Task#1

Write ASM instructions that calculate EAX * 25 using binary multiplication.

Task#2

Give an assembly language program to move -128 in BX and expand EBX. Using shift and rotate instruction.

Task#3

The time stamp field of a file directory entry uses bits 0 through 4 for the seconds, bits 5 through 10 for the minutes, and bits 11 through 15 for the hours. Write instructions that extract the minutes and copy the value to a byte variable named **bMinutes**.

Task#4

Write a series of instructions that shift the lowest bit of AX into the highest bit of BX without using the SHRD instruction. Next, perform the same operation using SHRD.

Task#5

Implement the following C++ expression in assembly language, using 32-bit **signed** operands:

val1 = (val2 / val3) * (val1 / val2);

Task#6

Create a procedure **Extended_Add** procedure to add two 64-bit (8-byte) integers.

Task#7

Write a procedure named **IsPrime** that sets the Zero flag if the 32-bit integer passed in the EAX register is prime. Optimize the program's loop to run as efficiently as possible. Write a test program that prompts the user for an integer, calls **IsPrime**, and displays a message indicating whether the value is prime. Continue prompting the user for integers and calling **IsPrime** until the user enters 1.

Task#8

Write a program that performs simple encryption by rotating each plaintext byte a varying number of positions in different directions. For example, in the following array that represents the encryption key, a negative value indicates a rotation to the left and a positive value indicates a rotation to the right. The integer in each position indicates the magnitude of the rotation:

key BYTE 2, 4, 1, 0, 3, 5, 2, 4, 4, 6

Your program should loop through a plaintext message and align the key to the first 10 bytes of the message. Rotate each plaintext byte by the amount indicated by its matching key array value. Then align the key to the next 10 bytes of the message and repeat the process.