Problem 1: Hexadecimal Addition

Approach:
This given C program involves handling overflows in hexadecimal addition for two numbers. The steps involved in solving this problem are as follows:

Read Input:
The program reads two hexadecimal numbers (num1 and num2) from standard input (scanf) to EOF.
Perform Addition:

The variable'result' stores the sum of num1 plus num2.
Calculate hexadecimal digits:

The function hexDigits calculates hexadecimal digits in a specified number. This information can then be utilized in order to structure an output.
Check for overflow:

This allows the program to determine whether or not there is an overflow condition, as determined by comparing the sum of num1 and num2 to ULLONG_MAX.

Print-formatted result:
Then, the program prints the input numbers, a line of dashes for separation, and finally the outcome. It also prints another line saying "overflow."
Improvements Made:

This program stores and retrieves numbers in a hexadecimal format using %llX as its format specifier.
Appropriate overflow conditions were checked and handled.
It is also commented on for clarity.
After processing the input, the program exits successfully.
Problem 2: Hexadecimal Multiplication

Approach:

B. The second C Program deals with the hexadecimal multiplication of two numbers. The steps involved in solving this problem are as follows:

Read Input:
Likewise, the scanf() function is employed to scan two hexadecimals from standard input up to EOF.
Perform Multiplication:

Finally, the last step involves multiplying and storing the product of num1 and num2 into a variable, which is labelled as result.
Calculate Hexadecimal Digits:

It reuses the hexDigits function to determine the number of hexadecimal digits in the product.
Print-formatted result:

The output is displayed, showing the input numbers followed by the gap, which represents separation, and then the result in two bases.
Improvements Made:

For reading and printing hexadecimal numbers, the program incorporates the format specifier %llX.
The code has clear comment.
On execution of the input, it is successful and terminates successfully.
General Improvements:

File Handling:

Some commented-out parts handle files in the code. These sections, depending on a requirement, allow for reading of input from files but are uncommented initially, making them readable.
Consistency:

This guarantees uniformity in the use of format specifiers for printing, which also ensures readability.

Error Handling:

A second program that involves disabled error handling code for file opening is commented out. This code can be ‘uncommented’ in case there is a requirement for file handling.

Conclusion:

The two programs are good at resolving the identified issues whereby a hexadecimal addition is performed, and another program implements multiplication. The code has also been modularized, commented on well, and provided with checks against an overflow. File handling sections can be switched on according to a particular function, and the program will take information from the files.