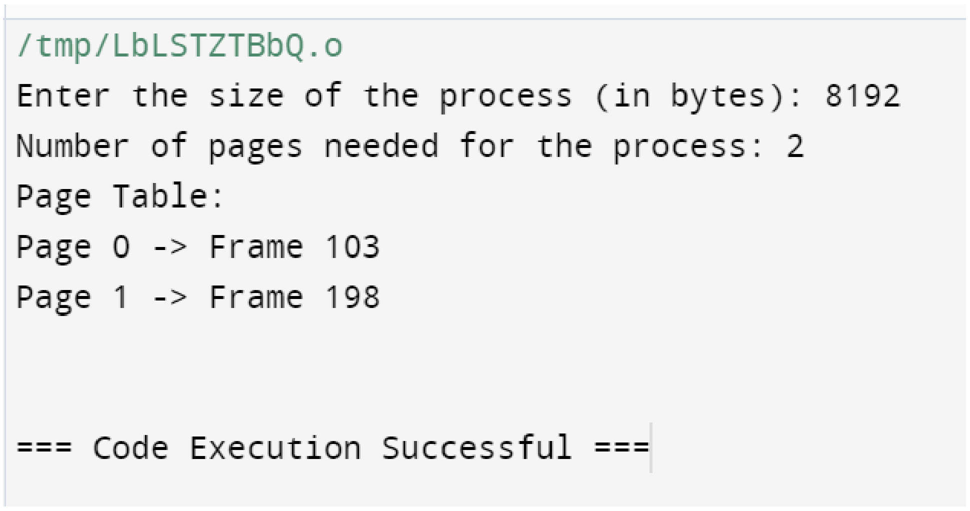
Operating system CA2

Name: Roma Shirodkar
Roll No: 56
Div: D10B

Q.1 To write a c program to implement Paging technique for memory management.
Ans.

```
#include <stdio.h>
#include <stdlib.h>
#define PAGE_SIZE 4096
#define NUM_PAGES 256

// Function to simulate paging technique for memory management
void paging(int process_size, int page_table[]) {
 int num_pages_needed = process_size / PAGE_SIZE;
 if (process_size % PAGE_SIZE != 0)
 num_pages_needed++;
 printf("Number of pages needed for the process: %d\n", num_pages_needed);
 printf("Page Table:\n");
 for (int i = 0; i < num_pages_needed; i++) {
 page_table[i] = rand() % NUM_PAGES; // Assigning random frame numbers for
simulation
 printf("Page %d -> Frame %d\n", i, page_table[i]);
 }
}


int main() {
 int process_size;
 int page_table[NUM_PAGES];
 printf("Enter the size of the process (in bytes): ");
 scanf("%d", &process_size);
 paging(process_size, page_table);
 return 0;
}
```

Output



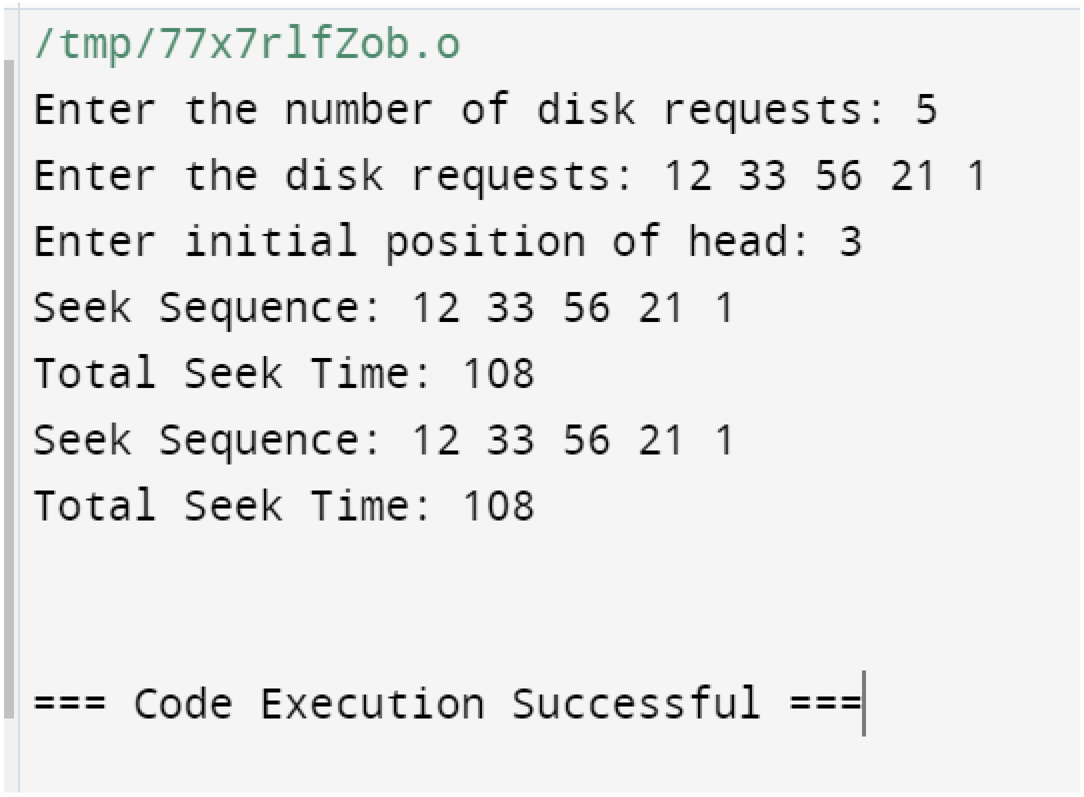
```
/tmp/LbLSTZTBbQ.o
Enter the size of the process (in bytes): 8192
Number of pages needed for the process: 2
Page Table:
Page 0 -> Frame 103
Page 1 -> Frame 198


=== Code Execution Successful ===
```

