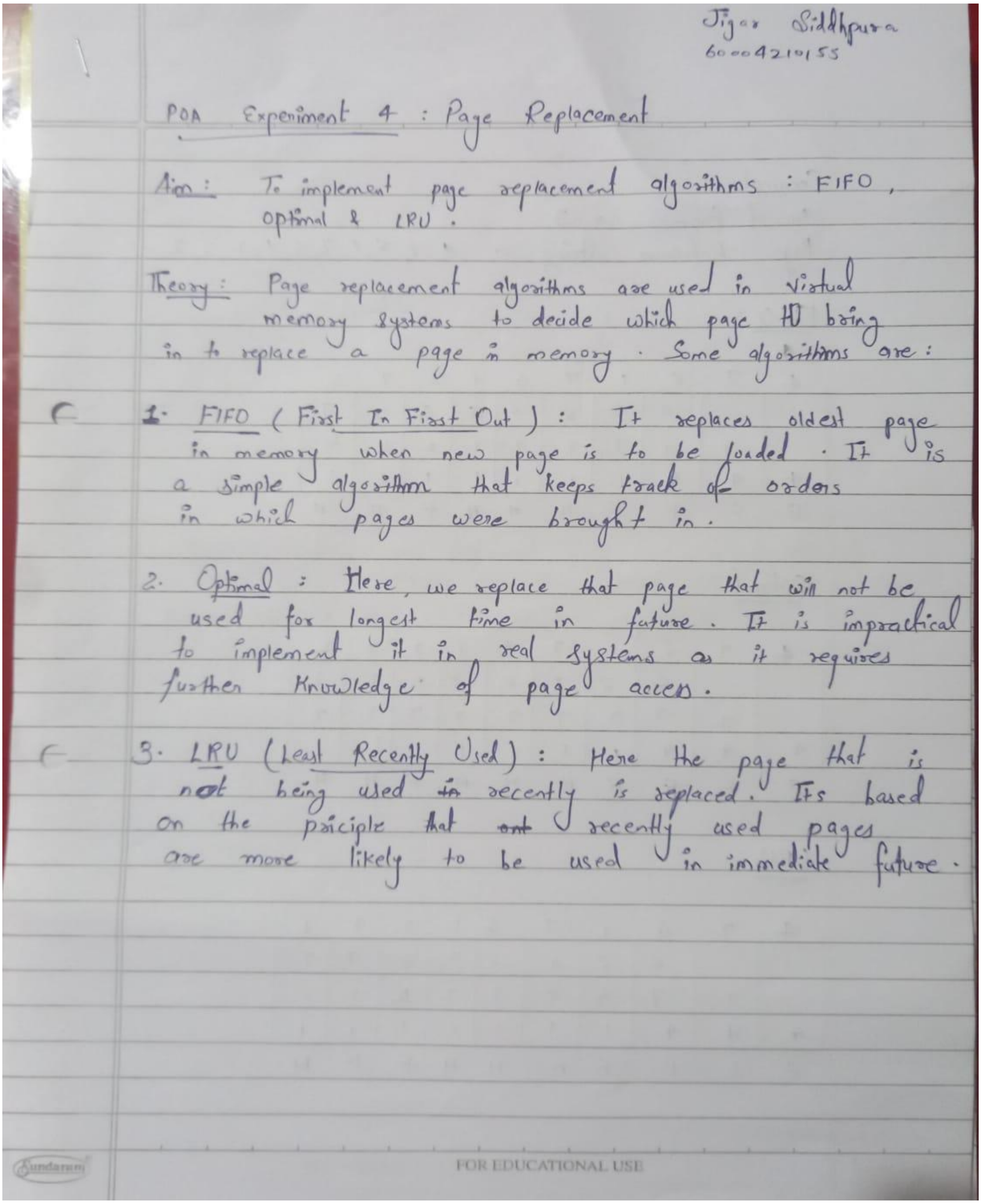
Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

POA EXPERIMENT 3



Jigar Siddhpura
60004210155

POA Experiment 4 : Page Replacement

Aim : To implement page replacement algorithms : FIFO, optimal & LRU.

