Drag-and-drop the correct sequence number of the assembly code to form a program where 7 numbers in an array are added and stored in `eax` register. Note that your sequence must absolutely match the line numbers to the left-most column of the table. The answers for Lines 2, 3 and 5 have been provided. Complete the rest.

Line 1	`mov esi, array` ✗ *#lea esi, array#*
Line 2	`mov eax, 0`
Line 3	`mov ecx, 0`
Line 4	`sumLoop: add eax, [esi]` ✓
Line 5	`add esi, 4`
Line 6	`inc ecx` ✓
Line 7	`cmp ecx, 7` ✓
Line 8	`jl sumLoop` ✓

jnl sumLoop | inc ecx | dec ecx | lea esi, array | jl sumLoop | sumLoop: add eax, [esi] | cmp ecx, 7 | mov esi, array

Your answer is partially correct.

Drag-and-drop the correct arguments and/or instructions to the missing places in the following program segment to push a string onto a stack.

```
char ✓  myArray[MAX_SZ] = "Hello XJTLU";

_asm ✓  {
    mov ecx, MAX_SZ-1 ✓
    mov esi, 0 ✓

    cycleIt ✓ :
        movzx ✓  eax, myArray[ esi ✓ ]
        push ✓  eax
        inc ✓  esi
        loop ✓  cycleIt
}
```

loop | asm | push | cycleIt | esi | movzx | MAX_SZ | [esi] | inc | char | mov | dec | _asm | 0 | MAX_SZ-1 | jmp | pop | ecx

Drag-and-drop the correct sequence number of the assembly code to form a program that sorts an array of 7 integers in an **ascending** order (Bubble Sort). Note that your sequence must absolutely match the line numbers to the left-most column of the table. Complete Lines 4, 7-10.

Line 1	`lea esi, array`
Line 2	`mov ecx, 7`
Line 3	`outerLoop: mov edx,ecx`
Line 4	`innerLoop: cmp edx, ecx` ✓
Line 5	`jz noExchange`
Line 6	`mov eax, [esi + ecx * 4 - 4]`
Line 7	`mov ebx, [esi + edx * 4]` ✗ *#mov ebx, [esi + edx * 4 - 4]#*
Line 8	`cmp ebx, eax` ✓
Line 9	`jnl noExchange` ✗ *#jl noExchange#*
Line 10	`mov [esi + ecx * 4], ebx` ✗ *#mov [esi + ecx * 4 - 4], ebx#*
Line 11	`mov [esi + edx * 4 - 4], eax`
Line 12	`noExchange: dec edx`
Line 13	`jnz innerLoop`
Line 14	`loop outerLoop`

mov [esi + ecx * 4 - 4], ebx | cmp ebx, eax | jnl noExchange | jl noExchange | innerLoop: cmp edx, ecx | mov ebx, [esi + edx * 4 - 4] | mov ebx, [esi + edx * 4] | mov [esi + ecx * 4], ebx

Your answer is partially correct.

Drag-and-drop the correct arguments and/or instructions to the missing places for the following program that sums all the numbers in an array.

```
int arraySize = 5;
int intArray[ arraySize ✔ ] = {12, 3, 7, 23, 9};
int totalAmt = 0 ✔ ;

_asm{
     lea ✔ edi, intArray ✔
     mov ecx, arraySize ✔
     mov eax, totalAmt

     addTotalAmt ✔ :
          add ✔ eax, [edi]
          add edi, 4 ✔
          loop ✔ addTotalAmt

     mov totalAmt ✔ , eax
}
```

movzx | intArray | loop | arraySize-1 | totalAmt | arraySize | add | addTotalAmt | 4 | [intArray] | lea | 0 | jmp | 8