

**University of Dhaka**  
**CSE 3113: Microprocessor and Assembly Language Lab**  
**Lab Exam**  
**Date: 01.07.2025**

1. Write an assembly program that calls a function **string\_length** to find the length of a string and stores the result in memory.
  
2. Write an assembly program that calls a function **remove\_first** that takes an array and its length and removes the first element of the array.

**Input:**

R0 = array base address, R1 = length

3. Write an assembly program that calls a function **remove\_last** that takes an array and its length and removes the last element of a list.

**Inputs:**

R0 = array base address, R1 = length

4. Write an assembly program that calls a function **sum\_avg** that takes an array and its length, returns the **sum** of all elements in R3 and the **average** in R4.

**Input:**

R0 = array base address, R1 = length

5. Write an assembly program that calls a function **find\_min\_max** that takes an array and its length and returns minimum in R3, maximum in R4

**Input:**

R0 = array base address, R1 = length

6. Write an assembly program that calls a function **array\_ascend** that takes an array of **n** numbers and arranges the array in ascending order.

7. Write an assembly program that calls a function, **reverse\_array**, which takes an array and its length, and reverses a 32-bit integer array in memory.  $\text{arr[]} = \{1, 2, 3, 4\} \rightarrow \text{arr[]} = \{4, 3, 2, 1\}$

**Input:**

R0 = array base address, R1 = length

8. Write an assembly program that calls a function **swap\_array** that takes an array and its length and swaps elements using a two-pointer technique: start and end.

**Input:**

R0 = array base address, R1 = length

9. Write an assembly program that calls a function **odd\_even** that takes an array and its length, and counts how many even and odd numbers exist in a given array.

**Output:**

R0 = number of even numbers, R1 = number of odd numbers

10. Write an assembly program that calls a function **merge\_array** that takes two sorted arrays and their respective lengths and merges them into a third array (sorted).

**Input:**

R0 = base A, R1 = len A, R2 = base B, R3 = len B, R4 = base of result array

**Output:**

R0 = total elements in result

11. Write an assembly program that calls a function **rotate\_k** that takes an array, its length and k (rotation offset) and rotates the array left by k positions.

**Input:**

R0 = base, R1 = length, R2 = K (rotation offset)

**Example:**

$\text{arr} = [1, 2, 3, 4, 5], k = 2 \rightarrow [3, 4, 5, 1, 2]$