Theory : Page replacement algorithms are used in virtual memory systems to decide which page to bring in to replace a page in memory. Some algorithms are :

1. FIFO (First In First Out) : It replaces oldest page in memory when new page is to be loaded. It is a simple algorithm that keeps track of orders in which pages were brought in.

2. Optimal : Here, we replace that page that will not be used for longest time in future. It is impractical to implement it in real systems as it requires further knowledge of page access.

3. LRU (Least Recently Used) : Here the page that is not being used ~~in~~ recently is replaced. Its based on the priciple that ~~ont~~ recently used pages are more likely to be used in immediate future.

Example :

No. of frames = 3

Page Refrence String = 4, 7, 6, 1, 7, 6, 1, 2, 7, 2

FIFO :

4	7	6	1	7	6	1	2	7	2
		6	6	6	6	6	6	7	7
	7	7	7	7	7	7	2	2	2
4	4	4	1	1	1	1	1	1	1
F	F	F	F	H	H	H	F	F	H

Optimal :

4	7	6	1	7	6	1	2	7	2
		6	6	6	6	6	2	2	2
	7	7	7	7	7	7	7	7	7
4	4	4	1	1	1	1	1	1	1
F	F	F	F	H	H	H	F	H	H

LRU :

4	7	6	1	7	6	1	2	7	2
		6	6	6	6	6	6	7	7
	7	7	7	7	7	7	2	2	2
4	4	4	1	1	1	1	1	1	1
F	F	F	F	H	H	H	F	F	H

Observation :

FIFO : Page fault = 6 , Page hit = 4
∴ Hit ratio = 0.4 , Fault ratio = 0.6

Optimal : Page fauit = 5 , Page hit = 5
∴ Hit ratio = Fault ratio = 0.5

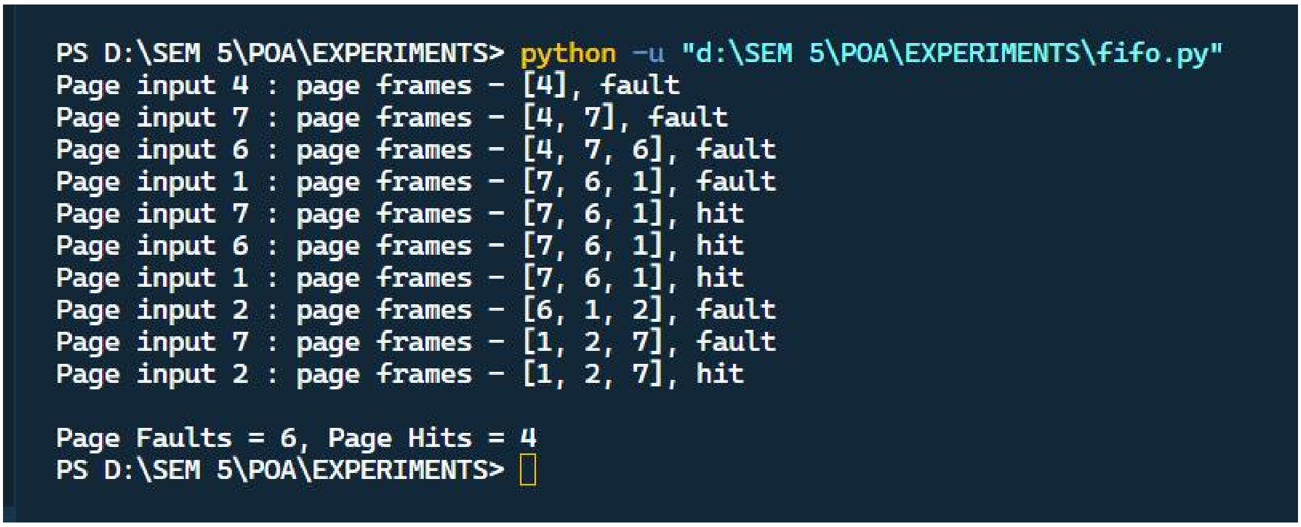
LPU : Page fault = 6 , Page hit = 4
∴ Hit ratio = 0.4 , Fault ratio = 0.6

Conclusion : Hence, we implement page replacement algorithms using FIFO, LRU & optimal & Calculated page fault, page hit, hit ratio & fault ratio.

