

## LAB # 10 TASKS

### **Task # 01:**

A graphics engine performs real-time rendering of sprites, where calculating each sprite's pixel area is executed tens of thousands of times per frame. You are required to implement a procedure `CalcSpriteArea` that computes the area of a sprite, receives the width and height as parameters, validates the input (returning 0 if either value is zero or negative), and returns the result in EAX.

In the main program, the procedure will be tested with multiple inputs, and it must return execution while ensuring the correct result is produced. The procedure must follow standard stack usage conventions, preserve any necessary registers, and be suitable for repeated high-frequency calls in the rendering loop.

### **Task # 02:**

A server application maintains two configuration values representing its primary communication port and its failover backup port. These values must occasionally exchange roles during runtime. You are required to implement a procedure `SwapPorts` that swaps the contents of the two port variables directly in memory, based on the addresses provided to it by the calling program.

In the main program, the addresses of the two port variables are passed to the procedure, and the procedure must return execution while ensuring the values have been successfully exchanged in memory. The procedure must also manage its own stack usage and clean up the passed parameters upon return.

### **Task # 03:**

A financial software application processes lists of transactions and requires a reliable way to compute their total. You are required to implement a procedure `SumTransactionList` that receives a pointer to an array of DWORD transactions and the count of elements, computes the sum, and returns the result in EAX.

In the main program, the procedure will be called using `INVOKE` with the array transactions DWORD 150, 200, 75, 400, and it must return execution while ensuring the correct sum, 825, is produced. The procedure must use MASM's high-level directives, declare any necessary local variables, and follow proper stack usage conventions to ensure maintainability and safety for repeated use.