??? -> Theo lý thuyết mà làm

**Solution:**

**foo1 //4**

**y //1**

**foo //2**

**z //3**

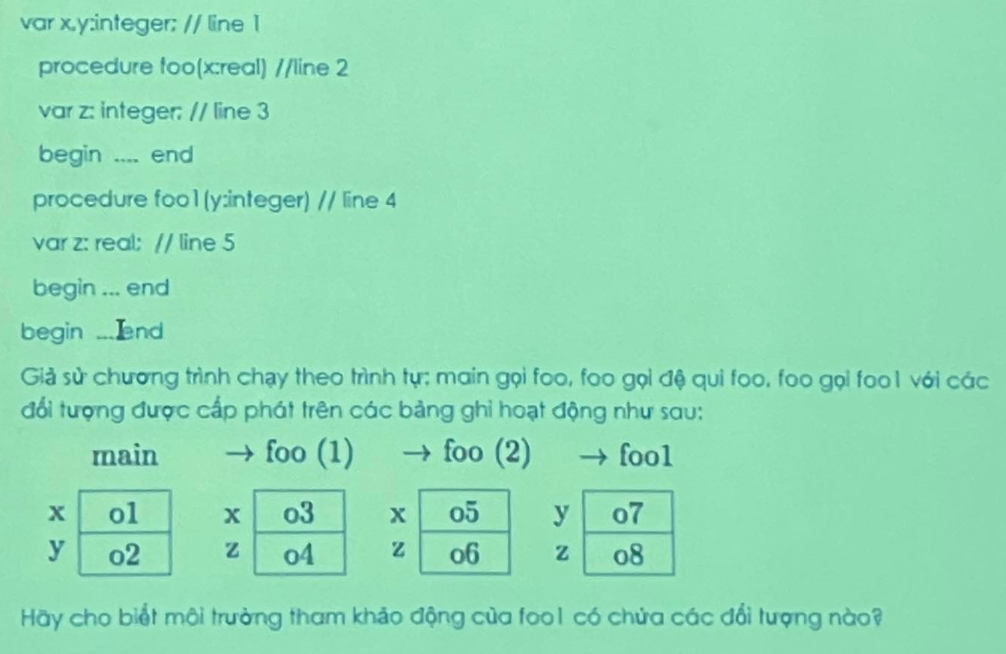
**x //2**

**block** structure language -> static scoping language (Java, C++, C, C#, Pascal)

**scripting** language -> Dynamic scope

1. Xác định static hay dynamic scope (Ngôn ngữ có cấu trúc khối) -> Static
2. Nếu static -> Đọc code
3. Nếu dynamic -> Nhin quy trình gọi hàm
4. Tên hàm thuộc scope cha, vô hàm khai báo lại
5. In C++, Pascal, in foo(), foo1() cannot be call if foo1() is declared after foo(). Can overcome by having foo1() declared inside a class

## question 2

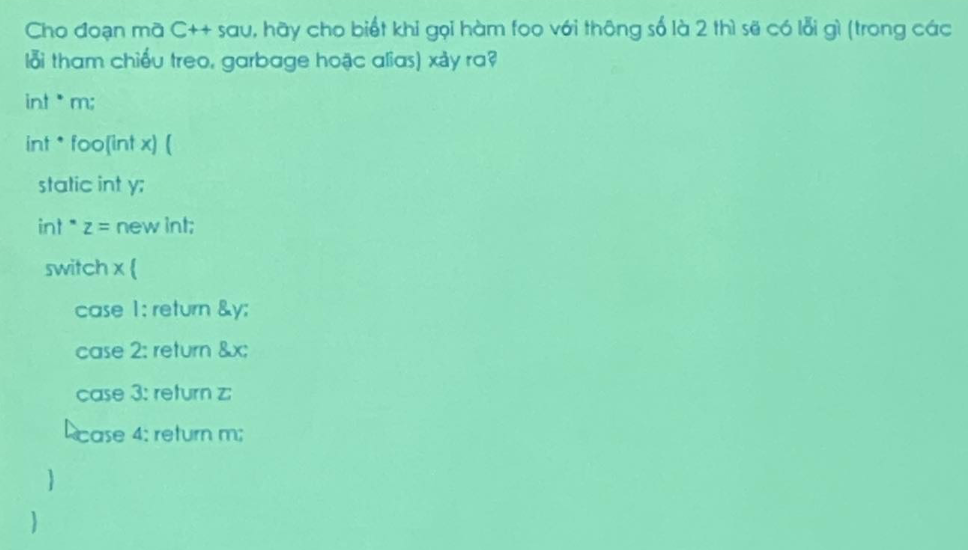


**Solution: o1, o7, o8**

**Đề thi sẽ cho code vô begin … end -> Sau khi gọi code đó, in giá trị biến -> in giá trị gì ? (Chỗ khó của phần Name)**

1. Static -> Bỏ qua bảng

## question 3



Trong đề sẽ dễ hơn vầy ???

