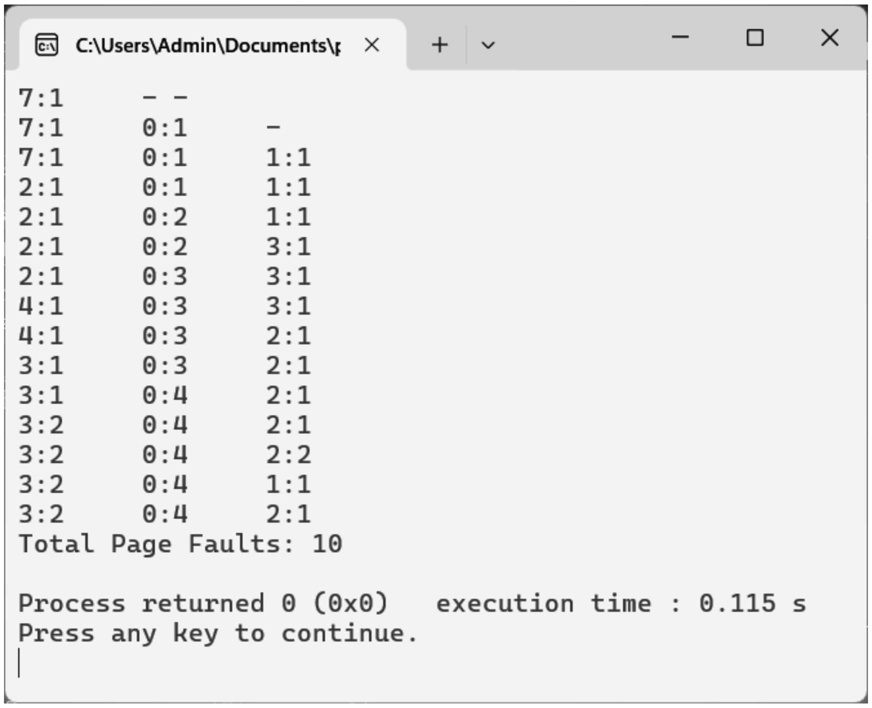
OPERATING SYSTEM
CONTINUOUS ASSESSMENT - 2

NAME: **GANESH SANJAY PANDHRE**
DIV: **D10B**
ROLL NO.: **44**

Q.1) To write a c program to implement LFU page replacement algorithm.

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

#define FRAME_SIZE 3
#define INVALID_PAGE -1

typedef struct {
    int page_number;
    int frequency;
    int timestamp;
} Page;

void initialize_frame(Page frame[], int n) {
    for (int i = 0; i < n; i++) {
        frame[i].page_number = INVALID_PAGE;
        frame[i].frequency = 0;
        frame[i].timestamp = -1;
    }
}

int find_least_frequent(Page frame[], int n) {
    int min_frequency = frame[0].frequency;
    int min_timestamp = frame[0].timestamp;
    int index = 0;

    for (int i = 1; i < n; i++) {
        if (frame[i].frequency < min_frequency ||
           (frame[i].frequency == min_frequency && frame[i].timestamp < min_timestamp)) {
            min_frequency = frame[i].frequency;
            min_timestamp = frame[i].timestamp;
            index = i;
        }
    }

    return index;
}
```

```
void print_frame(Page frame[], int n) {
    for (int i = 0; i < n; i++) {
        if (frame[i].page_number != INVALID_PAGE) {
            printf("%d:%d\t", frame[i].page_number, frame[i].frequency);
        } else {
            printf("- ");
        }
    }
    printf("\n");
}

int main() {
    int page_requests[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2};
    int n = sizeof(page_requests) / sizeof(page_requests[0]);

    Page frame[FRAME_SIZE];
    initialize_frame(frame, FRAME_SIZE);

    int page_faults = 0;
    int timestamp = 0;

    for (int i = 0; i < n; i++) {
        bool page_hit = false;

