Assignment - 09
CS341: Operating System Lab

1. You are tasked with developing an advanced memory management tool for an operating system that efficiently handles page requests from processes. Given a non-sorted array of integers representing a sequence of page numbers requested by a process (with page numbers ranging from 0 to n, where n is the maximum page number requested), your goal is to identify the first missing page number in the sequence. The challenge is to optimize your solution to run in linear time O(n) and require constant space O(1), without using any additional data structures. Your implementation should robustly handle cases where page requests may not follow a consecutive order and should be able to gracefully manage large input sizes. For example, if the input sequence is [3, 0, 1, 2, 5, 6, 4], your algorithm should efficiently determine that the first missing page is 7. Ensure that your solution can be easily integrated into the existing memory management system, providing both the missing page number and an explanation of the algorithm used to achieve the result.

2. In modern operating systems, efficient memory management is crucial for ensuring optimal performance and resource utilization. One key aspect of memory management is the implementation of page replacement strategies, which dictate how the system handles page faults when the allocated physical memory is full. In this problem, you are tasked with evaluating three common page replacement algorithms: First-In-First-Out (FIFO), Least Recently Used (LRU), and Optimal Page Replacement. You will analyze their performance by taking a sequence of page reference numbers and a fixed number of frames as input. The goal is to calculate the number of page faults incurred by each algorithm during the execution of the reference string. FIFO operates on a simple principle of replacing the oldest page in memory, whereas LRU tracks the usage of pages over time to replace the least recently accessed page, thus providing a more efficient approach in many scenarios. The Optimal algorithm, while theoretical, replaces the page that will not be used for the longest period in the future, serving as a benchmark for evaluating the effectiveness of the other strategies. You will implement these algorithms in a C program, prompting the user for the number of pages, the reference string, and the number of frames. After processing the input, the program will output the total page faults for each algorithm, allowing for a comparative analysis of their efficiencies in managing memory under different workloads.

3. In a variable partitioned memory management system, efficient memory utilization is essential, yet external fragmentation often poses a significant challenge. This problem requires you to analyze the memory fragmentation resulting from dynamic memory allocation and deallocation processes. You will be given a set of allocated memory blocks, each with a specific size, reflecting how memory is currently utilized. Your task is to calculate the total external fragmentation after a series of deallocations. External fragmentation occurs when free memory is scattered in small, non-contiguous blocks, making it impossible to allocate large memory requests even if the total free memory is sufficient. To solve this problem, you will write a C program that takes user input for the sizes of the allocated memory blocks and the sizes of the blocks to be deallocated. After processing the deallocations, the program will compute the total amount of external fragmentation in the system. This analysis will help illustrate the impact of memory allocation strategies and the importance of effective memory management in reducing fragmentation and improving overall system performance.

4. Memory leaks pose a significant challenge in software development, particularly within the context of operating systems, where efficient memory management is crucial for maintaining overall system performance. In this assignment, you are tasked with investigating the impact of memory leaks by implementing a C program that allows for dynamic memory allocation based on user input. The program should prompt the user to specify the number of memory allocations and the size of each allocation. For each allocation, the program will allocate memory without freeing it, simulating a real-world scenario where memory leaks occur. As the program runs, it will demonstrate the gradual consumption of memory resources due to unfreed allocations. At the conclusion of the program, you will analyze the implications of these memory leaks on system performance, including increased memory usage and potential system instability. Additionally, you will discuss techniques for detecting and mitigating memory leaks, emphasizing best practices in memory management to ensure efficient resource utilization in an operating system environment. Through this exercise, you will gain insight into the importance of proper memory management and the consequences of neglecting it in software applications.ions and ensure orderly task execution between the processes.