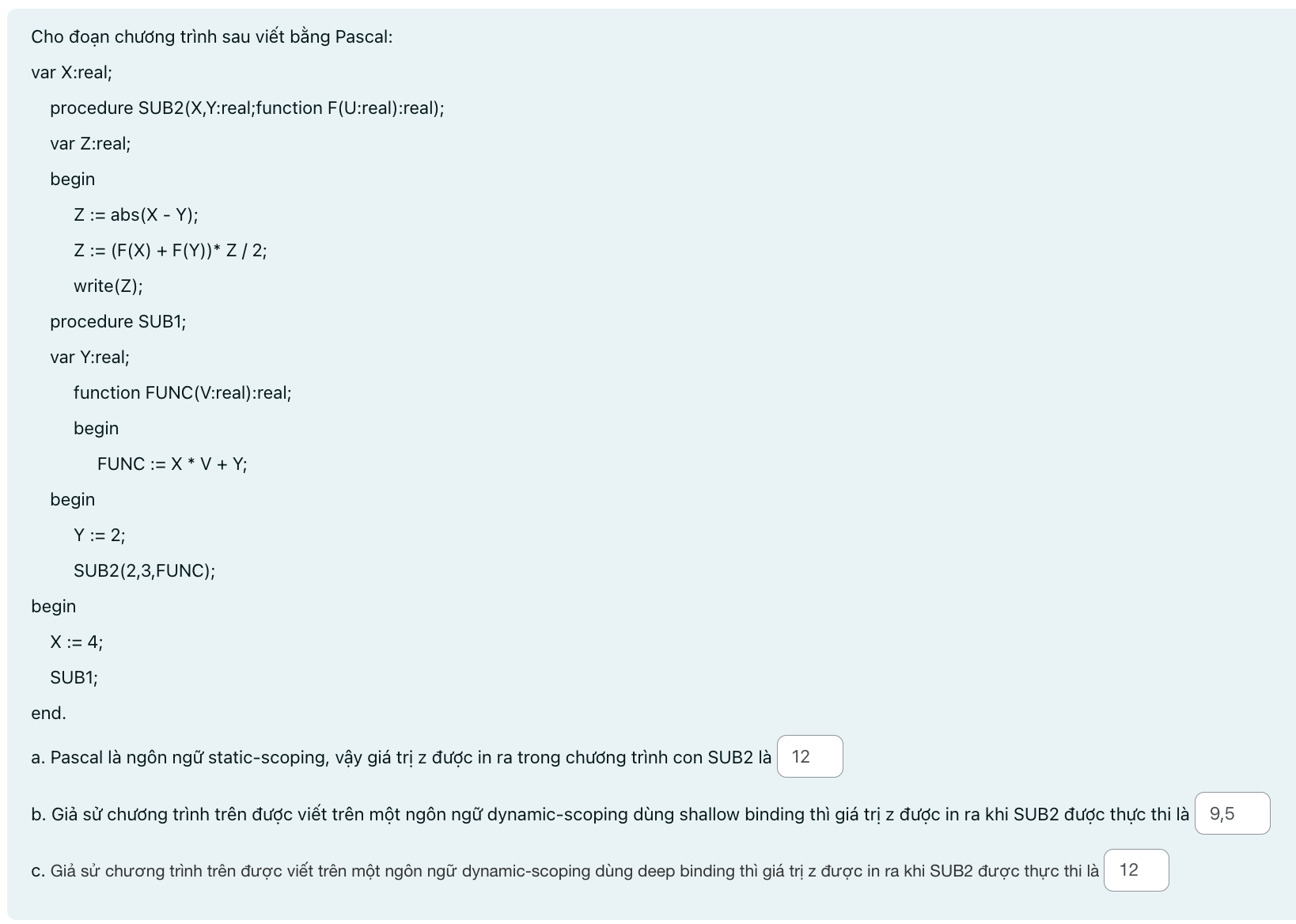
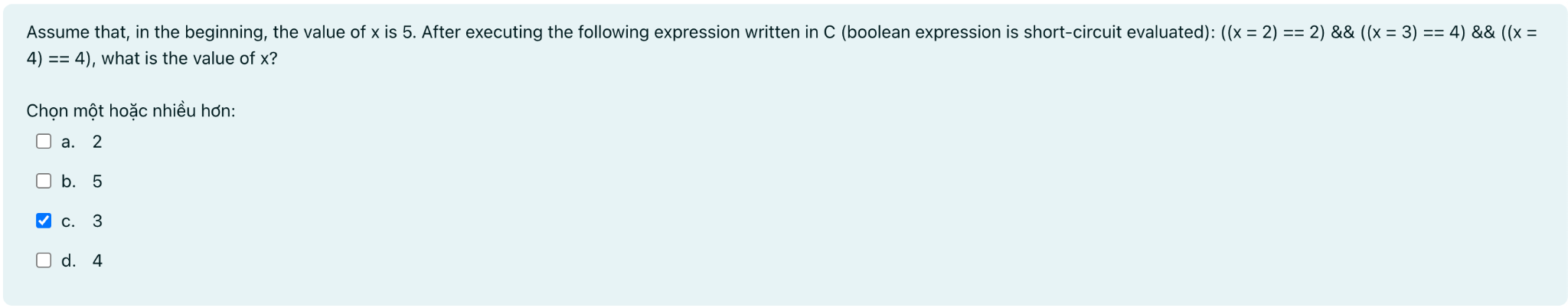
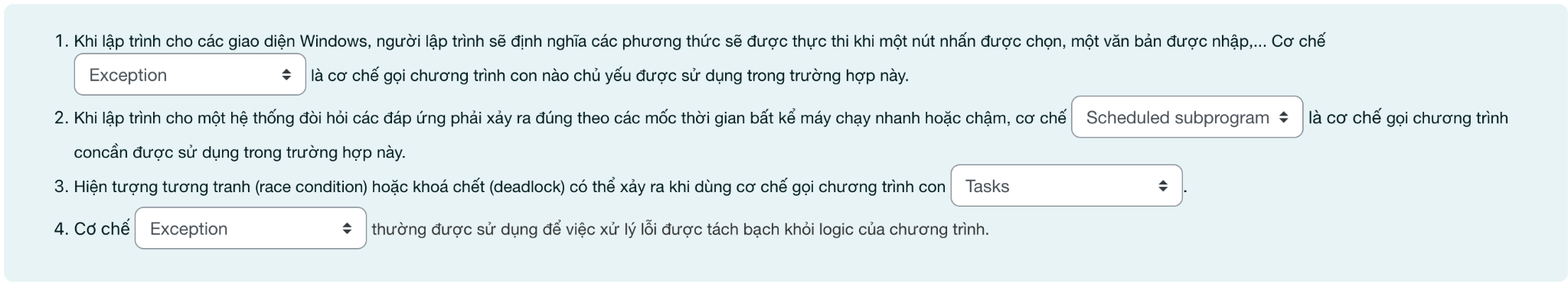
**1. Deep and shallow binding**