Dấu hiệu nhận biết:

* Dangling reference (Tham chiếu treo :)) ): Cầm address nhưng ko còn j ở addr
* Garbage: Có object nhưng ko có/còn gì trỏ tới nó

int \*a = new int;

a = … // object new int không thể nào dùng dc nữa

* Alias: Nhiều tên tham chiếu tới 1 object

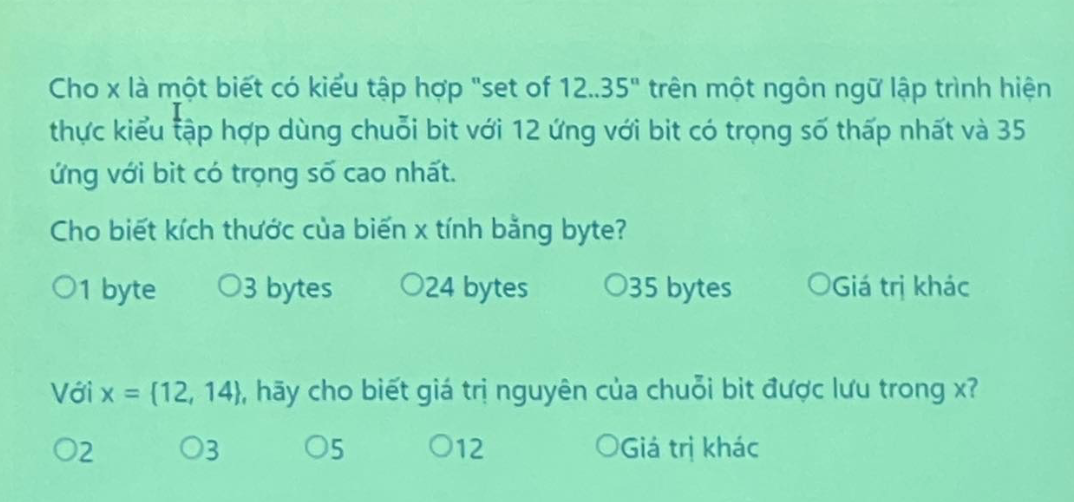
Case 1: Garbage (new int)

Case 2: Dangling reference (&x), Garbage (new int)

Case 3: No error

Case 4: Garbage (new int)