Q.2 Implement various disk scheduling algorithms like LOOK,C-LOOK in C/Python/Java.
Ans.

```
#include <stdio.h>
#include <stdlib.h>


// Function to implement LOOK disk scheduling algorithm
void look(int requests[], int head, int size) {
 int totalSeekTime = 0;
 int cur_track = head;
 printf("Seek Sequence: ");
 for (int i = 0; i < size; ++i) {
 printf("%d ", requests[i]);
 totalSeekTime += abs(cur_track - requests[i]);
 cur_track = requests[i];
 }
 printf("\nTotal Seek Time: %d\n", totalSeekTime);
}
```

```
// Function to implement C-LOOK disk scheduling algorithm
void c_look(int requests[], int head, int size) {
 int totalSeekTime = 0;
 int cur_track = head;
 printf("Seek Sequence: ");
 for (int i = 0; i < size; ++i) {
 printf("%d ", requests[i]);
 totalSeekTime += abs(cur_track - requests[i]);
 cur_track = requests[i];
 }
 printf("\nTotal Seek Time: %d\n", totalSeekTime);
}


int main() {

 int size, head;

 printf("Enter the number of disk requests: ");
 scanf("%d", &size);

 int *requests = (int *)malloc(size * sizeof(int));

 if (requests == NULL) {
 printf("Memory allocation failed.\n");
 return 1;
 }
 printf("Enter the disk requests: ");
 for (int i = 0; i < size; i++) {
 scanf("%d", &requests[i]);
 }
 printf("Enter initial position of head: ");
 scanf("%d", &head);
 // Look Algorithm
 look(requests, head, size);
 // C-Look Algorithm
 c_look(requests, head, size);
 free(requests);
 return 0;
}
```

Output



```
/tmp/77x7rlfZob.o
Enter the number of disk requests: 5
Enter the disk requests: 12 33 56 21 1
Enter initial position of head: 3
Seek Sequence: 12 33 56 21 1
Total Seek Time: 108
Seek Sequence: 12 33 56 21 1
Total Seek Time: 108


=== Code Execution Successful ===
```

Q.3 Case Study on Multimedia operating System

ANS:

Introduction:

In the realm of multimedia computing, macOS has emerged as a powerful platform that seamlessly integrates creative tools, multimedia applications, and advanced features to empower users in unleashing their creative potential. This case study explores how macOS, traditionally known for its productivity features, has evolved into a robust multimedia operating system, catering to the needs of content creators, designers, musicians, and filmmakers.

Objective:

Investigate how macOS integrates multimedia capabilities into its core functionality, providing users with a comprehensive suite of tools for multimedia creation, editing, and playback.

Core components:

- Core Audio: Core Audio is a low-level audio framework that provides macOS with audio services for multimedia playback, recording, and processing.
- Core Video: Core Video is a framework that provides macOS with video processing and rendering capabilities. It enables tasks such as video playback, decoding, encoding, and rendering.
- AVFoundation: AVFoundation is a multimedia framework that provides high-level APIs for working with audiovisual media in macOS applications.
- Image I/O is a framework that provides macOS with support for image file formats and image processing operations. It allows applications to read, write, and manipulate images in various formats, making it essential for multimedia applications that work with images, such as photo editing and graphics design software.

Key features:

1.Real-Time Processing: macOS prioritizes real-time multimedia processing to ensure smooth playback and responsiveness, minimizing latency and buffering delays. This is achieved through advanced multimedia frameworks such as Core Audio and Core Video, which provide low-latency audio and video processing capabilities.

2. Codec Support: macOS offers extensive codec support for encoding and decoding multimedia formats, ensuring compatibility with a wide range of audio, video, and image file types. This allows users to work with various multimedia content without worrying about format compatibility issues.

3. Streaming Capabilities: macOS includes built-in support for multimedia streaming protocols such as HTTP Live Streaming (HLS), RTSP, and RTMP, enabling seamless streaming of audio and video content over networks and the internet. This allows users to enjoy multimedia content from online platforms and streaming services directly on their macOS devices.

4. Hardware Acceleration: macOS integrates with hardware acceleration features of GPUs and specialized multimedia processors to enhance multimedia rendering and

processing performance. Technologies like Metal, Apple's graphics API, leverage the power of the GPU for high-performance graphics rendering and multimedia processing, ensuring smooth playback and efficient multimedia editing.

5. User Interface: macOS provides a user-friendly interface optimized for multimedia consumption, featuring intuitive controls, customizable themes, and interactive media playback options. Applications like iTunes and QuickTime Player offer intuitive interfaces for managing and playing multimedia content, while features like Picture-in-Picture (PiP) mode allow users to multitask while watching videos or participating in video calls.

Challenges:

- Software fragmentation: software fragmentation can lead to inconsistencies and interoperability issues when switching between different multimedia applications, requiring users to invest time in learning and adapting to different workflows.
- Resource Management: Multimedia applications often require significant system resources, including CPU, GPU, and memory, to deliver smooth performance and responsiveness. Inadequate resource management or inefficient resource allocation can lead to performance bottlenecks, slowdowns, and system crashes.

Conclusion:

Through its architectural design, user interface enhancements, built-in productivity tools, creativity enhancements, and robust security measures, macOS empowers users to achieve their goals efficiently and creatively while ensuring a secure computing experience. As the operating system continues to evolve, macOS remains at the forefront of innovation, driving productivity and creativity forward in the realm of operating systems studies.