1.

Let's first analyze the MARIE architecture that has 16-bit instructions. 4 bits are allocated for the opcode, allowing for $2^4 = 16$ possible operations, and 12 bits (16-4) for the address.

(A) If the instruction set has 150 different operations, then we need to find 2^x value closest to 150. Therefore, x = 8. Thus, there are needed 8 bits for the opcode.
(B) 24bits-8bits = 16bits. Hence, there are left 16 bits for the address part.
(C) 2^{16}
(D) $2^{24}-1$ = 16,777,215. The largest unsigned binary number is 16,777,215.
(E) $-2^{(24-1)}$ = -8,388,608 . The smallest signed binary number is -8,388,608 .

2.

The input validation would be handled the same way as the multiplication program.

Check for zeros
Check if x and y are equal to 0. If yes, exit the program. If one of the input, x or y is equal to 0 it prints the value x, y and z (is equal to 0).
Check the sign of x and y
If x or y is negative, it makes converts it into positive by subtracting x or y by the same value twice this way producing its absolute value.
Check for overflow/underflow
When checking the sign, the program also checks for overflow and underflow conditions. If any of the inputs, x or y is negative and it' consequently converted to positive. The overflow/underflow condition is checked by comparing if the result of the subtraction, which should be result in a positive value, is less than 0 (or still negative). If that the case, then there was an overflow or underflow, meaning that the range of the input type (decimal in this program) was surpassed.
The overflow/underflow condition is also checked in the multiplication procedures.

Then, I would compare the absolute value of the inputs to see which one is larger. The greatest value (gValue) would be loaded for the shift operation. In order to find out how much to shift first we need to check how many digits the smallest value (svalue) has. Therefore, we compare to 10. Note that we are doing gValue x sValue.

The algorithm we are using is subtract 2 to sValue until it is less or equal to 0. If it is equal to 0, then we shift the number of times sValue was subtracted (counter). If it's less than zero, then we shift the number of times sValue was subtracted (counter) and we add to the result gValue for how many times the last value before the counter becomes negative.

gValue x sValue = load value | shift counter | add to gVlaue x last value

If it is smaller it's smaller then 10, or 1 digit then we will have hardcoded shift conditions

If sValue = 2	If sValue = 3	If sValue = 4	If sValue = 5
Load gValue	Load gValue	Load gValue	Load gValue
Shift one	Shift one	Shift two	Shift two
	Add gValue x 1		Add gValue x 1

So forth, until 9.

When, sValue = 9

Load gValue

Shift three

Add gValue x 1

Now, when the sValue is more than 1 digit than we divide it by 8 subtracting said number to nine AC is less or equal to zero. We do this procedure N times. N being the result of the division of sValue to 8, always increment to the result. Finally, we add to the result to gValue x lasValue.

25 x 23 = 575

25 = 000000011001

23 – 8 = 15

15 – 8 = 7

7 – 8 = -1

STOP N = 2 lastValue = 7

Loop (<=N)

{

Load gValue

Shift three

}

Add gValue (25 x 7)

Remember we have harcoded the result for multiplication for sValue from 2-9.

0000000110010000 = 200

200 x N = 200 x 2 = 400

25 x 7 = 175

Result = 400 + 175 = 575.

The current multiply does not compare the input to find the greastest. And it multiplies by adding the absolute values of x, y times using a loop. By using shift we use fewer instructions then our previous program. The reason we use 8 is so that we won't shift too above the program limitations since the address has 12bits.

3.

Overflow is a condition where a register is not large enough to contain the result of an arithmetic operation. In signed arithmetic operations, overflow is detectable when the carry in to the sign bit does not equal the carry out from the sign bit. For instance, when a value is incremented and the result is beyond the scope of the range. There is a status or flag register that holds information indicating various conditions, such as an overflow in the ALU.

4.

(A) Add 10 consecutives numbers from x to x+10

(B) It doesn't handle overflow/underflow.