        // Check if page is already in frame
        for (int j = 0; j < FRAME_SIZE; j++) {
            if (frame[j].page_number == page_requests[i]) {
                frame[j].frequency++;
                page_hit = true;
                break;
            }
        }

        if (!page_hit) {
            // Page fault occurred
            int empty_frame = -1;
            for (int j = 0; j < FRAME_SIZE; j++) {
                if (frame[j].page_number == INVALID_PAGE) {
                    empty_frame = j;
                    break;
                }
            }

            if (empty_frame != -1) {
                frame[empty_frame].page_number = page_requests[i];
                frame[empty_frame].frequency = 1;
            frame[empty_frame].timestamp = timestamp++;
          } else {
            int least_freq_index = find_least_frequent(frame, FRAME_SIZE);
            frame[least_freq_index].page_number = page_requests[i];
            frame[least_freq_index].frequency = 1;
            frame[least_freq_index].timestamp = timestamp++;
          }

          page_faults++;
        }

        print_frame(frame, FRAME_SIZE);
    }

    printf("Total Page Faults: %d\n", page_faults);

    return 0;
}
```



```
C:\Users\Admin\Documents\p
7:1     - -
7:1     0:1     -
7:1     0:1     1:1
2:1     0:1     1:1
2:1     0:2     1:1
2:1     0:2     3:1
2:1     0:3     3:1
4:1     0:3     3:1
4:1     0:3     2:1
3:1     0:3     2:1
3:1     0:4     2:1
3:2     0:4     2:1
3:2     0:4     2:2
3:2     0:4     1:1
3:2     0:4     2:1
Total Page Faults: 10

Process returned 0 (0x0)   execution time : 0.115 s
Press any key to continue.
```

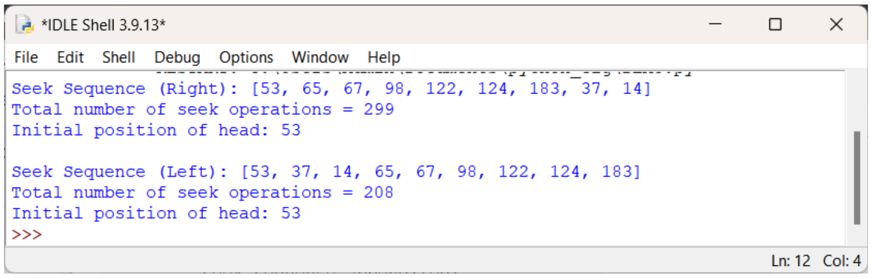
Q.2) Implement various disk scheduling algorithms like LOOK, C-LOOK in C/Python/Java.

<u>1. LOOK algorithm in python.</u>

```
def look(arr, head, direction):
    seek_sequence = []

    # Splitting requests into two parts:
```

```
    # 1. Requests below the current head position
    # 2. Requests above the current head position
    lower_requests = [req for req in arr if req < head]
    upper_requests = [req for req in arr if req > head]

    lower_requests.sort(reverse=True)
    upper_requests.sort()

    # Adding head position as the initial point
    seek_sequence.append(head)

    # Traversing in the chosen direction
    if direction == "left":
        for req in lower_requests:
            seek_sequence.append(req)

        for req in upper_requests:
            seek_sequence.append(req)
    else:
        for req in upper_requests:
            seek_sequence.append(req)

        for req in lower_requests:
            seek_sequence.append(req)

    return seek_sequence

def calculate_seek_operations(sequence):
    operations = 0
    for i in range(1, len(sequence)):
        operations += abs(sequence[i] - sequence[i-1])
    return operations

# Example
requests = [98, 183, 37, 122, 14, 124, 65, 67]
initial_head = 53

initial_direction = "right"

sequence = look(requests, initial_head, initial_direction)
print("Seek Sequence (Right):", sequence)
print("Total number of seek operations =", calculate_seek_operations(sequence))
print("Initial position of head:", initial_head)

initial_direction = "left"
```

```
sequence = look(requests, initial_head, initial_direction)
print("\nSeek Sequence (Left):", sequence)
print("Total number of seek operations =", calculate_seek_operations(sequence))
print("Initial position of head:", initial_head)
```

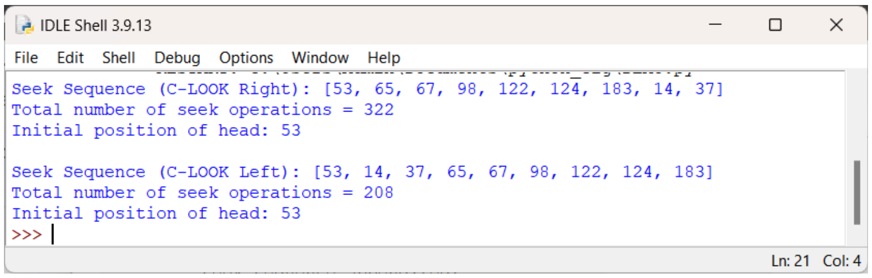


```
Seek Sequence (Right): [53, 65, 67, 98, 122, 124, 183, 37, 14]
Total number of seek operations = 299
Initial position of head: 53

Seek Sequence (Left): [53, 37, 14, 65, 67, 98, 122, 124, 183]
Total number of seek operations = 208
Initial position of head: 53
>>>
```

2. C-LOOK algorithm in python.

