Assignment:

Problem: Finding No of Page Faults using FIFO, Optimal and LRU Page replacement algorithms
Note: Quick Answer is provided in the beginning. Detailed Explanation for every algorithm with frame size is provided. Programs written for all the algorithms are pasted in the last.

Reference String : 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6
Quick Answer:

No of Frames	FIFO	Optimal	LRU
3	16	11	15
4	14	8	10

Detailed Explanation:
No of Frames : 3
FIFO : 16 Page Faults

Input	FIFO Queue	Page Fault	Explanation
1	1 * *	1	Queue has space
2	1 2 *	2	Queue has space
3	1 2 3	3	Queue has space
4	2 3 4	4	Remove 1 as 1 is the First IN
2	2 3 4	4	2 is already present
1	3 4 1	5	Remove 2
5	4 1 5	6	Remove 3
6	1 5 6	7	Remove 4
2	5 6 2	8	Remove 1
1	6 2 1	9	Remove 5
2	6 2 1	9	2 is already present
3	2 1 3	10	Remove 6
7	1 3 7	11	Remove 2
6	3 7 6	12	Remove 1
3	3 7 6	12	3 already present
2	7 6 2	13	3 is removed
1	6 2 1	14	7 is removed
2	6 2 1	14	1 is already present
3	2 1 3	15	6 is removed
6	1 3 6	16	2 is removed

Optimal: 11 Page Faults

Input	Queue	Page Fault	Explanation
1	1 * *	1	Queue has space
2	1 2 *	2	Queue has space
3	1 2 3	3	Queue has space
4	1 2 4	4	2,1 occurs before 3. Hence 3 is removed
2	1 2 4	4	2 already present
1	1 2 4	4	1 already present
5	1 2 5	5	1,2 occurs before 4. Hence 4 is removed
6	1 2 6	6	1,2 occurs before 5. Hence 5 is removed
2	1 2 6	6	2 already present
1	1 2 6	6	1 already present
2	1 2 6	6	2 already present
3	3 2 6	7	2,6 occurs before 1. Hence 1 is removed
7	3 7 6	8	3,6 occurs before 7. Hence 7 is removed
6	3 7 6	8	6 is already present
3	3 7 6	8	3 is already present
2	3 2 6	9	3,6 occurs before 7. Hence 7 is removed
1	3 2 1	10	3,2 occurs before 6. Hence 6 is removed
2	3 2 1	10	2 already present
3	3 2 1	10	3 already present
6	6 2 1	11	References ends. So it does not matter which is one is removed

LRU: 15 Page Faults

Input	List	Page Fault	Explanation
1	1 * *	1	Has Space
2	1 2 3	2	Has Space
3	1 2 3	3	Has Space
4	4 2 3	4	1 needs to be removed as it is the least recently used
2	4 2 3	4	2 already present
1	4 2 1	5	3 needs to be removed as that is LRU
5	2 1 5	6	4 is LRU
6	1 5 6	7	2 is LRU
2	2 5 6	8	1 is LRU
1	2 1 6	9	5 is LRU
2	2 1 6	9	2 already present
3	2 1 3	10	6 is LRU
7	2 7 3	11	1 is LRU
6	6 7 3	12	2 is LRU
3	6 7 3	12	3 already present
2	6 2 3	13	7 is LRU
1	1 2 3	14	6 is LRU
2	1 2 3	14	2 is already present
3	1 2 3	14	3 is already present
6	6 2 3	15	1 is LRU

No of Frames: 4
FIFO: 14 Page Faults

Input	FIFO Queue	Page Fault	Explanation
1	1 * * *	1	Queue has space
2	1 2 * *	2	Queue has space
3	1 2 3 *	3	Queue has space
4	1 2 3 4	4	Queue has space
2	1 2 3 4	4	2 is already present
1	1 2 3 4	4	1 is already present
5	2 3 4 5	5	1 is removed as FIFO
6	3 4 5 6	6	2 is removed as FIFO
2	4 5 6 2	7	3 is removed as FIFO
1	5 6 2 1	8	4 is removed as FIFO
2	5 6 2 1	8	2 is already present
3	6 2 1 3	9	5 is removed as FIFO
7	2 1 3 7	10	6 is removed as FIFO
6	1 3 7 6	11	2 is removed as FIFO
3	1 3 7 6	11	3 is already present
2	3 7 6 2	12	1 is removed as FIFO
1	7 6 2 1	13	3 is removed as FIFO
2	7 6 2 1	13	2 is already present
3	6 2 1 3	14	7 is removed as FIFO
6	6 2 1 3	14	6 is already present