FIFO:

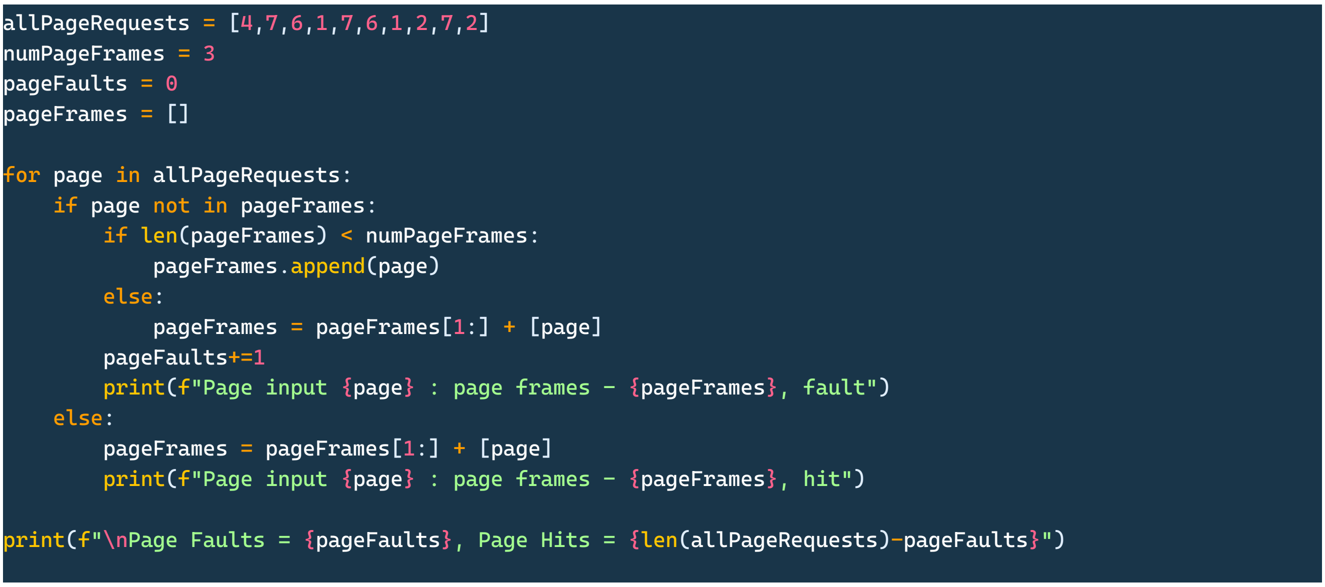
```
allPageRequests = [4,7,6,1,7,6,1,2,7,2]
numPageFrames = 3
pageFaults = 0
pageFrames = []

for page in allPageRequests:
    if page not in pageFrames:
        if len(pageFrames) < numPageFrames:
            pageFrames.append(page)
        else:
            pageFrames = pageFrames[1:] + [page]
        pageFaults+=1
        print(f"Page input {page} : page frames - {pageFrames}, fault")
    else:
        print(f"Page input {page} : page frames - {pageFrames}, hit")

print(f"\nPage Faults = {pageFaults}, Page Hits = {len(allPageRequests)-pageFaults}")
```

Output :



