



**Batch: B-2 Roll No: 16010422234 Name: Chandana Galgali Date: 04/10/2024**

**Experiment No: 7**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim:** To implement the Least Recently Used (LRU) page replacement algorithm.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources needed:** Text editor and JAVA/C compiler.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Theory:**

**Pre lab/Prior concepts:**

Algorithm:

1. Accept the number of frames from the user in n.
2. Accept the size of the reference string in m.
3. Read reference string into array s.
4. Take a stack of maximum size=n.
5. Read one page number from s and enter it in the stack. Increment stack pointer.
6. Read the second page number from s. compares it with the page number existing in the stack. If matches then hit++ and goto step 7 else goto step 8.
7. Remove that page from the stack. Shift all the contents of the stack down by one. And put the current page on stack top. (least recently used page will be at the bottom of stack)
8. If the stack is empty (frame is free) then insert the current page on the stack top else shift the content of the stack down by one and insert the current page on the stack top.
9. Print the current stack.
10. Repeat steps 6 to 9 till all the content of reference string is processed
11. Calculate and print Hit ratio.
12. Stop.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Results:**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**def lru\_page\_replacement(n, reference\_string):**

**stack = [] # Initialize an empty stack**

**hits = 0**

**misses = 0**

**frame\_states = [] # To store the state of frames after each page request**

**results = [] # To store hit/miss results**

**for page in reference\_string:**

**if page in stack:**

**# Page hit**

**hits += 1**

**stack.remove(page) # Remove the page from its current position**

**stack.append(page) # Push it to the top (most recent)**

**results.append('Hit')**

**else:**

**# Page miss**

**misses += 1**

**if len(stack) < n:**

**stack.append(page) # If there's space in stack**

**else:**

**stack.pop(0) # Remove the least recently used page**

**stack.append(page) # Add the new page**

**results.append('Miss')**

**# Record the state of the frames**

**frame\_states.append(list(stack))**

**# Print the current stack**

**print(f"\nCurrent stack: {stack} | Hits: {hits} | Misses: {misses}")**

**return hits, misses, frame\_states, results**

**def main():**

**n = int(input("Enter the number of frames (n): "))**

**m = int(input("Enter the size of the reference string (m): "))**

**reference\_string = input("Enter the reference string (space-separated): ").split()**

**hits, misses, frame\_states, results = lru\_page\_replacement(n, reference\_string)**

**print(f"\nTotal Hits: {hits}")**

**print(f"Total Misses: {misses}")**

**print(f"Hit Ratio: {hits / (hits + misses) if (hits + misses) > 0 else 0:.2f}")**

**# Create a DataFrame to visualize the frame states**

**df = pd.DataFrame(frame\_states, columns=[f'Frame {i + 1}' for i in range(n)])**

**df.insert(0, 'Reference String', reference\_string)**

**df.insert(4, 'Hit/Miss', results) # Insert Hit/Miss column**

**print("\nFrame States:")**

**print(df)**

**print("\n")**

**# Plotting the frame states**

**fig, ax = plt.subplots(figsize=(10, 3))**

**ax.axis('tight')**

**ax.axis('off')**

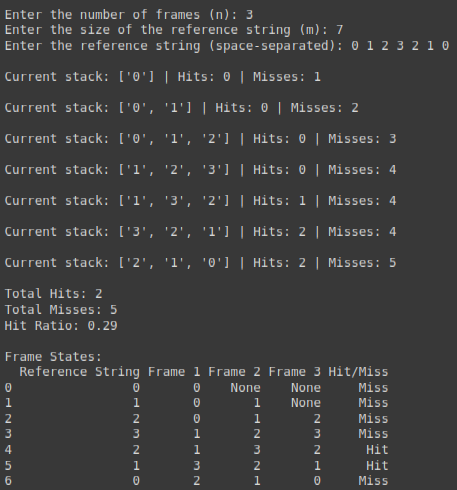
**ax.table(cellText=df.values, colLabels=df.columns, cellLoc='center', loc='center')**

