QUESTION 1  
  
Meets Expectation  
  
The student correctly identifies that the HAL stands for Hardware Abstraction Layer and explains that it provides prewritten functions that abstract hardware operations. The student also provides an example of how the HAL simplifies interaction with the hardware by allowing programmers to use high-level functions instead of writing low-level functions for individual bit operations.  
  
Great job! You have correctly identified that the HAL is the Hardware Abstraction Layer and explained its purpose in simplifying interaction with the hardware. Your example of using high-level functions provided by the HAL instead of writing low-level functions for individual bit operations is a good illustration of how the HAL can make programming more efficient. To further improve your answer, you could provide some additional details on why it is useful to have a HAL, such as how it promotes code reusability and portability across different hardware platforms. Keep up the good work!  
  
QUESTION 2  
  
Does not meet expectation.  
  
The student work contains some errors and omissions. While the student correctly identifies that the \_\_io\_putchar function takes in a character as a parameter and outputs it to a specified IO channel, they do not mention that it enables printing to the UART or console. Additionally, the student does not mention that the function is used by the printf function or that it is required because different projects may require different hardware for printing.  
  
To improve, make sure to include all the necessary information in your explanation.  
  
QUESTION 3  
  
Meets Expectation  
  
The student correctly identifies that the debugger was required in order to examine the values stored in the registers. They also mention that using printf to print out the contents of the registers is possible, but it does not allow for creating breakpoints in the code. This is a good explanation of why the debugger was necessary in this lab.  
  
Well done! Just one suggestion for improvement: it would be helpful to provide a specific example of how examining the values in the registers using the debugger can be useful in debugging code. This will make your explanation more concrete and easier to understand. Keep up the good work!  
  
QUESTION 4  
  
Does not meet expectation.  
  
The student work contains errors. While they correctly identify that the new value for PSP is computed by offsetting the value of MSP\_init\_val, they incorrectly state that the offset is 0x400. The offset should actually be the size of the main stack, which is 0x400 bytes. Additionally, the student does not address why knowing the initial location of MSP is necessary for this to work.  
  
QUESTION 5  
  
Meets Expectation  
  
The student correctly identifies that to find multiple new stacks, we would need to use the same basic method of offsetting the address from a known, valid stack location. They also correctly identify that we would need to offset each new stack address in some way that ensures that the new address remains a valid stack address.  
  
Well done! You have correctly identified that each stack is the same size and occupies consecutive blocks in memory. You also correctly mention that the start of each stack is a constant number of bytes away from the start of the last stack. Additionally, you mention that if the MSP initial value is known, we can find the starting address of the nth thread stack by subtracting n \* (the size of each stack) from the MSP initial value.  
  
To improve your answer, you could provide more details on how to ensure that the new address remains a valid stack address. Consider discussing the importance of aligning the stack addresses to ensure proper stack operation.