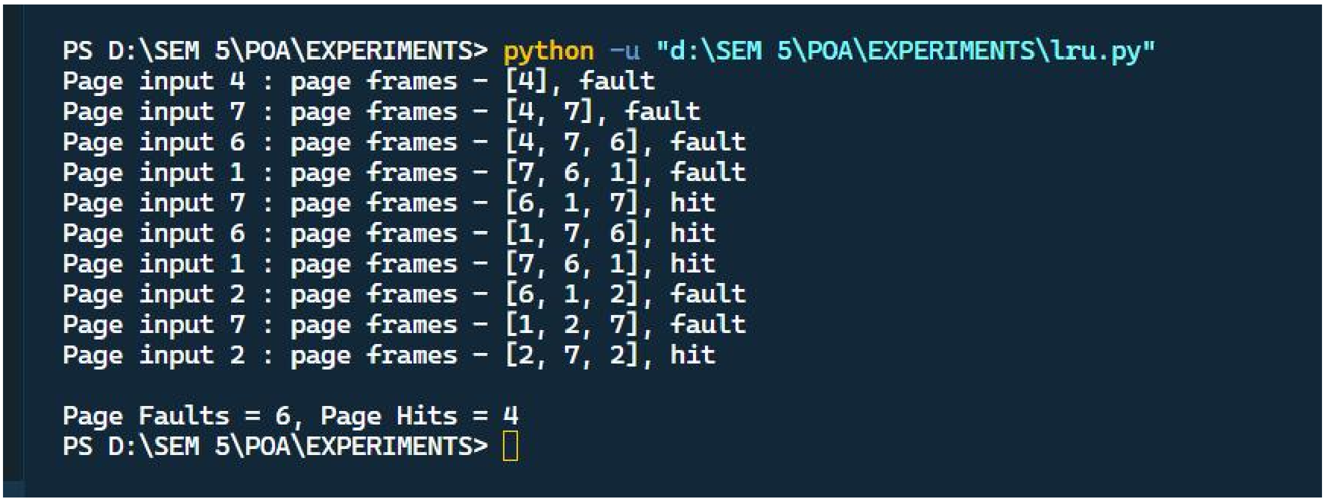
```
PS D:\SEM 5\POA\EXPERIMENTS> python -u "d:\SEM 5\POA\EXPERIMENTS\fifo.py"
Page input 4 : page frames - [4], fault
Page input 7 : page frames - [4, 7], fault
Page input 6 : page frames - [4, 7, 6], fault
Page input 1 : page frames - [7, 6, 1], fault
Page input 7 : page frames - [7, 6, 1], hit
Page input 6 : page frames - [7, 6, 1], hit
Page input 1 : page frames - [7, 6, 1], hit
Page input 2 : page frames - [6, 1, 2], fault
Page input 7 : page frames - [1, 2, 7], fault
Page input 2 : page frames - [1, 2, 7], hit

Page Faults = 6, Page Hits = 4
PS D:\SEM 5\POA\EXPERIMENTS> 
```

LRU:



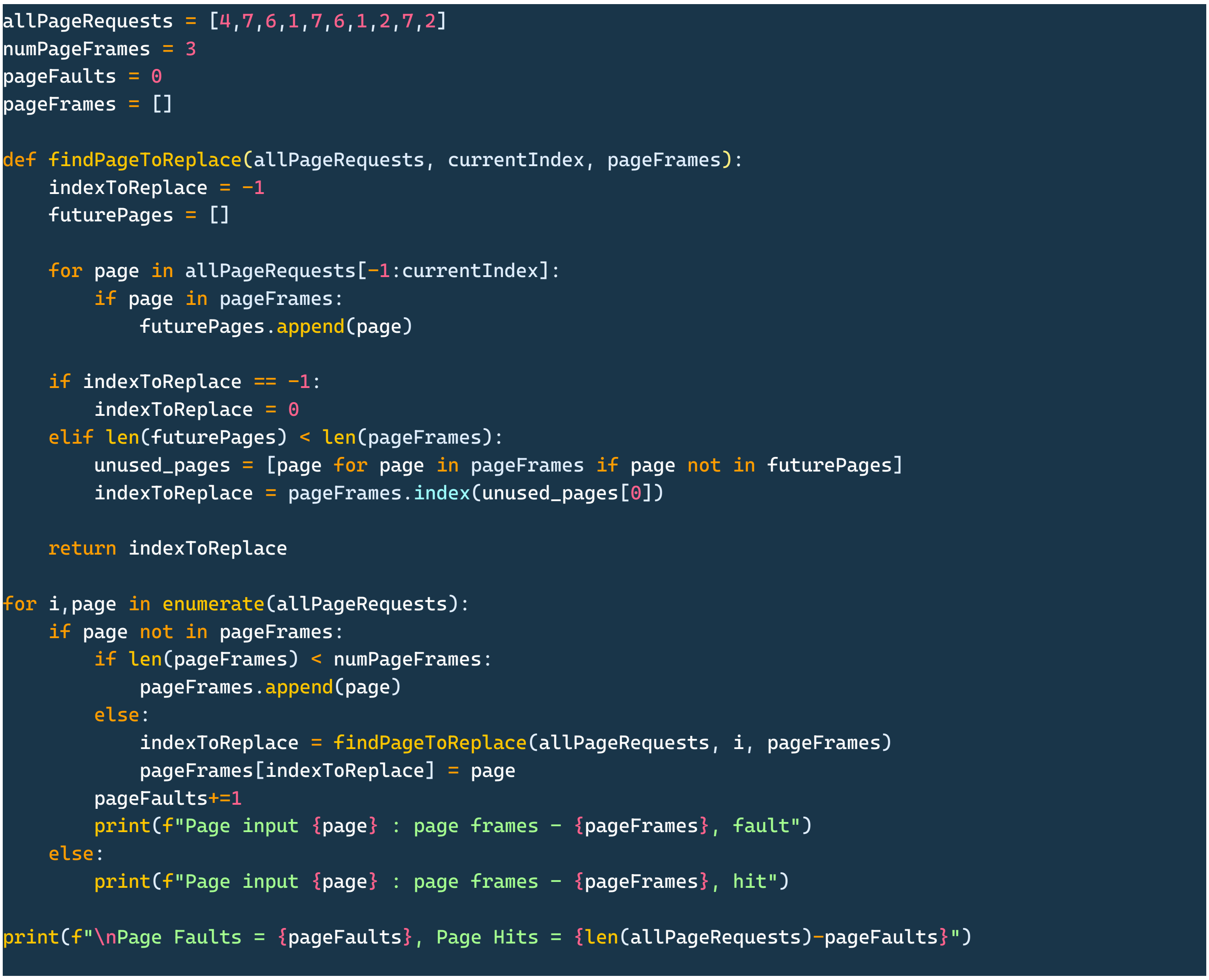
```python
allPageRequests = [4,7,6,1,7,6,1,2,7,2]
numPageFrames = 3
pageFaults = 0
pageFrames = []

for page in allPageRequests:
    if page not in pageFrames:
        if len(pageFrames) < numPageFrames:
            pageFrames.append(page)
        else:
            pageFrames = pageFrames[1:] + [page]
        pageFaults+=1
        print(f"Page input {page} : page frames - {pageFrames}, fault")
    else:
        pageFrames = pageFrames[1:] + [page]
        print(f"Page input {page} : page frames - {pageFrames}, hit")

print(f"\nPage Faults = {pageFaults}, Page Hits = {len(allPageRequests)-pageFaults}")
```

Output :



