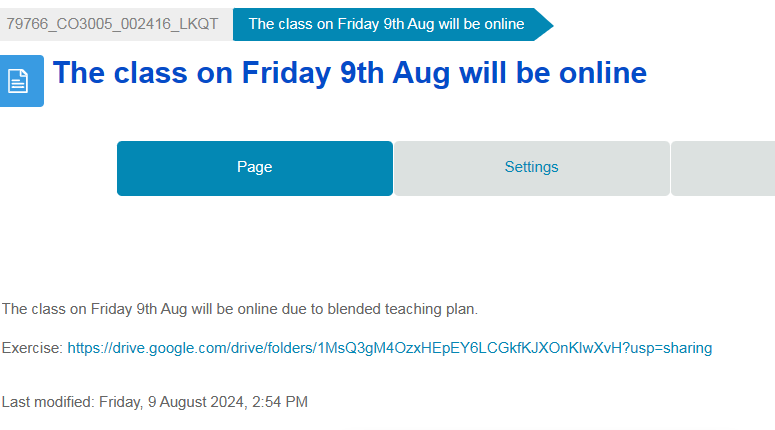
**Other works:** [https://drive.google.com/drive/folders/1MsQ3gM4OzxHEpEY6LCGkfKJXOnKIwXvH?usp=sharing](https://drive.google.com/drive/folders/1w3E6wpb2khBGzqUUjZnOcaiLVOApTJVn?usp=sharing" \t "https://lms.hcmut.edu.vn/mod/page/_blank)

**Meet: <https://www.youtube.com/watch?v=RrjA84Mr-9w>**

**Please go to LMS system to get questions**

**I will be back at 15:15**



**Question 1.** Which of the following Java bytecode instructions is used to return an integer from a method?

**- A) `ireturn`**

- B) `areturn`

- C) `dreturn`

- D) `freturn`

**Question 2.**Given a method declared in class Y, written in Java, as follows:

`public static void bar(double d, B b, int[] arr);`

Where `B` is a previously declared class. The minimum size of the local variable array needed is:

- A) 3

**- B) 4**

- C) 5

- D) 6

Double: 2 bytes

Int: 1 byte, consider only allocate

B: at least 1 byte

Because the JVM uses a 32-bit architecture for its stack frames and local variable arrays, any data type that is larger than 32 bits occupies more than one "slot" in the local variable array.

double is 64 bits so it takes 2 slot

**Question 3.** Consider the following Java code snippet:

```java

boolean result = a || (b && c);

```

The corresponding bytecode would be:

1. `iload\_1`a

2. `ifne Label1` if not equal 0

3. `iload\_2` b

4. `ifne Label1`

5. `iload\_3`c

6. `ifne Label2` and

7. `Label1:`

8. \_\_\_\_\_\_\_

9. `Label2:`

…

The code at line 8 should be:

- A) `**iconst\_1**`

- B) `iconst\_0`

- C) `istore\_1`

- D) `istore\_2`

**Question 4.**In a given function:

```java

void foo(int a, int b) {

a = b + 1;

b = a - 2;

}

```

If `a` and `b` are stored in local variable indices 0 and 1 respectively, and assuming short-circuit evaluation, what is the correct bytecode instruction to store the result of `b = a - 2`?

- A) `istore\_0`

**- B) `istore\_1` store b**

- C) `iload\_2`

- D) `iload\_1`

**Question 5.** The effect of short-circuiting is most evident at line \_\_\_ of the provided bytecode sequence.

**- A) 2** and/or can short-circuiting, if 0 with and, skip; if 1 with or, skip

- B) 4

- C) 6

- D) 8

**Question 6.** Which of the following statements about lazy evaluation is true?

**- A) It avoids unnecessary calculations** (in and, or)

- B) It is always less efficient than eager evaluation

- C) It is mostly used to maximize CPU usage

- D) It is not used in functional programming models

Use the following code to answer questions 7 to 10:

```python

def example\_function(a, b, c):

a = a + b

b = b + c

c = a + c

print(a, b, c)

x = 2

y = 3

z = 4

example\_function(x, y, z)

print(x, y, z)

