DIVYANSHU MAKHARIA
221070036

ASSIGNMENT 8

AIM:

Implementation of page replacement algorithm (FIFO,LRU,LFU)

THEORY:

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in. Page replacement becomes necessary when a page fault occurs and there are no free page frames in memory.

Following are some of the algorithms:

1. **First In First Out (FIFO):** This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.
2. **Least Recently Used:** In this algorithm, page which is least recently used will be replaced.Maintains a record of the order in which pages are accessed.
3. **Least Frequently Used:**In this algorithm,record of count of each page access is maintained.The page which is least frequently used is replaced.

DIVYANSHU MAKHARIA
221070036

CODE:

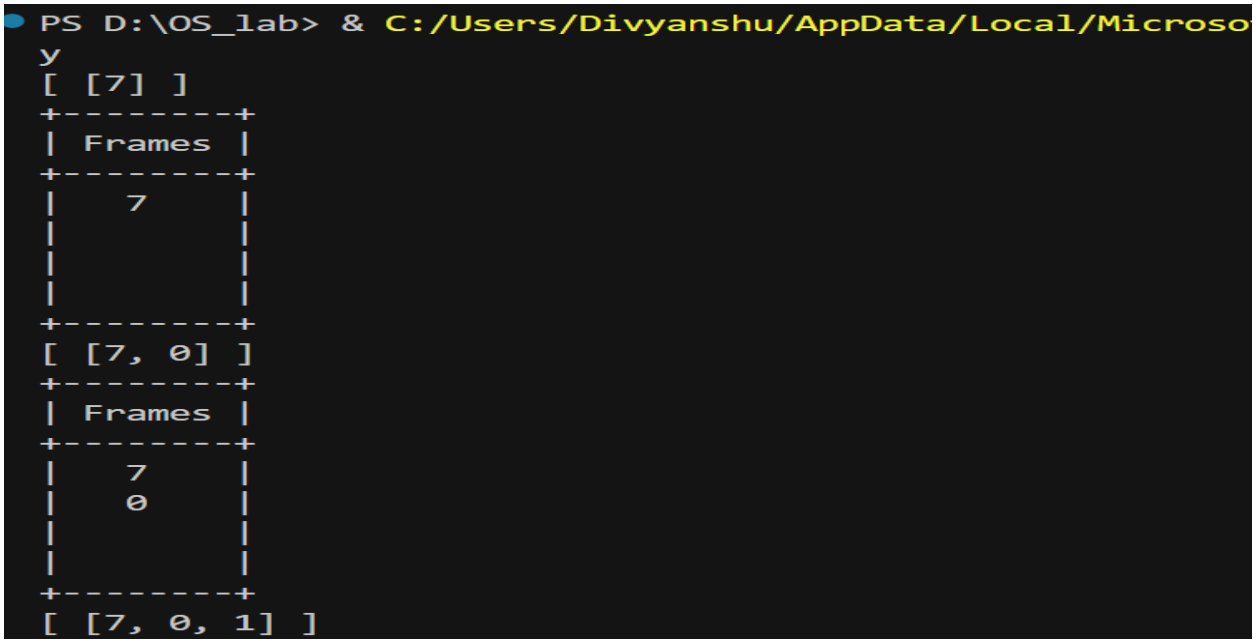
```
from collections import deque
from collections import defaultdict
from prettytable import PrettyTable
table = PrettyTable()
table.field_names=["Frames"]


def fifo(page_reference_string, num_frames):
    page_faults = 0
    frames = []
    hitmiss=[]
    for page in page_reference_string:
        table.clear_rows()
        if page not in frames:
            page_faults += 1
            if len(frames) < num_frames:
                frames.append(page)
            else:
                frames.pop(0)
                frames.append(page)
            hitmiss.append('Miss')
        else:
          hitmiss.append('Hit')
        print('[',frames,']')
        for i in frames:
            table.add_row([str(i)])
        if len(frames)<num_frames:
              for i in range(num_frames-len(frames)):
                  table.add_row([" "])
        print(table)
    print(hitmiss)
    return page_faults

def lru(page_reference_string, num_frames):

    page_faults = 0
    frames = deque(maxlen=num_frames)
    hitmiss=[]
```

```
    for page in page_reference_string:
        table.clear_rows()
        if page not in frames:
            page_faults += 1
            if len(frames) == num_frames:
                frames.popleft()
            frames.append(page)
            hitmiss.append('Miss')
        else:
            frames.remove(page)
            frames.append(page)
            hitmiss.append('Hit')
        frame1=list(frames)
        print('[',frame1,']')
        for i in frame1:
            table.add_row([str(i)])
        if len(frame1)<num_frames:
              for i in range(num_frames-len(frame1)):
                  table.add_row([" "])
        print(table)

    print(hitmiss)
    return page_faults

def lfu(page_reference_string, num_frames):
    page_faults = 0
    frames = []
    hitmiss=[]
    page_access_count = defaultdict(int)

    for page in page_reference_string:
        table.clear_rows()
        if page not in frames:
            page_faults += 1
            if len(frames) == num_frames:
                # Remove the least frequently used page
                least_used_page = min(frames, key=lambda x:
page_access_count[x])
                frames.remove(least_used_page)
            frames.append(page)
```

```
            hitmiss.append('Miss')
        else:
            hitmiss.append('Hit')
        page_access_count[page]+=1
        print('[',frames,']')
        for i in frames:
            table.add_row([str(i)])
        if len(frames)<num_frames:
              for i in range(num_frames-len(frames)):
                  table.add_row([" "])
        print(table)
    print(hitmiss)
    return page_faults

def main():
    with open('input.txt', 'r') as file:
        page_reference_string = list(map(int,
file.readline().strip().split()))
        num_frames = int(file.readline().strip())

    page_faults = fifo(page_reference_string, num_frames)
    print("Page faults using FIFO algorithm:", page_faults)

    page_faults = lru(page_reference_string, num_frames)
    print("Page faults using LRU algorithm:", page_faults)

    page_faults = lfu(page_reference_string, num_frames)
    print("Page faults using LFU algorithm:", page_faults)

if __name__ == "__main__":
    main()
```