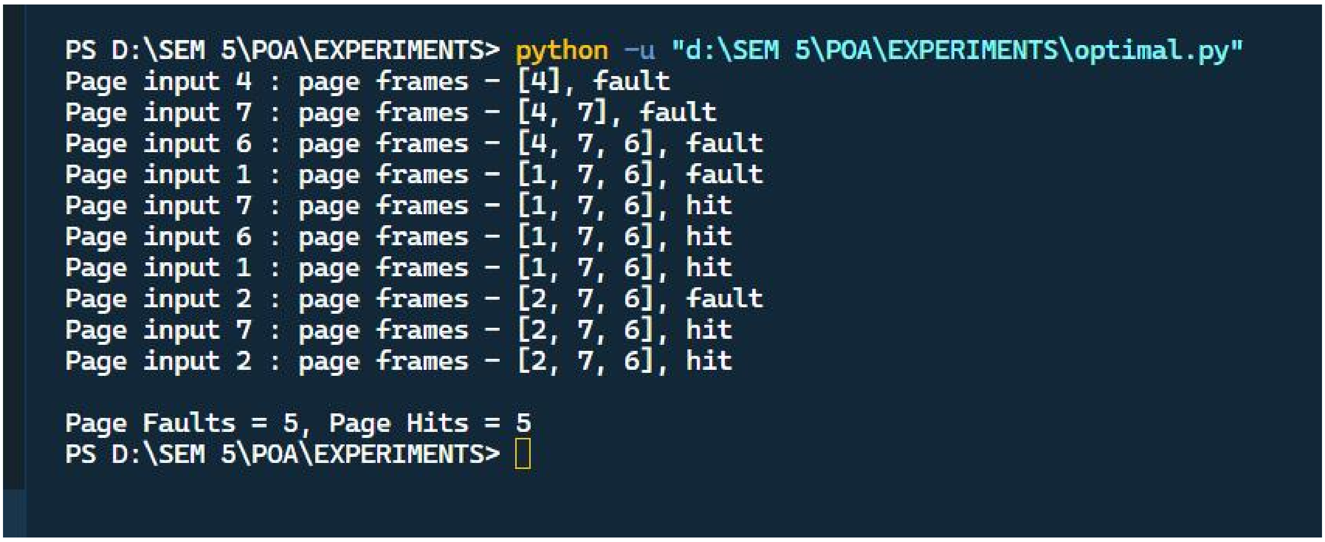
```
PS D:\SEM 5\POA\EXPERIMENTS> python -u "d:\SEM 5\POA\EXPERIMENTS\lru.py"
Page input 4 : page frames - [4], fault
Page input 7 : page frames - [4, 7], fault
Page input 6 : page frames - [4, 7, 6], fault
Page input 1 : page frames - [7, 6, 1], fault
Page input 7 : page frames - [6, 1, 7], hit
Page input 6 : page frames - [1, 7, 6], hit
Page input 1 : page frames - [7, 6, 1], hit
Page input 2 : page frames - [6, 1, 2], fault
Page input 7 : page frames - [1, 2, 7], fault
Page input 2 : page frames - [2, 7, 2], hit

Page Faults = 6, Page Hits = 4
PS D:\SEM 5\POA\EXPERIMENTS>
```

OPTIMAL :



```python
allPageRequests = [4,7,6,1,7,6,1,2,7,2]
numPageFrames = 3
pageFaults = 0
pageFrames = []

def findPageToReplace(allPageRequests, currentIndex, pageFrames):
    indexToReplace = -1
    futurePages = []

    for page in allPageRequests[-1:currentIndex]:
        if page in pageFrames:
            futurePages.append(page)

    if indexToReplace == -1:
        indexToReplace = 0
    elif len(futurePages) < len(pageFrames):
        unused_pages = [page for page in pageFrames if page not in futurePages]
        indexToReplace = pageFrames.index(unused_pages[0])

    return indexToReplace

for i,page in enumerate(allPageRequests):
    if page not in pageFrames:
        if len(pageFrames) < numPageFrames:
            pageFrames.append(page)
        else:
            indexToReplace = findPageToReplace(allPageRequests, i, pageFrames)
            pageFrames[indexToReplace] = page
        pageFaults+=1
        print(f"Page input {page} : page frames - {pageFrames}, fault")
    else:
        print(f"Page input {page} : page frames - {pageFrames}, hit")

print(f"\nPage Faults = {pageFaults}, Page Hits = {len(allPageRequests)-pageFaults}")
```

Output :



```
PS D:\SEM 5\POA\EXPERIMENTS> python -u "d:\SEM 5\POA\EXPERIMENTS\optimal.py"
Page input 4 : page frames - [4], fault
Page input 7 : page frames - [4, 7], fault
Page input 6 : page frames - [4, 7, 6], fault
Page input 1 : page frames - [1, 7, 6], fault
Page input 7 : page frames - [1, 7, 6], hit
Page input 6 : page frames - [1, 7, 6], hit
Page input 1 : page frames - [1, 7, 6], hit
Page input 2 : page frames - [2, 7, 6], fault
Page input 7 : page frames - [2, 7, 6], hit
Page input 2 : page frames - [2, 7, 6], hit

Page Faults = 5, Page Hits = 5
PS D:\SEM 5\POA\EXPERIMENTS> 
```