**OPERATING SYSTEM**

**CONTINUOUS ASSESSMENT - 2**

NAME: **Ajinkya Sunil Patil**

DIV: **D10B**

ROLL NO.: **46**

**Q.1) To write a c program to implement the 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++;

}

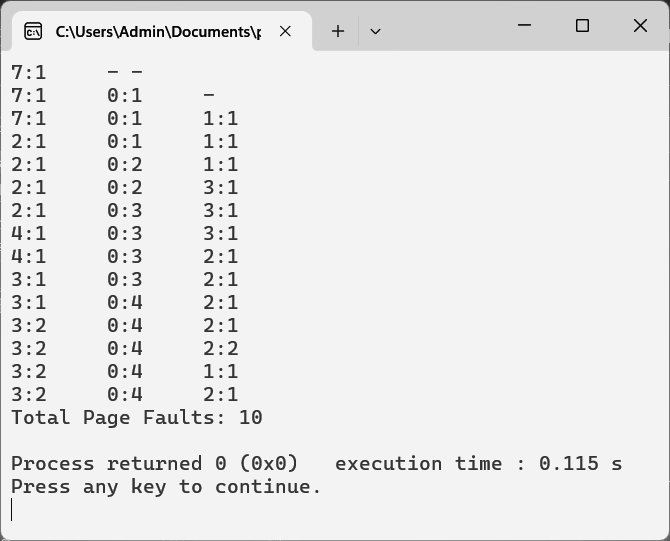
print\_frame(frame, FRAME\_SIZE);

}

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

return 0;

}



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

1. LOOK algorithm in python.

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)

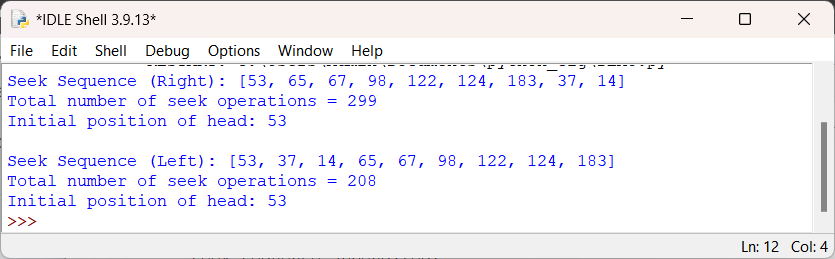
initial\_direction = "left"

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)



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)

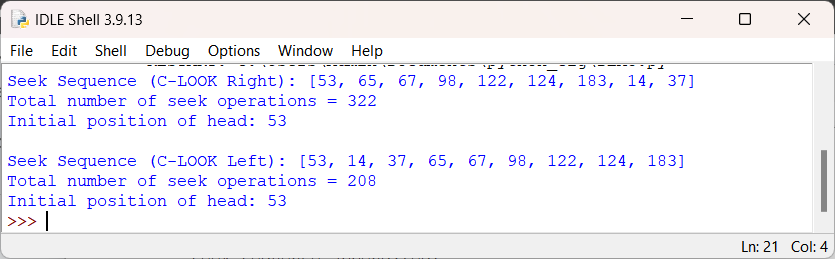
initial\_direction = "left"

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)



**Q.3) Case Study on Mobile Operating System.**

**Case Study: HarmonyOS Mobile Operating System**

Introduction:

HarmonyOS, developed by Huawei, is a relatively new entrant in the mobile operating system landscape. Designed as a versatile, open-source platform, HarmonyOS aims to provide a seamless and integrated user experience across a wide range of devices, including smartphones, tablets, smart TVs, wearables, and IoT devices. This case study examines the history, market strategy, key features, app ecosystem, security, and potential challenges of HarmonyOS.

History:

* Developed by Huawei as an alternative to Android due to trade restrictions and geopolitical challenges.
* Officially launched in 2019, with a focus on creating a unified operating system for various smart devices.

Market Strategy:

* Initially targeted at the Chinese market, where Huawei has a strong presence and brand recognition.
* Plans to expand globally and collaborate with international developers, manufacturers, and partners to build a robust and diverse ecosystem.

Key Features:

* Adaptive UI: HarmonyOS offers an adaptive user interface that can adjust seamlessly to different device types and screen sizes, providing a consistent and intuitive user experience across devices.
* Distributed Capabilities: Unique distributed architecture allows HarmonyOS to leverage the combined capabilities of multiple devices, enabling features like seamless multi-device collaboration, real-time sharing, and synchronized app experiences.
* High Performance: Optimized performance and resource management, ensuring smooth and responsive operation across various hardware platforms and device categories.

App Ecosystem:

* AppGallery is Huawei's official app store, aiming to compete with established app stores like Google Play Store and Apple's App Store.
* Growing ecosystem with a focus on attracting both local and international developers through incentives, support, and collaboration opportunities.
* Compatibility with Android apps through Huawei Mobile Services (HMS) and Huawei's AppGallery Connect, allowing users to access a vast library of apps while ensuring security and quality.

Security:

* Emphasizes privacy and security with features like a microkernel architecture, enhanced encryption methods, and rigorous security protocols.
* Regular security updates, collaboration with global security organizations, and transparent communication with users and developers to build trust and credibility.

Potential Challenges:

* Ecosystem Building: Establishing a comprehensive and diverse ecosystem of apps, services, and devices is crucial for HarmonyOS's success and adoption.
* Geopolitical Challenges: Ongoing trade restrictions and geopolitical tensions may impact Huawei's ability to collaborate with international partners, access key technologies, and expand its global footprint.
* Competition: Facing strong competition from established mobile operating systems like Android and iOS, as well as emerging platforms like Samsung's Tizen and Google's Fuchsia OS, requires innovative strategies, differentiation, and value proposition to attract users and developers.

Conclusion:

HarmonyOS represents Huawei's ambitious vision for a unified and integrated operating system that transcends traditional boundaries and enables a seamless digital experience across various smart devices. While it faces significant challenges and competition in the highly competitive mobile operating system market, HarmonyOS's innovative features, commitment to privacy and security, and strategic initiatives to build a vibrant ecosystem position it as a potential disruptor and alternative to established players. As Huawei continues to invest in R&D, partnerships, and global expansion, HarmonyOS's evolution, adoption, and impact on the future of connected devices will be closely watched by industry stakeholders, consumers, and tech enthusiasts alike.