Optimal: 8 Page Faults

Input	FIFO Queue	Page Fault	Explanation
1	1 * * *	1	Queue has space
2	1 2 * *	2	Queue has space
3	1 2 3 *	3	Queue has space
4	1 2 3 4	4	Queue has space
2	1 2 3 4	4	2 is already present
1	1 2 3 4	4	1 is already present
5	1 2 3 5	5	4 is far away compared to 1,2,3
6	1 2 3 6	6	5 is far away compared to 1,2,3
2	1,2,3,6	6	2 is already present
1	1,2,3,6	6	2 is already present
2	1,2,3,6	6	2 is already present
3	1,2,3,6	6	3 is already present
7	7,2,3,6	7	1 is far away compared to 2,3,6
6	7,2,3,6	7	6 is already present
3	7,2,3,6	7	3 is already present
2	7,2,3,6	7	2 is already present
1	1,2,3,6	8	7 is far away compared to 2,3,6
2	1,2,3,6	8	2 is already present
3	1,2,3,6	8	3 is already present
6	1,2,3,6	8	6 is already present

LRU : 10 Page Faults

Input	FIFO Queue	Page Fault	Explanation
1	1 * * *	1	Queue has space
2	1 2 * *	2	Queue has space
3	1 2 3 *	3	Queue has space
4	1 2 3 4	4	Queue has space
2	1 2 3 4	4	2 is already present
1	1 2 3 4	4	1 is already present
5	1 2 5 4	5	3 is LRU
6	1 2 5 6	6	4 is LRU
2	1 2 5 6	6	2 is already present
1	1 2 5 6	6	1 is already present
2	1 2 5 6	6	2 is already present
3	1 2 3 6	7	5 is LRU
7	1 2 3 7	8	6 is LRU
6	2 3 7 6	9	1 is LRU
3	2 3 7 6	9	3 is already present
2	2 3 7 6	9	2 is already present
1	2 3 1 6	10	7 is LRU
2	2 3 1 6	10	2 is already present
3	2 3 1 6	10	3 is already present
6	2 3 1 6	10	6 is already present

FIFO Program:

```
def is_there(l,f):
    for e in l:
        if e is f :
            return True
    return False


def shift_add(e,l,fs):
    for i in range(0,fs-1):
        l[i] = l[i+1]
    l[fs-1] = e


def find(input,fs,debug):
    s = map(int,input.split(","))
    pf = 0
    q = [None] * fs
    for i in range(0, fs):
        q[i] = s[i]
        pf = pf + 1

    for i in range(fs, len(s)):
        if debug:
            print "-----------------"
            print "Accessing ", s[i]
            print q
        if is_there(q, s[i]) is False:
            shift_add(s[i], q, fs)
            pf = pf + 1
        if debug:
            print q
            print "PF ", pf
    print "Final Page Faults ",pf
    print "**************************************************"


#find("7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1",3,False)
#find("1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6",3,True)
find("1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6",4,True)
```

Optimal Program:

```
def is_there(l,f):
    for e in l:
        if e is f :
            return True
    return False

def find_pos(e,pos,s):
    for i in range(pos,len(s)):
        if e == s[i]:
            return i
    return 20000


def find_max_pos(pos_array):
    max = pos_array[0]
    max_pos = 0
    for i in range(1, len(pos_array)):
        if max < pos_array[i]:
            max = pos_array[i]
            max_pos = i
    return max_pos


def find_which_one_to_replace(s,pos,q):
    pos_array = [None] * len(q)
    for i in range(0,len(q)):
        pos_array[i] = find_pos(q[i],pos+1,s)
    return q[find_max_pos(pos_array)]
```

```
def replace(q,e,ne):
    for i in range(0,len(q)):
        if q[i] == e:
            q[i] = ne
            break
    return q


def find(input,fs,debug):
    s = map(int,input.split(","))
    pf = 0
    q = [None] * fs
    for i in range(0, fs):
        q[i] = s[i]
        pf = pf + 1
    pos = fs

    for i in range(fs, len(s)):
        if is_there(q, s[i]) is False:
            e = find_which_one_to_replace(s,pos,q)
            q = replace(q,e,s[i])
            print q
            pf = pf + 1
        pos = pos + 1
    print "Final Page Faults ",pf
    print "***********************************************"

#find("7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1",3,False)
#find("1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6",3,True)
find("1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6",4,True)
```

LRU Program:

```
def is_there(l,f):
    for e in l:
        if e[0] is f :
            return True
    return False



def find_lru(q):
    min_lru = q[0][0]
    min_lru_time =  q[0][1]
    min_lru_pos = 0
    for i in range(1,len(q)):
        if q[i][1] < min_lru_time:
            min_lru = q[i][0]
            min_lru_time = q[i][1]
            min_lru_pos = i
    return min_lru_pos

def find_pos(q,e):
    for i in range(0,len(q)):
        if e is q[i][0]:
            return i

def find(input,fs,debug):
    s = map(int,input.split(","))
    pf = 0
    q = [None] * fs
    for i in range(0, fs):
        q[i] = (s[i] , i)
        pf = pf + 1
    for i in range(fs, len(s)):
        if is_there(q,s[i]) is False:
            pf = pf + 1
            min_lru_pos = find_lru(q)
            q[min_lru_pos] = (s[i],i)
        else:
            pos = find_pos(q,s[i])
            q[pos] = (s[i],i)
    print "Final Page Faults ",pf
    print "*************************************************"


find("1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6",4,True)
find("1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6",3,True)
```