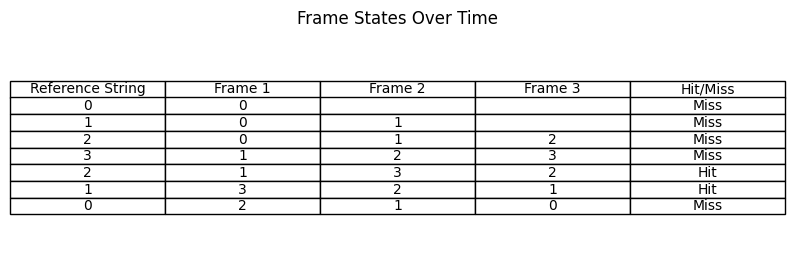
**plt.title('Frame States Over Time')**

**plt.show()**

**if \_\_name\_\_ == "\_\_main\_\_":**

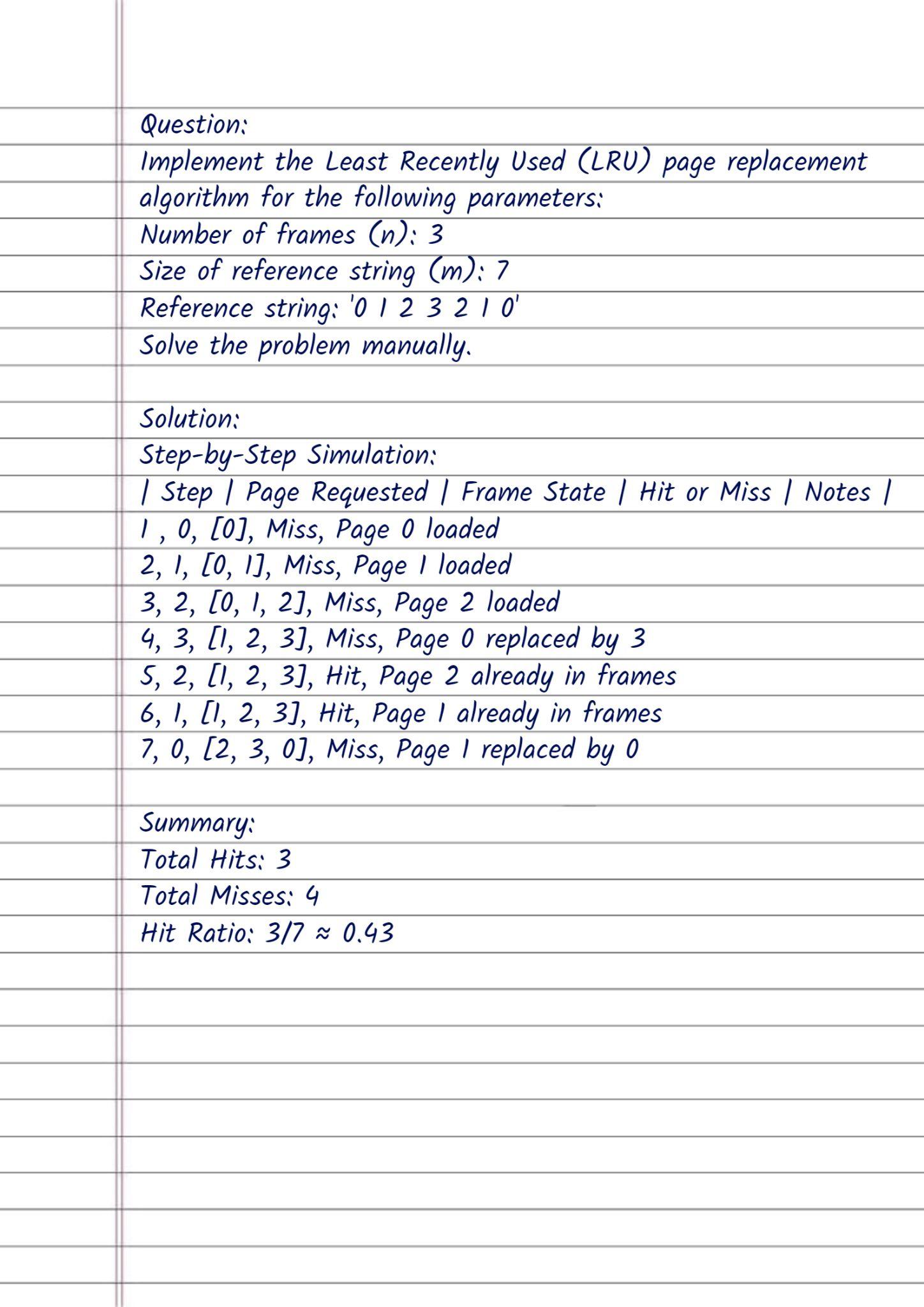
**main()**

****

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Questions:** Solve the same problem manually.



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Outcomes:** CO3 – Understand I/O management, memory management and file management

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Conclusion: (Conclusion to be based on outcomes achieved)**

The Least Recently Used (LRU) page replacement algorithm is essential for effective memory management in operating systems. By retaining the most recently accessed pages, LRU optimizes resource utilization and enhances system performance. Understanding such algorithms is crucial for improving I/O operations and overall system efficiency.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD / DD**

**Signature of faculty in-charge with date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**References:**

**Books:**

1. Applied Operating System Concepts, 1st ed. Silberschatz, Galvin and Gagne, John Wiley Publishers.
2. Modern Operating Systems, Tanenbaum, PHI.
3. Operating System, 4th Edition, William Stallings, Pearson

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**