## question 4



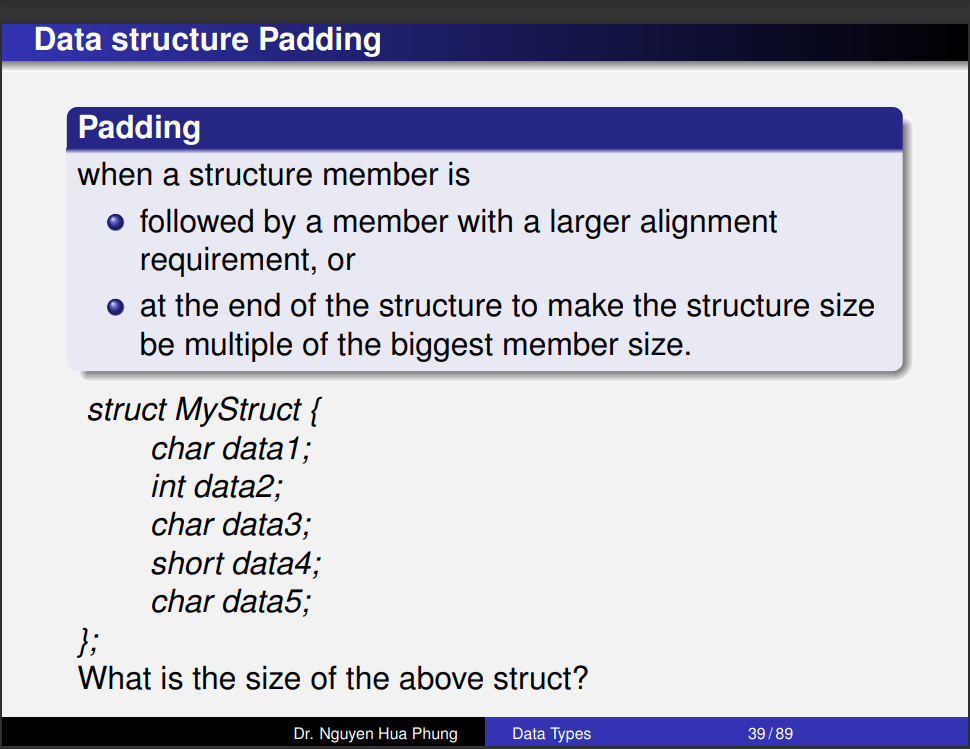
Đề thi có câu ở dưới

**Solution:**

**3 (35 - 12 + 1 bits = 24b = 3B)**

**5 (0000 0000 0000 0000 0000 0101)**

## question 5

****

**Solution: 16 bytes**

**0 data 1**

**1 pad**

**2 pad**

**3 pad**

**4 data2**

**5 …D**

**6 …**

**7 …**

**8 data3**

**9 pad**

**10 data4**

**11 …**

**12 data5**

**13 pad**

**14 pad**

**15 pad để size = 16 chia hết cho size(data2)**

**struct Struct2 {**

**char data1;**

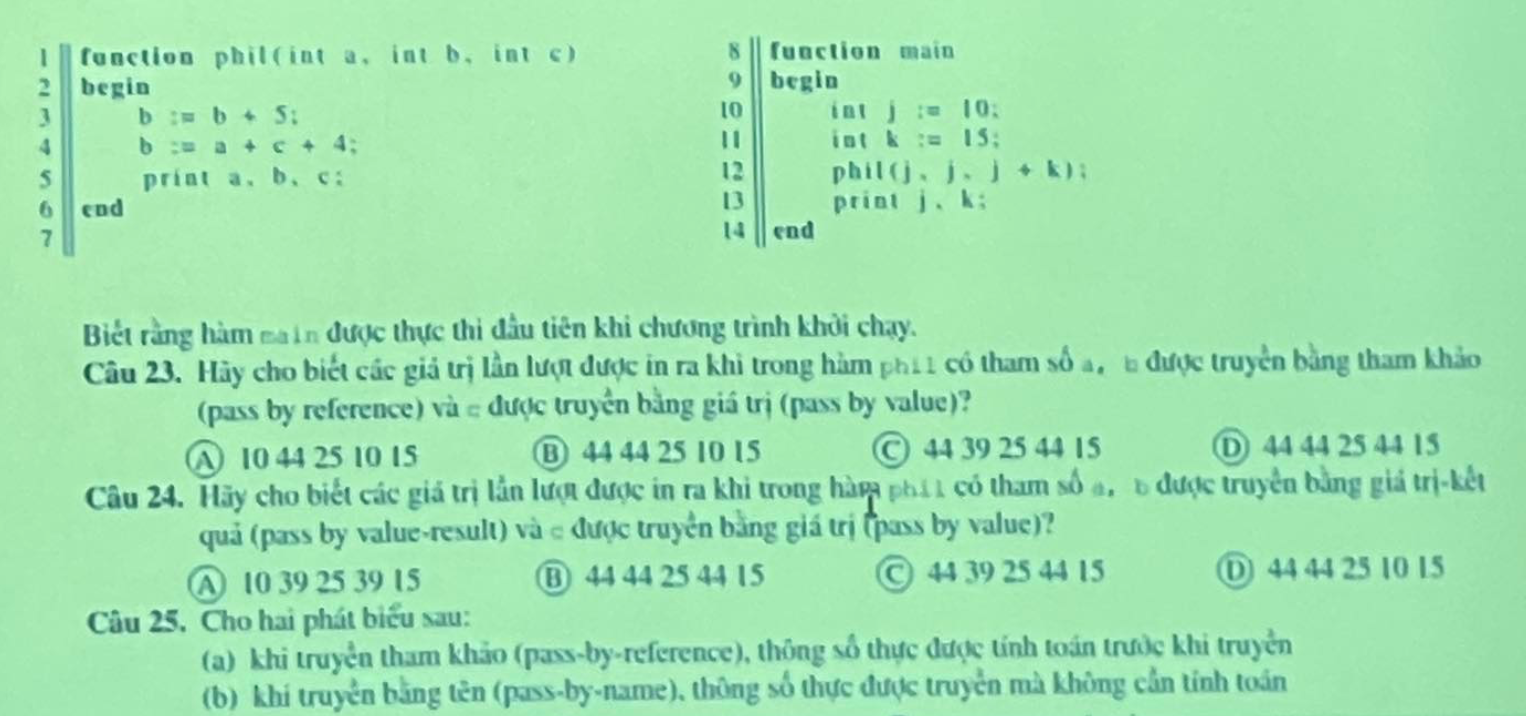
**MyStruct data2;**

**}**

**Cái này size 20 bytes ??? Tự tìm hiểu sau**

**i think so, char (1byte) + 3 pad + myStruct (16 bytes) = 20bytes**

## Đề 231

****

phil(j, j, j + k)

23)

b (j) = 10 + 5 = 15

b (j) = 15 + 25 + 4 = 44

print j, j, 15 = **44 44 25**

print j, k = **44 15**

24)

Pass by value-result: Tạo ra ô nhớ mới

a = 10

b = 10

c = 25

**b = b + 5 = 15**

**b = a + c + 4 = 39**

**print(a, b, c) -> 10 39 25**

**end method ->**

**j = a = 10**

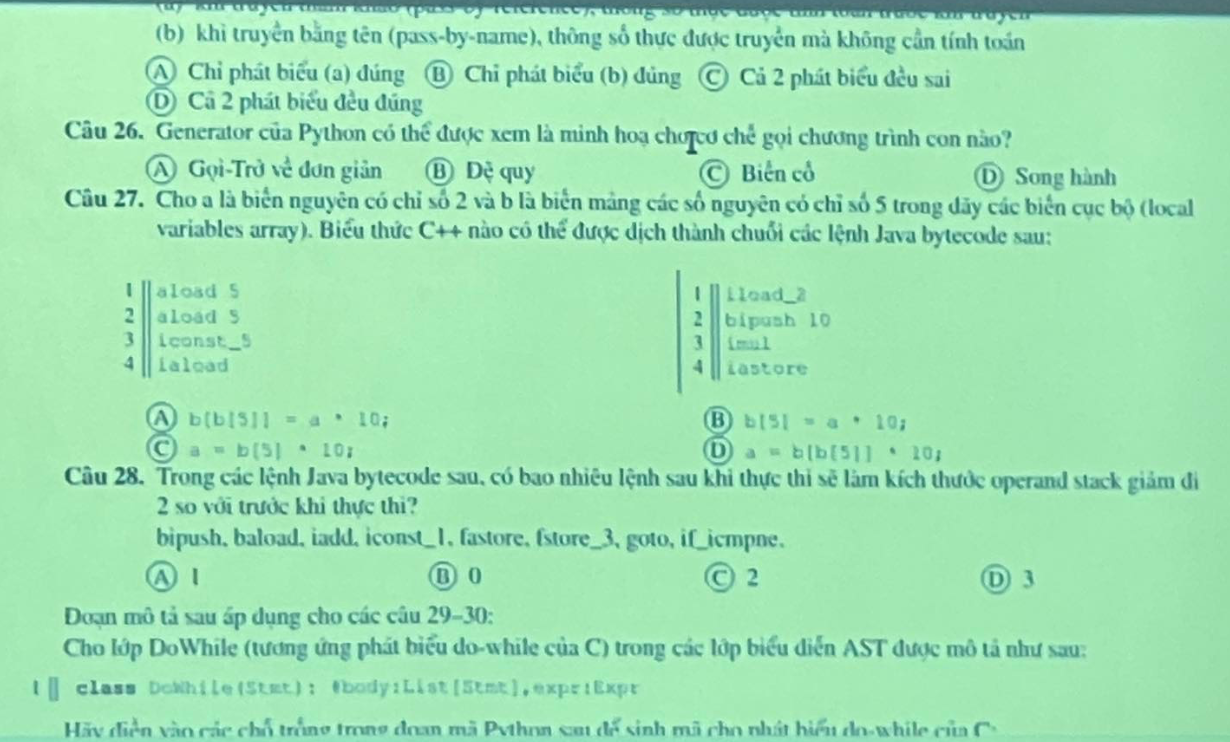
**j = b = 39 (a and b are passed by value-result) Assign ngược lại, từ trái qua phải**

