Aim: Demonstrate the following page Replacement algorithms

Description: In 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.

(a) FIFO (b) LRU

Program:

(a) fifo:

```
n=int(input("Enter the no of frames: "))
pages=list(map(int,input("Enter page string: ").split(" ")))
page_faults=0
f=0
frame=[" " for _ in range(n)]
for i in pages:
    if i not in frame:
        frame[f]=i
        page_faults+=1
        f+=1
        if f==n:
            f=0
    print(frame)
print("No of page faults:",page_faults)
```

(b) CRO:

```
n=int(input("Enter the no of frames : "))
pages=list(map(int,input("Enter the page string : ").split(" ")))
page_faults=0
f=0
frame=[" " for _ in range(n)]
miss=set()
for i in range(len(pages)):
    if pages[i] not in frame:
        if i>=n and page_faults>=n:
            c=0
            miss.clear()
            for j in range(i,0,-1):
                if (pages[j-1] not in miss) and pages
                [j]!=pages[j-1]:
                    c+=1
                    miss.add(pages[j-1])
            if c==n:
                frame[frame.index(pages[j-1])]=pages[i]
            break
        else:
            frame[f]=pages[i]
            f+=1
        page_faults+=1
print(frame)
```

(3) CFU:

```
from collections import defaultdict

class CFUCache:
    def __init__(self, capacity):
        self.capacity = capacity
        self.page_freq = defaultdict(int)
        self.page_data = {}
        self.freq_pages = defaultdict(list)
        self.min_freq = 0

    def _update_frequency(self, page):
        frequency = self.page_freq[page]
        self.page_freq[page] = frequency + 1
        self.freq_pages[frequency].remove(page)
        if frequency == self.min_freq and not self.freq_pages
                                                [frequency]:
            self.min_freq += 1
        self.freq_pages[frequency + 1].append(page)

    def get(self, page):
        if page in self.page_data:
            self._update_frequency(page)
            return self.page_data[page]
        return -1
```

```
def put(self, page, data):
    if self.capacity == 0:
        return
    if page in self.page_data:
        self.page_data[page] = data
    else:
        if len(self.page_data) >= self.capacity:
            evict_page = self.freq_page
            del self.page_data[evict_page]
            del self.page_freq[evict_page]
        self.page_data[page] = data
        self.page_freq[page] = 1
        self.page_pages[1].append(page)
        self.min_freq = 1
if __name__ == "__main__":
    cache = LFUCache(3)
    cache.put(1, "Data 1")
    cache.put(2, "Data 2")
    cache.put(3, "Data 3")
    print(cache.get(1))
    print(cache.get(2))
```