

### Problem Statement 1: (5 marks) (time limit per test 3 seconds)

You are required to simulate the behavior of a virtual memory management system that divides the address space into fixed-size pages. You will implement the following TLB replacement algorithms:

- **FIFO (First-In-First-Out)**
- **LIFO (Last-In-First-Out)**
- **LRU (Least Recently Used)**
- **Optimal Algorithm**

You will be provided with a predefined size for the address space, page size, and TLB size. Your task is to develop a simulation that demonstrates how different algorithms behave.

**Instructions: This part has to be done in C++**

#### 1. TLB Replacement Algorithms:

- Implement each of the following TLB replacement algorithms:
  - **FIFO**: The oldest entry in the TLB is replaced when the TLB is full.
  - **LIFO**: The most recent entry in the TLB is replaced.
  - **LRU**: The least recently used entry in the TLB is replaced.
  - **Optimal**: Replace the entry that will not be used for the longest time in the future.

#### Deliverables:

- Source code of the simulation, properly commented.
- You will have to print out the number of TLB Hits in the format given below.

#### Input

- The first line contains a single integer **T** ( $1 \leq T \leq 100$ ), representing the number of test cases.
- For each test case, the input is structured as follows:
  1. **Address Space Size**:
    - A single integer **S** ( $1 \leq S \leq 4096$ ), denoting the size of the address space in megabytes (MB).
  2. **Page Size**:
    - A single integer **P** ( $1 \leq P \leq S$ ), representing the size of each page in kilobytes (KB).
  3. **TLB Size**:
    - A single integer **K** ( $1 \leq K \leq 1024$ ), indicating the number of entries in the Translation Lookaside Buffer (TLB).
  4. **Number of Memory Accesses**:

- A single integer **N** ( $1 \leq N \leq 10^6$ ), specifying the total number of memory address accesses.
5. **Memory Addresses:**
- **N** subsequent lines, each containing a memory address in hexadecimal format. Each address is a valid 32-bit hexadecimal number without any prefix (e.g., **1A2B3C4D**).

## Output

For each test case, output a single line containing four space-separated integers representing the number of TLB hits for each replacement algorithm in the following order:

1. **FIFO (First-In-First-Out) TLB Hits**
2. **LIFO (Last-In-First-Out) TLB Hits**
3. **LRU (Least Recently Used) TLB Hits**
4. **OPT (Optimal) TLB Hits**

## Problem Statement 2: (5 marks)

- Your functions should have the same calling conventions as the original *malloc*, *calloc*, and *realloc*.
- The name of your functions should be *my\_malloc*, *my\_calloc*, *my\_free*. The function signature should be the same as the original library functions.
- Complete the functions provided in template code. Make sure the file is named `<EntryNo>mmu.h` (2020MT60986mmu.h) during the submission.
- You are free to use any data structure (you will have to implement them on your own) for managing the free memory.
- You are required to use the system calls *mmap*, *brk*, and *sbrk*. Please refer to the man pages to learn about them in detail.
- We expect you to submit a working library, which, after linking to a program, should work exactly like the original *malloc*, *free* and *calloc*.

## Submission:

Submit a zip file containing a single cpp source code for Problem Statement 1 and the mmu.h file for Problem Statement 2