**print(j, k) -> 39, 15**

**Solution:**

**23) D (Vua trắc nghiệm)**

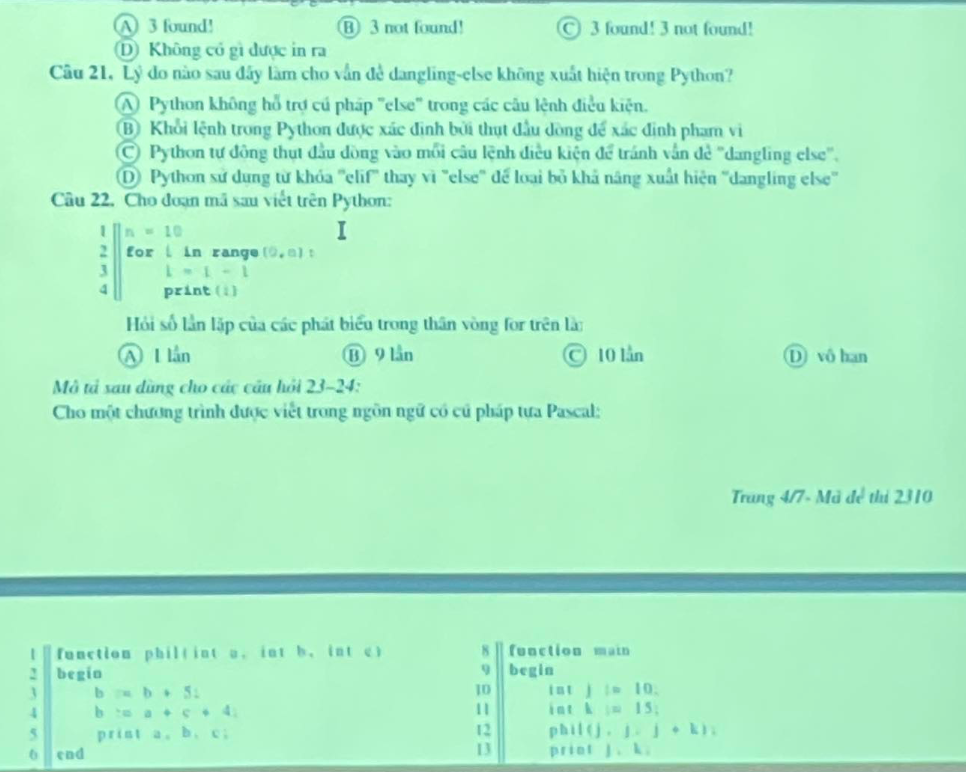
**24) A**

****

**Solution:**

**27) A**

**28) Chưa biết (A)**

****

**21) B**

**22) C (dùng for x in \_ => x là biến được bảo vệ, không thể thay đổi, nội dung thi CodeGen)**

=============================================================

# Ốn Lùng

## question 1

Given the following **Python** code:

x = "abcdef"

y = 1.25

z = x + y

The expression x + y causes a type error. When does this type error happen

a) Compiling time

b) Language design time

**c) Runtime**

d) Language implementation

**Explain: in python, type of variable is defined at runtime**

**In ZCode:** Compile time

## question 2

def foo(x, y, z) = y(z(x))

Write the type expression of function foo?

* Product type has higher priority than function type (\* is higher than -> then you don’t need ‘()’ )
* (...)->T3 : redundant

T1\*(**T2**->T3)\*(T1->T2)->T3

## question 3

**block** structure lang -> static scoping language (Java, C++, C, C#, Pascal)

scripting language -> Dynamic scope

var a,b,c

procedure sub1(a)

var b,d

body

procedure sub2(con cá chà pặc của t bị xóa hoài thế)

procedure sub3(c)

var d

body

body

body

sub1 -> sub2 -> sub3 -> sub1 -> sub2

o1 o4 o5 o7 o10

o2 o6 o8

o3 o9

**Static scope -> Cannot use ‘d’**

**Solution: b, c in main, a in sub2**

**dynamic scoping solution: o5,o8,o9,10**

**static scoping solution: o2,o3,o6 *(Need verification)***

## question 4

int x = 0;

int \*a = &x;

int \*b = a;

int d = \*a;

