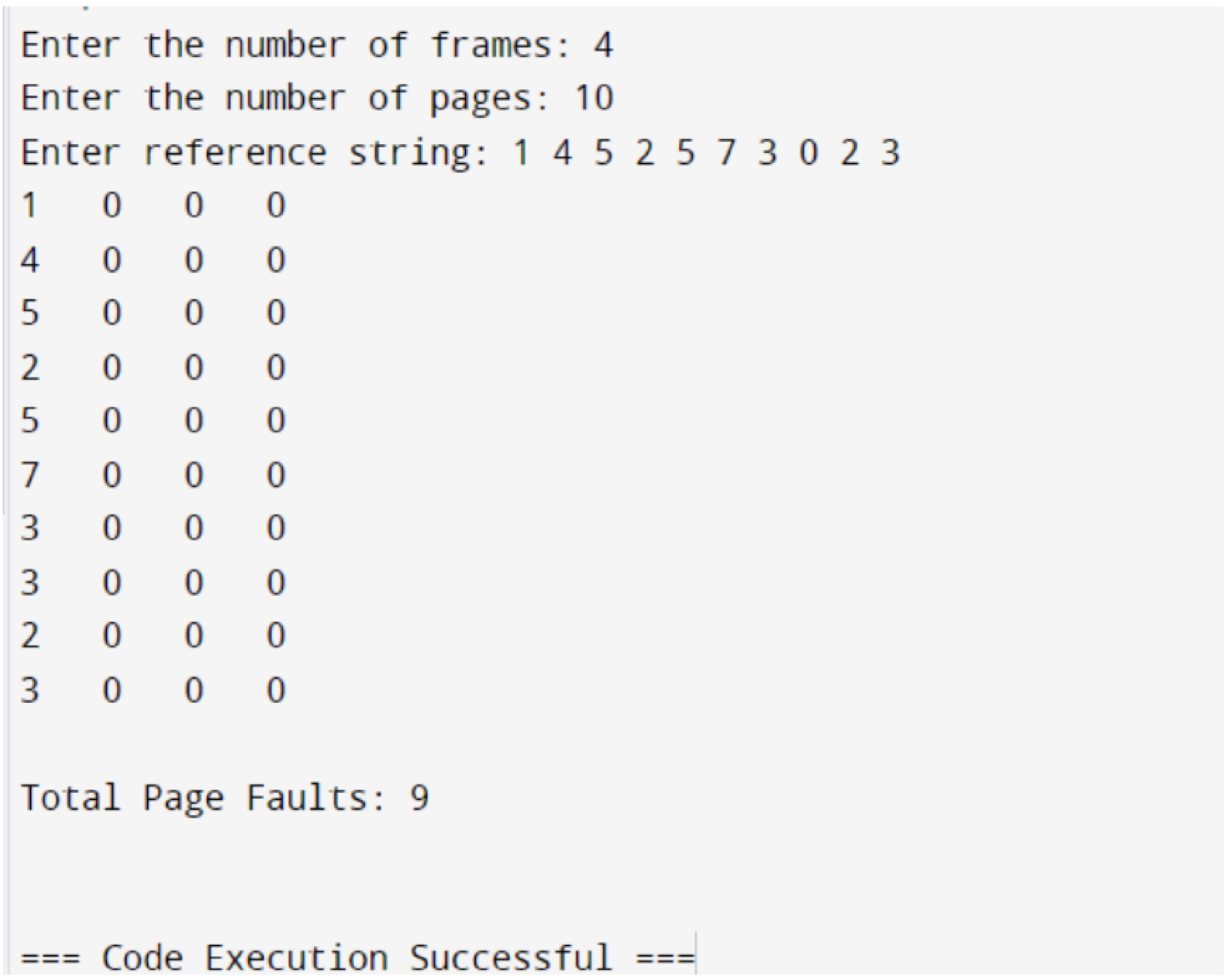
Name : Saachi Raheja
Class : D10B
Roll no.:49

Operating system CA -2

Q1. C program to implement LFU page replacement algorithm

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

#define MAX_PAGES 100

typedef struct {
    int pageNumber;
    int frequency;
    int time;
} Page;

int main() {
    int numFrames, numPages, pageFaults = 0;
    int frame[MAX_PAGES];
    Page pages[MAX_PAGES];

    printf("Enter the number of frames: ");
    scanf("%d", &numFrames);

    printf("Enter the number of pages: ");
    scanf("%d", &numPages);

    printf("Enter reference string: ");
    for (int i = 0; i < numPages; ++i) {
        scanf("%d", &pages[i].pageNumber);
        pages[i].frequency = 0;
        pages[i].time = -1;
    }