DIVYANSHU MAKHARIA
221070036

OUTPUT:



```
+--------+
| Frames |
+--------+
|    7   |
|    0   |
|    1   |
|        |
+--------+
[ [7, 0, 1, 2] ]
+--------+
| Frames |
+--------+
|    7   |
|    0   |
|    1   |
|    2   |
+--------+
[ [7, 0, 1, 2] ]
+--------+
| Frames |
+--------+
|    7   |
|    0   |
|    1   |
|    2   |
+--------+
```

```
[ [0, 1, 2, 3] ]
+--------+
| Frames |
+--------+
|    0   |
|    1   |
|    2   |
|    3   |
+--------+
[ [0, 1, 2, 3] ]
+--------+
| Frames |
+--------+
|    0   |
|    1   |
|    2   |
|    3   |
+--------+
[ [1, 2, 3, 4] ]
```

DIVYANSHU MAKHARIA
221070036

```
| Frames |
+--------+
|    1   |
|    2   |
|    3   |
|    4   |
+--------+
[ [1, 2, 3, 4] ]
+--------+
| Frames |
+--------+
|    1   |
|    2   |
|    3   |
|    4   |
+--------+
[ [1, 2, 3, 4] ]
+--------+
| Frames |
+--------+
|    1   |
|    2   |
|    3   |
|    4   |
+--------+
```

```
[ [2, 3, 4, 0] ]
+--------+
| Frames |
+--------+
|   2    |
|   3    |
|   4    |
|   0    |
+--------+
['Miss', 'Miss', 'Miss', 'Miss', 'Hit', 'Miss', 'Hit', 'Miss', 'Hit', 'Hit', 'Miss', 'Hit', 'Hit', 'Hi
t']
Page faults using FIFO algorithm: 7
```

```
[ [4, 0, 2, 3] ]
+--------+
| Frames |
+--------+
|   4    |
|   0    |
|   2    |
|   3    |
+--------+
['Miss', 'Miss', 'Miss', 'Miss', 'Hit', 'Miss', 'Hit', 'Miss', 'Hit', 'Hit', 'Hit', 'Hit', 'Hit', 'Hit
']
Page faults using LRU algorithm: 6
```

```
[ [0, 2, 3, 4] ]
+--------+
| Frames |
+--------+
|   0    |
|   2    |
|   3    |
|   4    |
+--------+
['Miss', 'Miss', 'Miss', 'Miss', 'Hit', 'Miss', 'Hit', 'Miss', 'Hit', 'Hit', 'Hit', 'Hit', 'Hit', 'Hit
']
Page faults using LFU algorithm: 6
```

CONCLUSION:

Thus, we have successfully implemented page replacement algorithms.