Name : Hanuman Bavane  
Roll No :13109  
Batch : A1  
  
 Practical 7

def print\_frame\_table(table, noofpages, capacity):

    print("\n----------------------------------------------------------------------")

    for i in range(capacity):

        for j in range(noofpages):

            if table[j][i] == -1:

                print("   -  ", end="")

            else:

                print(f"{table[j][i]:5d} ", end="")

        print()

    print("----------------------------------------------------------------------")

def fifo\_page\_replacement(noofpages, pages, capacity):

    frame = [-1] \* capacity

    table = [[-1] \* capacity for \_ in range(noofpages)]

    hit, fault, index = 0, 0, 0

    print("\n----------------------------------------------------------------------")

    for i in range(noofpages):

        search = -1

        for j in range(capacity):

            if frame[j] == pages[i]:

                search = j

                hit += 1

                print("  H  ", end="")

                break

        if search == -1:

            frame[index] = pages[i]

            fault += 1

            print("  F  ", end="")

            index = (index + 1) % capacity

        for j in range(capacity):

            table[i][j] = frame[j]

    print("\n----------------------------------------------------------------------")

    print\_frame\_table(table, noofpages, capacity)

    fault\_ratio = (fault / noofpages) \* 100

    hit\_ratio = (hit / noofpages) \* 100

    print(f"Page Fault: {fault}\nPage Hit: {hit}")

    print(f"Hit Ratio: {hit\_ratio:.2f}% \nFault Ratio: {fault\_ratio:.2f}%")

def least\_recently\_used(noofpages, pages, capacity):

    frame = [-1] \* capacity

    table = [[-1] \* capacity for \_ in range(noofpages)]

    arr = []

    hit, fault, index = 0, 0, 0

    isFull = False

    print("\n----------------------------------------------------------------------")

    for i in range(noofpages):

        if pages[i] in arr:

            arr.remove(pages[i])

        arr.append(pages[i])

        search = -1

        for j in range(capacity):

            if frame[j] == pages[i]:

                search = j

                hit += 1

                print("  H  ", end="")

                break

        if search == -1:

            if isFull:

                min\_loc = noofpages

                for j in range(capacity):

                    if frame[j] in arr:

                        temp = arr.index(frame[j])

                        if temp < min\_loc:

                            min\_loc = temp

                            index = j

            frame[index] = pages[i]

            fault += 1

            print("  F  ", end="")

            index += 1

            if index == capacity:

                index = 0

                isFull = True

        for j in range(capacity):

            table[i][j] = frame[j]

    print("\n----------------------------------------------------------------------")

    print\_frame\_table(table, noofpages, capacity)

    hit\_ratio = (hit / noofpages) \* 100

    fault\_ratio = (fault / noofpages) \* 100

    print(f"Page Fault: {fault}\nPage Hit: {hit}")

    print(f"Hit Ratio: {hit\_ratio:.2f}% \nFault Ratio: {fault\_ratio:.2f}%")

def optimal\_page\_replacement(noofpages, pages, capacity):

    frame = [-1] \* capacity

    table = [[-1] \* capacity for \_ in range(noofpages)]

    hit, fault, ptr = 0, 0, 0

    is\_full = False

    print("\n----------------------------------------------------------------------")

    for i in range(noofpages):

        search = -1

        for j in range(capacity):

            if frame[j] == pages[i]:

                search = j

                hit += 1

                print("  H  ", end="")

                break

        if search == -1:

            if is\_full:

                index = [None] \* capacity

                for j in range(i + 1, noofpages):

                    for k in range(capacity):

                        if pages[j] == frame[k] and index[k] is None:

                            index[k] = j

                            break

                max\_index = -1

                ptr = -1

                for j in range(capacity):

                    if index[j] is None:

                        ptr = j

                        break

                    elif index[j] > max\_index:

                        max\_index = index[j]

                        ptr = j

            frame[ptr] = pages[i]

            fault += 1

            print("  F  ", end="")

            if not is\_full:

                ptr += 1

                if ptr == capacity:

                    ptr = 0

                    is\_full = True

        for j in range(capacity):

            table[i][j] = frame[j]

    print("\n----------------------------------------------------------------------")

    print\_frame\_table(table, noofpages, capacity)

    hit\_ratio = (hit / noofpages) \* 100

    fault\_ratio = (fault / noofpages) \* 100

    print(f"Page Fault: {fault}\nPage Hit: {hit}")

    print(f"Hit Ratio: {hit\_ratio:.2f}% \nFault Ratio: {fault\_ratio:.2f}%")