    for (int i = 0; i < numPages; ++i) {
        int flag = 0;
        for (int j = 0; j < numFrames; ++j) {
            if (frame[j] == pages[i].pageNumber) {
                flag = 1;
                pages[i].frequency++;
                break;
            }
        }
        if (flag == 0) {
            int minimum = 0;
            for (int j = 1; j < numFrames; ++j) {
                if (pages[frame[j]].frequency < pages[frame[minimum]].frequency)
                    minimum = j;
                else if (pages[frame[j]].frequency == pages[frame[minimum]].frequency &&
                         pages[frame[j]].time > pages[frame[minimum]].time)
                    minimum = j;
            }
            frame[minimum] = pages[i].pageNumber;
            pages[i].frequency++;
            pageFaults++;
        }
        for (int k = 0; k < numFrames; ++k) {
            printf("%d\t", frame[k]);
        }
        printf("\n");
    }
    printf("\nTotal Page Faults: %d\n", pageFaults);
    return 0;
}
```

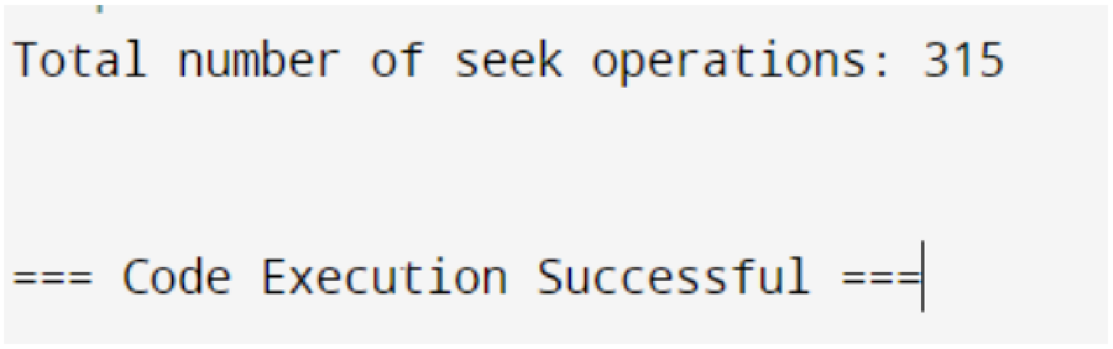


```
Enter the number of frames: 4
Enter the number of pages: 10
Enter reference string: 1 4 5 2 5 7 3 0 2 3
1   0   0   0
4   0   0   0
5   0   0   0
2   0   0   0
5   0   0   0
7   0   0   0
3   0   0   0
3   0   0   0
2   0   0   0
3   0   0   0

Total Page Faults: 9


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

Q2. Implement various disk scheduling algorithms like LOOK , C-LOOK:

LOOK:

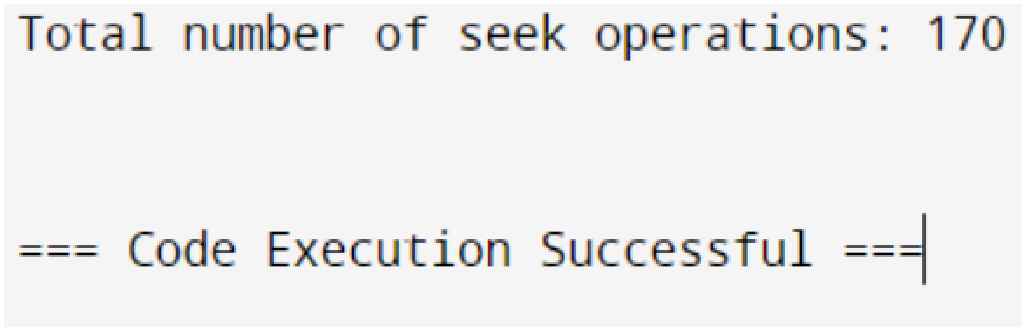
```
#include <stdio.h>
#include <stdlib.h>

void look(int arr[], int head, int size) {
    int seek_count = 0;
    int distance, cur_track;

    // Sorting disk request array
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    int left = 0, right = 0;

    // Finding the index of the first request larger than head
    for (int i = 0; i < size; i++) {
        if (arr[i] > head) {
            right = i;
            break;
        }
    }
    left = right - 1;

    // Iterating towards right from the initial position of head
    for (int i = right; i < size; i++) {
        cur_track = arr[i];
        distance = abs(cur_track - head);
        seek_count += distance;
        head = cur_track;
    }
    // Iterating towards left from the initial position of head
    for (int i = left; i >= 0; i--) {
        cur_track = arr[i];
        distance = abs(cur_track - head);
        seek_count += distance;
        head = cur_track;
    }

    printf("Total number of seek operations: %d\n", seek_count);
}

int main() {
    // Head
    int head = 50;

    // Disk requests
    int requests[] = {20, 78, 192, 37, 55, 154, 19, 30};
    int size = sizeof(requests) / sizeof(requests[0]);

    look(requests, head, size);

    return 0;
}
```



```
Total number of seek operations: 315


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

CLOOK:

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

// Function to implement C-LOOK Disk Scheduling Algorithm
void clook(int arr[], int head, int size) {
    int seek_count = 0;
    int distance, cur_track;

    // Sorting disk request array
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    int left = 0, right = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] > head) {
            right = i;
            break;
        }
    }
    left = right - 1;

    // Iterating towards right from the initial position of head
    for (int i = right; i < size; i++) {
        cur_track = arr[i];
        distance = abs(cur_track - head);
        seek_count += distance;
        head = cur_track;
    }

    // Iterating towards left from the initial position of head (C-SCAN)
    head = 0;
    for (int i = 0; i <= left; i++) {
        cur_track = arr[i];
        distance = abs(cur_track - head);
        seek_count += distance;
        head = cur_track;
    }

    printf("Total number of seek operations: %d\n", seek_count);
}

// Driver code
int main() {
    // Head
    int head = 50;
    // Disk requests
    int requests[] = {98, 183, 37, 122, 14, 124, 65, 67};
    int size = sizeof(requests) / sizeof(requests[0]);
    clook(requests, head, size);
    return 0;
}
```



```
Total number of seek operations: 170


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

Q3. Case study on Mobile operating system

Introduction:
Mobile operating systems (OS) have experienced significant evolution since the advent of smartphones, fundamentally altering how we communicate, work, and interact with technology. This report provides an in-depth analysis of the evolutionary milestones of seven prominent mobile operating systems: Android, iOS, KaiOS, HarmonyOS, Windows Mobile, BlackBerry OS, and Symbian.

1. Android:
Android, with versions like Cupcake, Ice Cream Sandwich, and Oreo, has continually introduced innovative features such as on-screen keyboards, multitasking enhancements, and performance optimizations.

2. iOS:
iOS has evolved through versions like iOS 2.0 and iOS 11, introducing groundbreaking features like the App Store, multitasking, and ARKit, elevating the user experience and functionality.

3. KaiOS:
KaiOS, progressing from Version 1.0 to 3.0, has expanded its app support and performance optimizations, catering to the needs of feature phone users with essential functionalities and digital service partnerships.

4. HarmonyOS:
HarmonyOS transitioned from Version 1.0, focusing on IoT devices, to Version 2.0, extending support to smartphones and tablets, with features like distributed file systems and collaborative capabilities.

5. Windows Mobile:
Windows Mobile, spanning versions like Windows Mobile 5.0 and Windows Phone 7, introduced improvements such as Office Mobile integration and Live Tiles for dynamic updates.

6. BlackBerry OS:
BlackBerry OS versions like BlackBerry OS 4.5 and BlackBerry 10 improved email support, introduced threaded messaging, and embraced Android app compatibility, marking a significant shift in the platform's direction.

7. Symbian:
Symbian OS versions, including Symbian OS 9.1 and Symbian OS 9.3, brought enhancements like Wi-Fi support and improved graphics performance, contributing to the platform's competitiveness.

Key Features:
1. User Interface (UI) Design: Intuitive interfaces optimized for touch interactions.
2. App Ecosystem: Robust platforms offering a vast array of applications.
3. Customization Options: Personalized device settings and appearance customization.
4. Security and Privacy: Implementation of robust security measures to protect user data.
5. Integration with Ecosystem Services: Seamless access to interconnected services across devices.
6. Performance Optimization: Techniques to ensure smooth operation and responsiveness.
7. Updates and Support: Regular updates and support for improved stability and features.

Impact:
1. Digital Connectivity: Enabling global communication and access to information.
2. Economic Growth: Driving innovation, entrepreneurship, and job creation.
3. Empowerment: Bridging the digital divide and promoting inclusion.
4. Social Interaction: Fostering real-time communication and community engagement.
5. Productivity and Efficiency: Empowering users with tools for on-the-go productivity.
6. Entertainment and Recreation: Offering diverse entertainment options for leisure.
7. Technological Innovation: Driving advancements in hardware, software, and user experience.

Future Outlook:
Continuous innovation and adaptation to user needs will be crucial for mobile OS developers to stay competitive in the rapidly evolving digital landscape. Emphasizing security, privacy, and seamless integration with emerging technologies will be essential for sustaining growth and relevance in the market.

With advancements in artificial intelligence, augmented reality, and 5G technology, mobile operating systems are poised to undergo further transformation, unlocking new possibilities for digital interaction, productivity, and entertainment. Collaboration across platforms and ecosystems will likely play a pivotal role in driving future innovation and shaping the mobile landscape.

Conclusion:

Mobile operating systems are indispensable in modern digital interactions, offering diverse features and benefits that empower users, drive economic growth, and foster innovation. From Android's versatility to iOS's ecosystem integration, each OS has significantly influenced the mobile landscape. As technology progresses, these platforms will continue to evolve, shaping the future of digital connectivity and innovation, and playing a crucial role in shaping the modern digital landscape.