Write the aliases created in the above code

a++ → b doesn’t change, because they’re different (khác memory cell)



a++; //a=a+4

**Solution: \*a, \*b, x**

## question 5

IEEE-754

1 bit of sign area

3 bit of exponent area

5 bit of fraction area

Write 5.3 in IEEE-754

**SOL:**

**5.3 → 5 + 0.3**

**sign bit: 0 (5.3 > 0)**

**5 → Binary: 101**

**Top down**

**5 1**

**2 0**

**1 1**

**0**

**Bottom up**

**0.3 0.6**

**0.6 1.2**

**0.2 0.4**

**… (Stop because of the mantissa size)**

**bias: 2^(#size - 1) - 1 = 3 (2^(#exp\_bits - 1) - 1)**

**101.01001100**

**→ 1.0101001100 x 2^2 → real\_exp=2**

* **mantissa: 0101001100**
* **real exp: 2**

**fake\_exp = bias + real\_exp = 3 + 2 = 5 → Binary: 101**

**ANSWER: 0 101 01010**

**e.g. 5.3**

**5 = 101**

**0.3 = 010**

**{how to get 010**

**0.3\*2 = 0.6 -> 0**

**0.6\*2 = 1.2 -> 1**

**0.2\*2 = 0.4 -> 0}**

**e.g 0.5 = 0.1**

**{how to get 0.1**

**0.5\*2 = 1.0 -> 1}**

**-> 5.3 = 101.010**

**101.010 \* 2^0 = 1.01010 \* 2^2**

**bias = 2^(expo size - 1) -1 = 2^(3 - 1) -1 = 3**

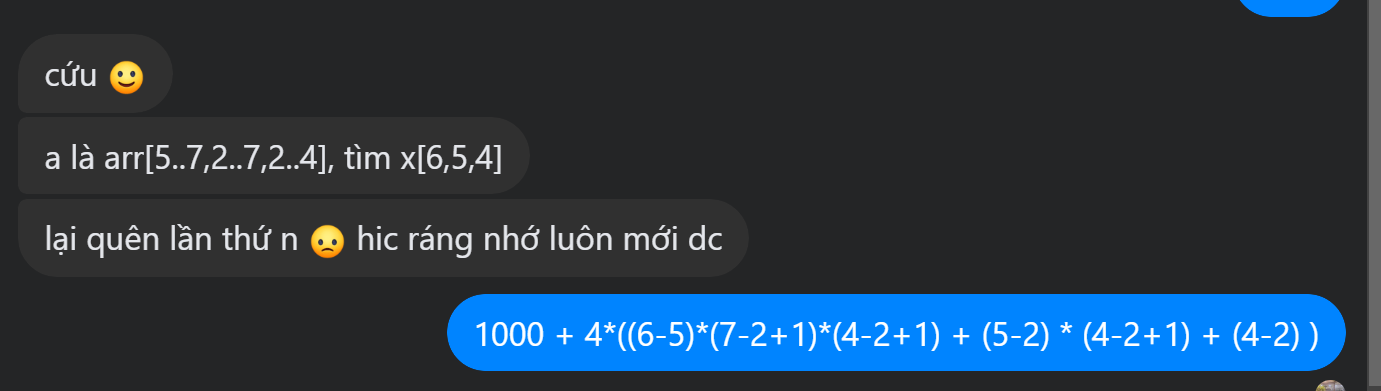
**exponent bit = expo + bias = 2 + 3 = 5 = 101**

**IEEE of 5.3: 0|101|01010**

## question 6

init address = 1000

int size = 4



**row major:** 1000 + 4 \* ((6-5)(7-2+1)(4-2+1) + (5-2)\*(4-2+1) + (4-2))

**column major:** 1000 + 4\* ((4-2)\*(7-5+1)\*(7-2+1) + (5-2)\*(7-5+1) + (6-5))

Generally:

