QUESTION 1  
  
Does not Meet Expectation.  
  
The student work does not fully meet the expectation because it does not provide a detailed explanation of why the HAL is used in a general embedded project. While the student mentions that the HAL abstracts hardware operations and provides pre-written functions, they do not fully explain the benefits of using the HAL in a project.  
  
Here's some actionable feedback for improvement:  
  
\* Provide more details on the benefits of using the HAL in a project, such as reducing the complexity of code, improving code reusability, and reducing the risk of errors due to incorrect hardware interactions.  
\* Explain how the HAL simplifies interaction with hardware, and provide examples of how it can be used to simplify code development.  
\* Consider including a brief overview of the hardware operations that the HAL abstracts, and how it provides a higher-level interface for software development.  
  
Hints:  
  
\* Think about how the HAL simplifies the process of developing software for embedded systems.  
\* Consider how the HAL can help reduce the learning curve for developers who are new to a particular microcontroller or hardware platform.  
\* Research and include some examples of common tasks that the HAL can simplify, such as input/output operations, memory management, and interrupt handling.</s>  
  
QUESTION 2  
  
Does not Meet Expectation.  
  
The student work partially identifies the function of \_\_io\_putchar but lacks a clear understanding of its purpose and requirement. The work mentions that the function takes a character as a parameter and outputs it to a specified IO channel, but it does not explain why this function is necessary. Additionally, the work mentions that the function uses the HAL\_UART\_Transmit function to send the character to the huart2 output channel, but it does not explain why this is necessary.  
  
Actionable feedback:  
  
\* To meet expectation, the student should research and explain the purpose and requirement of the \_\_io\_putchar function.  
\* The student should provide a clear and concise explanation of why the function is necessary, and how it enables printing to the UART or console.  
\* The student should also explain the role of the HAL\_UART\_Transmit function in the context of \_\_io\_putchar.  
  
Hint: The student can refer to the documentation of the Cortex M4 microcontroller and the Keil µVision IDE to understand the role of the \_\_io\_putchar function and how it interacts with other hardware and software components.</s>  
  
QUESTION 3  
  
Does not Meet Expectation  
  
The student work partially addresses the question, but there are some errors and omissions.  
  
The student correctly mentions that the debugger allows them to see the contents of all the registers, and that they cannot create breakpoints for the code using printf. However, the explanation of why the debugger is necessary is not complete. The student should have explained that the debugger allows them to examine the values stored in the registers at a specific point in the code, such as before a line is run, whereas printf only allows them to print out the values of variables at a particular point in time. Additionally, the student should have mentioned that the debugger allows them to set breakpoints and step through the code line by line, which is essential for debugging purposes.  
  
Actionable feedback:  
  
\* Provide a clear explanation of why the debugger is necessary for systems programming on the Cortex M4 microcontroller.  
\* Explain the difference between using the debugger and using printf to instrument code.  
\* Give examples of scenarios where the debugger would be particularly useful, such as identifying register values before a line is run, or stepping through code line by line to identify a bug.  
  
Hints:  
  
\* Think about the limitations of using printf to instrument code.  
\* Consider the benefits of being able to examine the values stored in registers at specific points in the code.  
\* Research how the debugger works and how it can be used to debug code.</s>  
  
QUESTION 4  
  
Based on the given rubric, I would classify the student work as "meets expectation".  
  
The student work correctly identifies that the new value for PSP is computed by offsetting the value of MSP\_init\_val. The student work also correctly explains that if they did not know the initial value for MSP, they could not be sure that their value for PSP would be a valid stack address.  
  
The only suggestion I would give for improvement is to provide more detail in their explanation. For example, they could explain what the MSP and PSP registers are used for, and why they need to be initialized with valid values. Additionally, they could provide more context on why the main stack's size was chosen to be 0x400 bytes, and why they chose to offset the MSP initial value by 0x400 to get the PSP value.  
  
Here's an example of how I would provide feedback to the student:  
  
"Great job on correctly identifying how the stack allocation method works! Your explanation is clear and concise. To take it to the next level, consider adding more detail to your explanation. For example, you could explain what the MSP and PSP registers are used for, and why they need to be initialized with valid values. Additionally, you could provide more context on why the main stack's size was chosen to be 0x400 bytes, and why you chose to offset the MSP initial value by 0x400 to get the PSP value. Keep up the good work!"</s>  
  
QUESTION 5  
  
Does not Meet Expectation.  
  
The student work does not fully address the problem. While it is correct that the start of each stack is a constant number of bytes away from the start of the last stack, the student work does not provide a clear method for calculating the starting address of each new stack. The suggestion to subtract n \\* (the size of each stack) from the MSP initial value is not a valid solution, as it does not take into account the possibility of overflow or underflow, and it does not ensure that the resulting address is a valid stack address.  
  
To meet expectation, the student work should provide a clear and correct method for calculating the starting address of each new stack, taking into account the size of each stack and the number of threads. The method should also ensure that the resulting address is a valid stack address, and that there is no overflow or underflow.  
  
Here's a hint for improvement: Consider using a formula that combines the size of each stack and the number of threads to calculate the starting address of each new stack. Make sure to check for overflow or underflow, and ensure that the resulting address is a valid stack address.</s>