Which Python statement calls the function printf in a shared library loaded as mysharedlib?

Question 1 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1) | cdll.printf("print number %d",123) | |
|  | |  |  | | --- | --- | | **2)** | **mysharedlib.printf(c\_string("print number %d"),c\_int(123))** | |
|  | |  |  | | --- | --- | | 3) | mysharedlib.printf("print number %d",123) | |
|  | |  |  | | --- | --- | | 4) | printf("print number %d",123) | |

## Question 2 (1 point)

The stdcall calling convention:

Question 2 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1) | i**s mainly used in older, 16-bit environmen**ts | |
|  | |  |  | | --- | --- | | 2) | pushes the parameters on the stack from left to right | |
|  | |  |  | | --- | --- | | **3)** | **is used only if cdecl is not** available | |
|  | |  |  | | --- | --- | | 4) | does not allow variable number of parameters | |

## Question 3 (1 point)

The DLL libraries that use the cdecl calling convention are:

Question 3 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1) | windll | |
|  | |  |  | | --- | --- | | **2)** | **cdll** | |
|  | |  |  | | --- | --- | | 3) | oledll | |
|  | |  |  | | --- | --- | | 4) | None of these | |

## Question 4 (1 point)

What typically happens when a program runs out of in-memory stack space?

Question 4 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **1)** | **The program aborts.** | |
|  | |  |  | | --- | --- | | 2) | The program requests more stack space from the operating system. | |
|  | |  |  | | --- | --- | | 3) | The operating system allocates another page of memory and places it at a virtual address just below. | |
|  | |  |  | | --- | --- | | 4) | The current stack is swapped out to disk and a new stack is allocated. | |

## Question 5 (1 point)

The ctypes module is used to:

Question 5 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1) | serialize C variables | |
|  | |  |  | | --- | --- | | 2) | convert the types of Python variables | |
|  | |  |  | | --- | --- | | 3) | serialize Python variables | |
|  | |  |  | | --- | --- | | 4) | **allow calling functions in DLL libraries** | |

## Question 6 (1 point)

What happens with a stack implementation using the PUSH and POP instructions on the Intel x86 platform?

Question 6 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1) | On POP, it decrements the stack pointer, and then loads the data. | |
|  | |  |  | | --- | --- | | 2) | On PUSH, it decrements the stack pointer, and then saves the data on the stack. | |
|  | |  |  | | --- | --- | | 3) | **On PUSH, it saves the data on the stack, and then decrements the stack pointer.** | |
|  | |  |  | | --- | --- | | 4) | On POP, it increments the stack pointer, and then loads the data. | |

## Question 7 (1 point)

The cdecl calling convention:

Question 7 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1) | pushes the parameters on the stack from left to right | |
|  | |  |  | | --- | --- | | 2) | does not allow variable number of parameters | |
|  | |  |  | | --- | --- | | 3) | **is used only if stdcall is not available** | |
|  | |  |  | | --- | --- | | 4) | requires that all parameters be passed on stack, and the return value, if any, be in EAX | |

## Question 8 (1 point)

Typically, every program receives allocated memory for at least the:

Question 8 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **1)** | **code, data, stack** | |
|  | |  |  | | --- | --- | | 2) | code, stack | |
|  | |  |  | | --- | --- | | 3) | code, data | |
|  | |  |  | | --- | --- | | 4) | code | |

## Question 9 (1 point)

Installing a software breakpoint means:

Question 9 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1) | **the debugger executes instructions one by one and stops at the breakpoint address** | |
|  | |  |  | | --- | --- | | 2) | replacing the instruction at the breakpoint address with an INT 3 instruction | |
|  | |  |  | | --- | --- | | 3) | telling the operating system you want to stop the program at a certain address | |
|  | |  |  | | --- | --- | | 4) | finding all occurrences of the breakpoint address in the program and passing it to the debugger | |

## Question 10 (1 point)

The DLL libraries that use the stdcall calling convention are:

Question 10 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **1)** | **windll and oledll** | |
|  | |  |  | | --- | --- | | 2) | cdll | |
|  | |  |  | | --- | --- | | 3) | oledll | |
|  | |  |  | | --- | --- | | 4) | windll | |