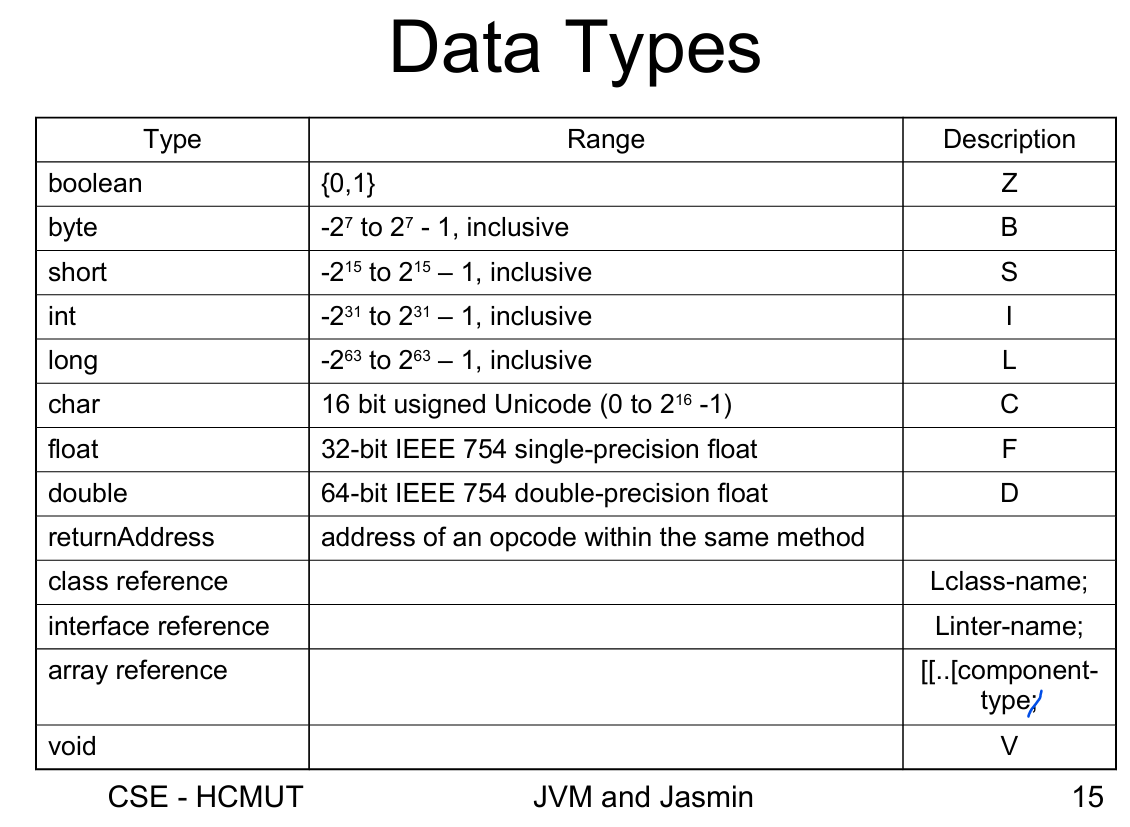
Row major? Traverse cố định row ĐẦU TIÊN: ([5,2,2] [5,2,3] [5,2,4] [5,3,2] [5,3,3] [5,3,4] [5,4,2] [5,4,3]...)

Column major? Traverse cố định column CUỐI CÙNG:

([5,2,2] [6,2,2] [7,2,2] [5,3,2] [6,3,2] [7,3,2] [5,4,2] [6,4,2] [7,4,2] …)

Row: 2 phần tử cạnh nhau khác ở index cuối

Column: 2 phần tử cạnh nhau khác ở index đầu



**Bảng bị sai: No semicolon after array reference**

Given the Java method declaration:

int foo(VD x[], boolean y[], float m)

where VD is a class name  
write jasmin code represent the function foo

**Solution: ([LVD;[ZF)I**

## question 7

try:

x = 7

return x

except Error:

x = x + 1

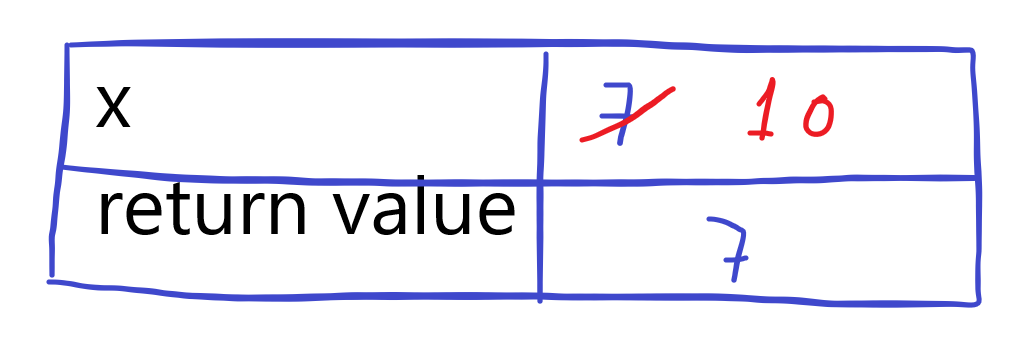
else:

x = x + 2

finally:

x = x + 3

**What value is returned by the above python code?**

****

**Solution: 7**

**In an alternate universe:**

try:

x = 7

return x

except Error:

return x + 1

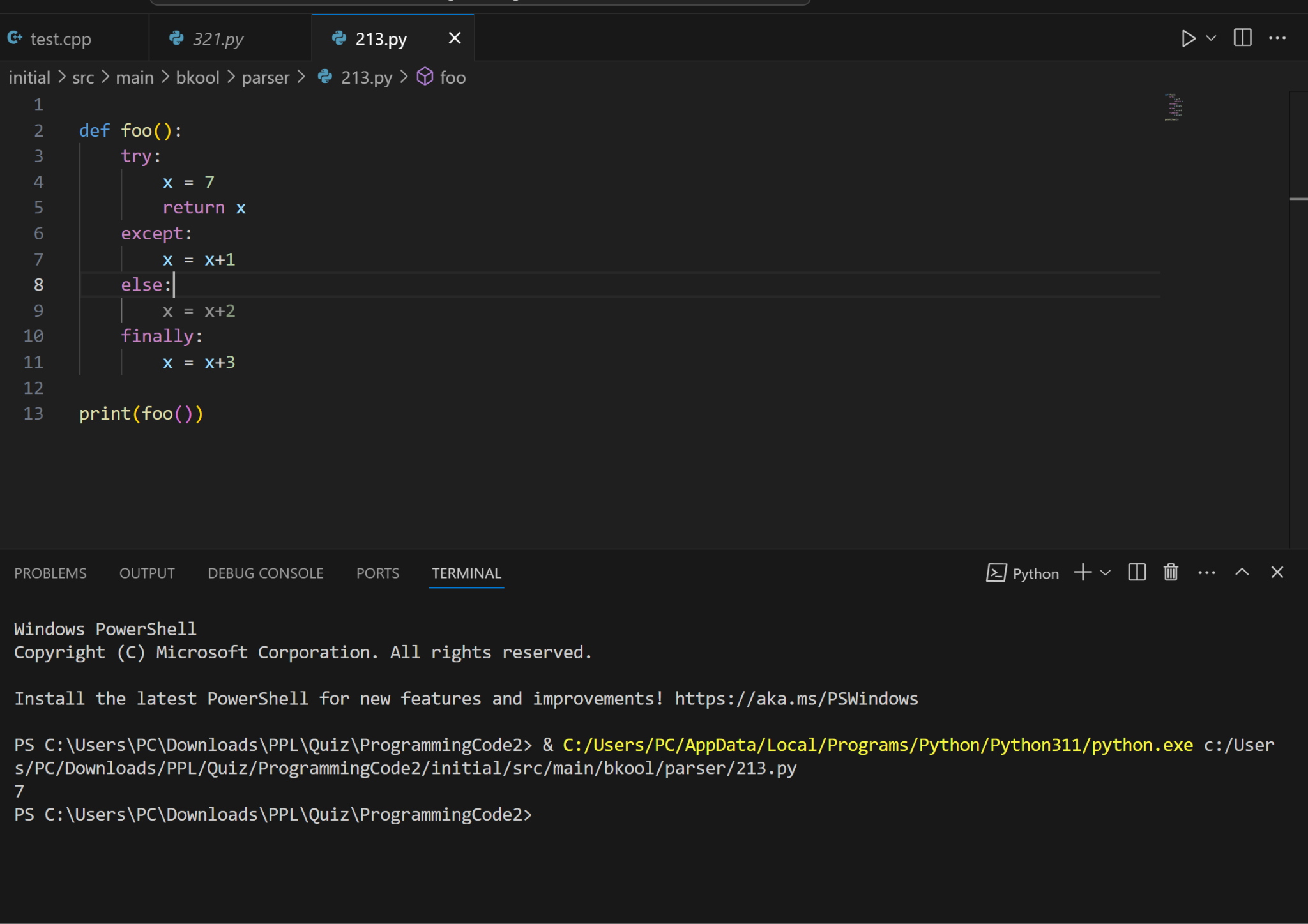
else:

return x + 2

finally:

return x + 3

=> Solution: 10

****

**Rule: except = else < finally (priority)**

Nếu có **return x + 3** thì mới return 10

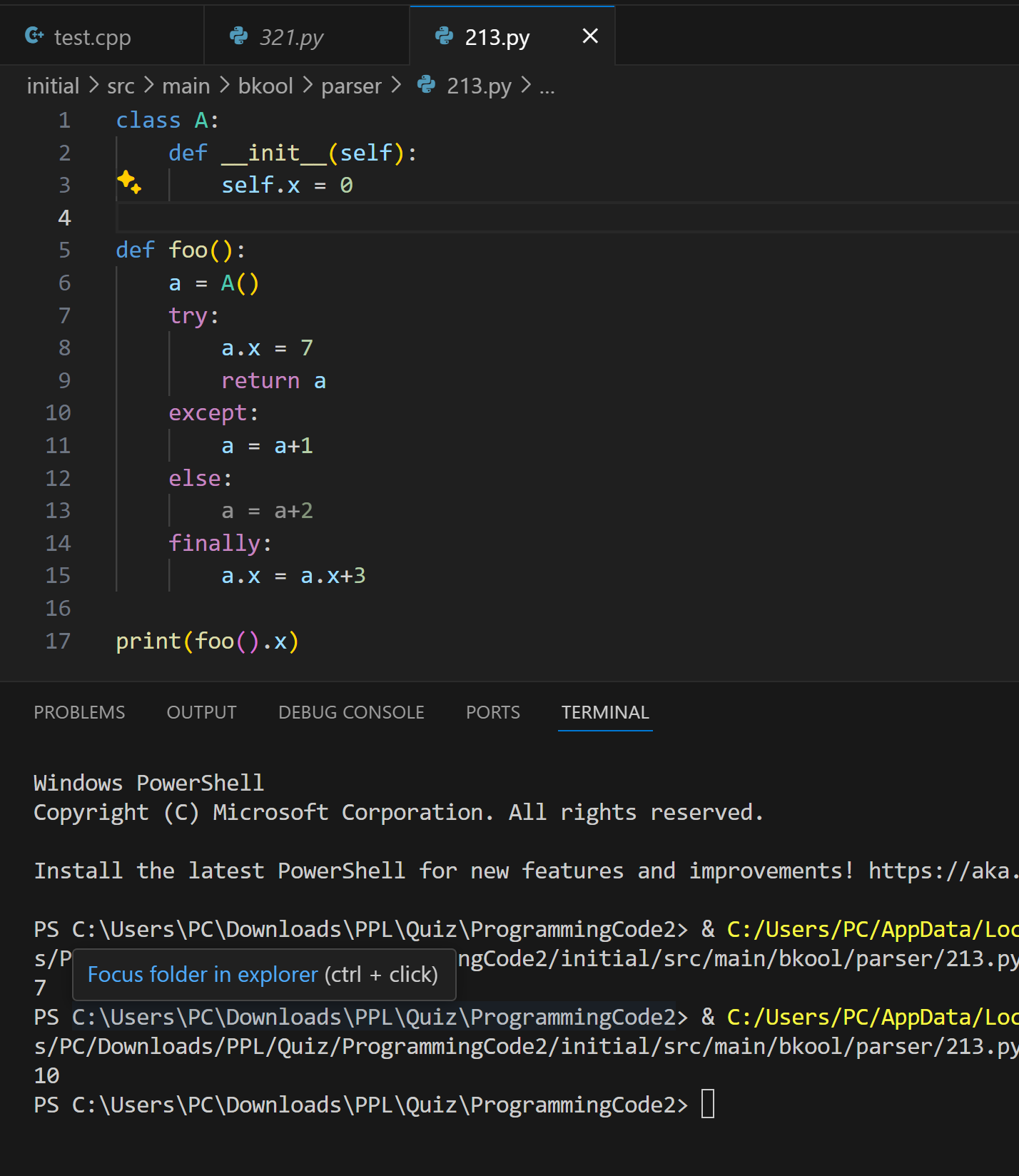
Vấn đề là: x = x + 3 doesn’t create a copy for return :)) → return x is still return 7