```
def c_look(arr, head, direction):
    seek_sequence = []

    # Splitting requests into two parts:
    # 1. Requests below the current head position
    # 2. Requests above the current head position
    lower_requests = [req for req in arr if req < head]
    upper_requests = [req for req in arr if req > head]

    lower_requests.sort()
    upper_requests.sort()

    # Adding head position as the initial point
    seek_sequence.append(head)

    # Traversing in the chosen direction
    if direction == "left":
        for req in lower_requests:
            seek_sequence.append(req)

        for req in upper_requests:
            seek_sequence.append(req)
    else:
        for req in upper_requests:
            seek_sequence.append(req)

        for req in lower_requests:
            seek_sequence.append(req)

    return seek_sequence
```

```
def calculate_seek_operations(sequence):
    operations = 0
    for i in range(1, len(sequence)):
        operations += abs(sequence[i] - sequence[i-1])
    return operations

# Example
requests = [98, 183, 37, 122, 14, 124, 65, 67]
initial_head = 53

initial_direction = "right"

sequence = c_look(requests, initial_head, initial_direction)
print("Seek Sequence (C-LOOK Right):", sequence)
print("Total number of seek operations =", calculate_seek_operations(sequence))
print("Initial position of head:", initial_head)

initial_direction = "left"

sequence = c_look(requests, initial_head, initial_direction)
print("\nSeek Sequence (C-LOOK Left):", sequence)
print("Total number of seek operations =", calculate_seek_operations(sequence))
print("Initial position of head:", initial_head)
```



```
Seek Sequence (C-LOOK Right): [53, 65, 67, 98, 122, 124, 183, 14, 37]
Total number of seek operations = 322
Initial position of head: 53

Seek Sequence (C-LOOK Left): [53, 14, 37, 65, 67, 98, 122, 124, 183]
Total number of seek operations = 208
Initial position of head: 53
>>> 
```

Q.3) Case Study on Mobile Operating System.

Case Study: **iOS** Mobile Operating System

Introduction:

iOS, developed by Apple Inc., is one of the most recognized and widely used mobile operating systems globally. This case study delves into the history, market share, user interface, app ecosystem, security, and key features of the iOS operating system.

History:

- Developed and exclusively owned by Apple Inc.
- The first iPhone, running iOS, was launched in 2007, revolutionizing the smartphone industry with its innovative design and user interface.

Market Share:

- While iOS holds a smaller overall market share compared to Android, it dominates the premium smartphone market.
- Strong brand loyalty, particularly in markets like the United States and Europe, contributes to its consistent performance and user base.

User Interface:

- iOS offers a uniform and consistent user interface across all Apple devices, including iPhones, iPads, and iPod Touch.
- The user interface is known for its simplicity, intuitive design, and ease of use, appealing to a wide range of users from tech novices to experts.

App Ecosystem:

- Apple's App Store is the exclusive source for iOS apps, known for its strict app review process and curated selection of high-quality apps.
- The App Store offers a vast array of apps and games across various categories, including productivity, entertainment, education, and more.

Security:

- iOS is renowned for its robust security features and commitment to user privacy.
- The closed-source nature of iOS, combined with Apple's stringent control over hardware and software integration, contributes to its reputation for superior security.
- Features like Face ID, Touch ID, end-to-end encryption, and regular software updates enhance the platform's security and protect users' data and devices from potential threats.

Key Features:

- Seamless Integration with Apple Ecosystem: iOS devices seamlessly integrate with other Apple products and services, such as iCloud, iMessage, FaceTime, and Apple Music, enhancing productivity and user experience.
- Siri: Apple's intelligent virtual assistant, Siri, provides voice-activated controls and personalized recommendations, improving user interaction and device functionality.
- Accessibility: iOS offers a comprehensive suite of accessibility features, including VoiceOver, Magnifier, and AssistiveTouch, ensuring that users with disabilities can easily navigate and use their devices.

Conclusion:

iOS, with its focus on simplicity, security, and seamless integration with the Apple ecosystem, has solidified its position as a leading mobile operating system, particularly in the premium smartphone market. While it may not offer the same level of customization as Android, its consistent user experience, high-quality app ecosystem, and strong commitment to privacy and security appeal to millions of users worldwide. As Apple continues to innovate and expand its product lineup, including iPhones, iPads, Macs, and wearables, iOS is expected to maintain its relevance and attract new users who value a premium and cohesive ecosystem.