# Function to choose algorithm based on user choice

def page\_replacement\_simulation():

    while True:

        print("\n1. FIFO Page Replacement")

        print("2. LRU Page Replacement")

        print("3. Optimal Page Replacement")

        print("4. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 4:

            print("Exiting program.")

            break

        noofpages = int(input("Enter the number of pages you want to enter: "))

        pages = list(map(int, input("Enter the pages (space-separated): ").split()))

        capacity = int(input("Enter the capacity of frame: "))

        if choice == 1:

            fifo\_page\_replacement(noofpages, pages, capacity)

        elif choice == 2:

            least\_recently\_used(noofpages, pages, capacity)

        elif choice == 3:

            optimal\_page\_replacement(noofpages, pages, capacity)

        else:

            print("Invalid choice, please select again.")

# Run the function with switch cases

page\_replacement\_simulation()

OUTPUT :

PS C:\Users\HP> & "C:/Program Files/Python312/python.exe" "c:/Users/HP/OneDrive/Desktop/spos practical/lexical/fifo.py"

1. FIFO Page Replacement

2. LRU Page Replacement

3. Optimal Page Replacement

4. Exit

Enter your choice: 1

Enter the number of pages you want to enter: 20

Enter the pages (space-separated): 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 3 3 6

Enter the capacity of frame: 3

----------------------------------------------------------------------------------------------------------

F F F F H F F F F F H F F F H F F F H F

----------------------------------------------------------------------------------------------------------

1 1 1 4 4 4 4 6 6 6 6 3 3 3 3 2 2 2 2 6

- 2 2 2 2 1 1 1 2 2 2 2 7 7 7 7 1 1 1 1

- - 3 3 3 3 5 5 5 1 1 1 1 6 6 6 6 3 3 3

----------------------------------------------------------------------------------------------------------

Page Fault: 16

Page Hit: 4

Hit Ratio: 20.00%

Fault Ratio: 80.00%

1. FIFO Page Replacement

2. LRU Page Replacement

3. Optimal Page Replacement

4. Exit

Enter your choice: 2

Enter the number of pages you want to enter: 20

Enter the pages (space-separated): 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 3 3 6

Enter the capacity of frame: 3

----------------------------------------------------------------------------------------------------------

F F F F H F F F F F H F F F H F F H H F

----------------------------------------------------------------------------------------------------------

1 1 1 4 4 4 5 5 5 1 1 1 7 7 7 2 2 2 2 6

- 2 2 2 2 2 2 6 6 6 6 3 3 3 3 3 3 3 3 3

- - 3 3 3 1 1 1 2 2 2 2 2 6 6 6 1 1 1 1

----------------------------------------------------------------------------------------------------------

Page Fault: 15

Page Hit: 5

Hit Ratio: 25.00%

Fault Ratio: 75.00%

1. FIFO Page Replacement

2. LRU Page Replacement

3. Optimal Page Replacement

4. Exit

Enter your choice: 3

Enter the number of pages you want to enter: 20

Enter the pages (space-separated): 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 3 3 6

Enter the capacity of frame: 3

----------------------------------------------------------------------------------------------------------

F F F F H H F F H H H F F H H F F H H H

----------------------------------------------------------------------------------------------------------

1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3

- 2 2 2 2 2 2 2 2 2 2 2 7 7 7 2 1 1 1 1

- - 3 4 4 4 5 6 6 6 6 6 6 6 6 6 6 6 6 6

----------------------------------------------------------------------------------------------------------

Page Fault: 10

Page Hit: 10

Hit Ratio: 50.00%

Fault Ratio: 50.00%

1. FIFO Page Replacement

2. LRU Page Replacement

3. Optimal Page Replacement

4. Exit

Enter your choice: 4

Exiting program.

PS C:\Users\HP>

Conclusion :

Among the three algorithms tested, the Optimal Page Replacement algorithm yielded the best performance, significantly reducing page faults and enhancing the efficiency of memory usage. FIFO performed the poorest, illustrating the limitations of a straightforward replacement strategy that does not account for page usage patterns. LRU provided a middle ground, offering improved performance through a more adaptive approach.