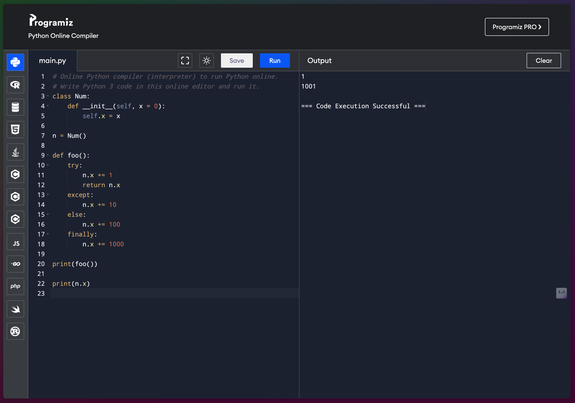
Kiểu cái finally nó sẽ tạo 1 cái copy của x chứ không tạo thẳng x

Damn Phùng chơi ác quá

***You just try in your code = I don’t remember***

***“And you cannot remember…” → Because “I don’t remember” 🙂***

******

******

## question 8

int \*a; //1

int\* foo(int\*& b) { //2

int c = 0; //3

b = new int; //4

return &c; //5

} //6

int main() { //7

int \*d = foo(a); //8

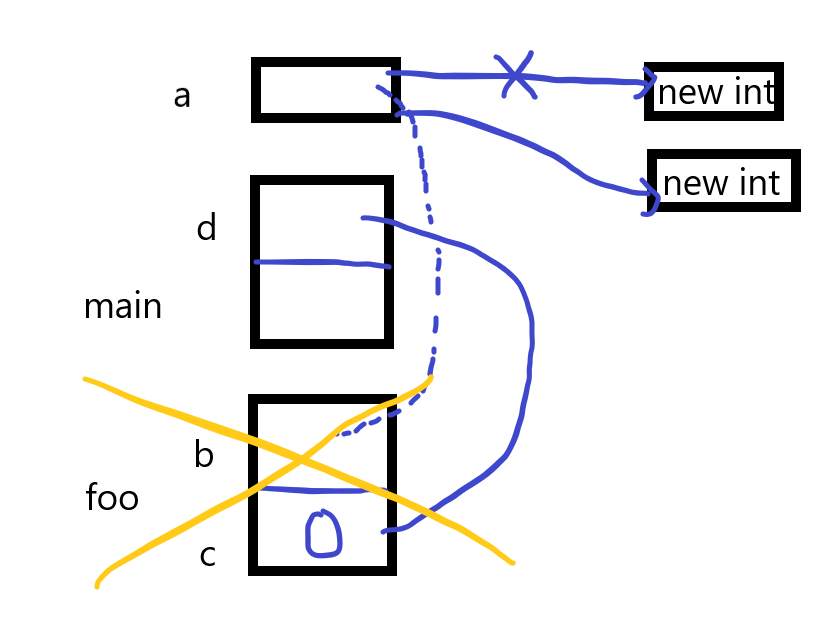
a = new int; //9

return \*d; //10

} //11

Choose the CORRECT statement?

* Aliases, Dangling references, Garbage,...



local of foo: b, c

a: pass by reference (có dấu &)

aliases: a and b //8

return &c → d is pointing to c

Thoát scope foo, c bị hủy → d đang bị dangling nhưng chưa có issue do chưa touch d

Chỉ khi access to d (dereference, for example) thì mới bị dangling, nếu không đụng chạm gì d thì nothing happen (không error luôn)

//9 → garbage

//10 → dangling (đụng tới d là cửng)

If remove //9, no garbage because after program execution, all memory will be deallocated by OS automatically

int\* foo(int\*& a) → no aliases (a ở ngoài bị shadow → không có different name nên không có aliases)

foo (){

int c

foo (c)

}

f(&a) → can’t use c from a

# TUTORIAL

non-local → start looking up from where it’s called

* Dynamic-scoping with Shallow-binding:
* Dynamic-scoping with Deep-binding:
* Static-scoping:

# FINAL EXAM

* Multiple choice
* MUST fill all questions even don’t know
* *NOT IMPORTANT HOW MANY QUESTIONS 🙂*
* Closed-book
* Most in second parts, still some questions from first part (First half of semester)

100 questions → Suppose we know 60 questions → Supposed to be 6pts (40 questions we should randomly choose)

* 🙂 YOU MUST PASS >= 15 questions or something or you get 0 🙂