**2. Short-circuit**



**3. Ứng dụng về các cơ chế gọi chương trình con**

*(Lưu ý: phải học thêm kết hợp với Python)*

**4. Hiện thực con trỏ hàm trong các ngôn ngữ lập trình đã học (C++/ Python)**

*Hãy viết tương đương đoạn mã sau đây trong Python:*

void fun(int a)

{

printf("Value of a is %d\n", a);

}

int main()

{

// fun\_ptr is a pointer to function fun()

void (\*fun\_ptr)(int) = &fun;

/\* The above line is equivalent of following two

void (\*fun\_ptr)(int);

fun\_ptr = &fun;

\*/

// Invoking fun() using fun\_ptr

(\*fun\_ptr)(10);

return 0;

}

**5. Python variable scope with nonlocal, global**

def outer\_function():

global\_var = "I'm a global variable"

def inner\_function1():

nonlocal\_var = "I'm a nonlocal variable"

print("Inner Function 1 - Before Modification")

print("Global variable:", global\_var)

print("Nonlocal variable:", nonlocal\_var)

def inner\_function2():

nonlocal nonlocal\_var

nonlocal\_var = "Modified nonlocal variable in inner\_function2"

global nonlocal\_var\_global

nonlocal\_var\_global = "I'm a global variable accessed in inner\_function2"

inner\_function2()

print("\nInner Function 1 - After Modification")

print("Global variable:", global\_var)

print("Nonlocal variable:", nonlocal\_var)

print("Nonlocal variable (global):", nonlocal\_var\_global)

inner\_function1()

print("\nOuter Function")

print("Global variable:", global\_var)

outer\_function()

Trong ví dụ này, chúng ta có một outer\_function chứa hai hàm lồng nhau: inner\_function1 và inner\_function2. Đây là cách code hoạt động:

1. outer\_function được định nghĩa và khởi tạo biến global\_var là biến cục bộ.
2. inner\_function1 được định nghĩa bên trong outer\_function và khởi tạo biến nonlocal\_var là biến cục bộ. Sau đó, nó in ra các giá trị của global\_var và nonlocal\_var trước khi thay đổi.
3. inner\_function2 được định nghĩa bên trong inner\_function1. Nó sử dụng từ khóa nonlocal để thay đổi biến nonlocal\_var và từ khóa global để truy cập đến biến toàn cục mới được gọi là nonlocal\_var\_global.
4. inner\_function2 được gọi từ inner\_function1.
5. Sau khi inner\_function2 thay đổi các biến, inner\_function1 in ra các giá trị của global\_var, nonlocal\_var, và nonlocal\_var\_global.
6. Cuối cùng, outer\_function in ra giá trị của global\_var.

Kết quả khi chạy code:

Inner Function 1 - Before Modification

Global variable: I'm a global variable

Nonlocal variable: I'm a nonlocal variable

Inner Function 1 - After Modification

Global variable: I'm a global variable

Nonlocal variable: Modified nonlocal variable in inner\_function2

Nonlocal variable (global): I'm a global variable accessed in inner\_function2

Outer Function

Global variable: I'm a global variable

**6. Một số hiện thực về kiểu:**

*6.1. Kiểu tập hợp*

Xem x là một biến thuộc loại tập hợp "tập hợp từ 5 đến 20" được thể hiện dưới dạng một chuỗi bit, trong đó 5 tương ứng với bit có giá trị thấp nhất và 20 tương ứng với bit có giá trị cao nhất. Cho x = {6, 7, 8}, giá trị số nguyên tương ứng của x khi được ép kiểu thành kiểu số nguyên là gì?

**7. Truyền tham số trong C++:**

*Hãy cho biết kết quả của đoạn mã này và giải thích:*

#include <iostream>

void modifyPointer(int\* ptr) {

ptr = new int;

\*ptr = 100;

}

void modifyReference(int& ref) {

int\* ptr = new int;

\*ptr = 200;

ref = \*ptr;

}

int main() {

int value1 = 50;

int\* ptr1 = &value1;

modifyPointer(ptr1);

int value2 = 50;

int& ref = value2;

modifyReference(ref);

std::cout << "Pointer Value: " << \*ptr1 << std::endl;

std::cout << "Reference Value: " << ref << std::endl;

delete ptr1;

return 0;

}