```

**Question 7.** If all parameters are passed by value, the printed values are:

**- A) `5 7 9` and `2 3 4`**

- B) `5 7 9` and `5 7 9`

- C) `2 3 4` and `5 7 9`

- D) `2 3 4` and `2 3 4`

a <- 2

b <- 3

c ~ z

a <- 5

b <- 7

c~z <- 9

⇒ print: 5 7 9

x <- 5

y <- 7

⇒ print: 5 7 9

**Question 8.** If parameters `a` and `b` are passed by value-result and `c` is passed by reference, the printed values are:

**- A) `5 7 9` and `5 7 9`**

- B) `5 7 11` and `5 7 11`

- C) `5 7 11` and `2 3 11`

- D) `2 3 9` and `2 3 4`

a ~ x

b <- 3

c ~ z

a~x <- 5

b <- 7

c

**Question 9.** If `a` is passed by reference, `b` by value, and `c` by pointer, the printed values are:

**- A) `5 7 7` and `5 3 7`**

- B) `5 7 11` and `5 3 11`

- C) `7 7 7` and `2 3 7`

- D) `7 7 11` and `2 3 11`

Pass by pointer: Maybe

b = b + c, since c is pointer, c overshadows b and write 7 to b

Then c = c + a, since c is pointer, c overshadows a and write 7 to c

Then in function it prints '5 7 7' but since b is passed by value, outside function it prints '5 3 7'

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | x | y | z | a | b | c |
|  | 2 | 3 | 4 |  |  |  |
|  | 2 | 3 | 4 | 2 | 3 | 4 |
|  | **5** |  |  | 5 |  |  |
|  |  |  |  |  | **7** |  |
|  |  |  | **7** |  |  | **7** |
|  | 5 | 3 | 7 | 5 | 7 | 7 |

=> example\_function(x, y, z) = ‘5 7 7’

print(x, y, z) = ‘5 3 7’

**Question 10.** If all parameters are passed by name, the printed values are:

- A) `5 7 11` and `5 7 11`

**- B) `5 7 9` and `2 3 4`**

- C) `7 10 14` and `7 10 14`

- D) `7 10 14` and `2 3 4`

```python

def example\_function(a, b, c):

    a = a + b

    b = b + c

    c = a + c

    print(a, b, c)

x = 2

y = 3

z = 4

example\_function(x, y, z)

print(x, y, z)

```

def example\_function(x, y, z):

    x = x + y

    y = y + z

    z = x + z

    print(x, y, z)

x <- 5

y <- 7

z <- 9

⇒ print: 5 7 9

⇒ print: 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Value | Value-result | Reference | Name |
| Mechanism | Copy value of outside variables to the inside variables | Copy value of outside variables to the inside variables  At the end of the function, it will copy back values of inside variables to outside variables | inside variable is an alias of outside ones | All inside variables can be replaced by the names of outside variables.  It does not affect the values of outside variables if the outside variables are not in global scope. |

i = 3

arr = [1, 2, 3, 4, 5]

def foo(a, b):

for i in range(0…a):

b += 1

foo(5, arr[i])

\*\* arr are passed by name.

—--------------

def foo(5, arr[i]):

for i in range(0…5):

arr[i] += 1

// arr = [2, 3, 4, 5, 6]

—---------------------------------------------

int a[5];

int i, X=1;

void swap (int&a, int&b,int&c)

{ int t;

c++;

t = a+X+c;

a = b;

b = t;

}

void main()

{

for (i=0;i<5;i++)

a[i] = 6 – i;

i = 2;

swap(i, a[i],X);

printf(“%d%d%d%d%d%d”,i,a[0], a[1], a[2], a[3], a[4]);

}

X = 1

a = [6, 5, 4, 3, 2]

i = 2

{

X++; // X = 2

t = i+X+X; // t = 6

i = a[i]; // i = a[2] = 4

a[i] = t; // a[4] = 6

}

⇒ print: 4 6 5 4 3 6