1. A process has nine pages numbered 0 – 8. It is given four frames (initially empty). Suppose that the process makes memory accesses using the following reference string.

7, 0, 8, 1, 3, 5, 7, 5, 8, 5, 2, 8, 4, 0, 4, 2, 5, 1, 0, 2, 6, 4, 1, 7, 2

Explain the working of the following page-replacement algorithms on this reference string.

(a) FIFO
(b) OPT
(c) LRU

2. Use the same reference string as in Exercise 1. The process is given four frames (initially empty).

7, 0, 8, 1, 3, 5, 7, 5, 8, 5, 2, 8, 4, 0, 4, 2, 5, 1, 0, 2, 6, 4, 1, 7, 2

Assume that the clock algorithm (second-chance LRU page replacement) is used. Assume that immediately after a page access or a page replacement, the reference bit of the accessed page is set to 1. Show the working of the clock algorithm on the above reference string. Show, in a table, the page numbers and the reference bits of the pages loaded to memory.

Shown in the row-major fashion below.

7	7(1), –(0), –(0), –(0)	0	7(1), 0(1), –(0), –(0)	8	7(1), 0(1), 8(1), –(0)	1	**7**(1), 0(1), 8(1), 1(1)	3	3(1), **0**(0), 8(0), 1(0)
5	3(1), 5(1), **8**(0), 1(0)	7	3(1), 5(1), 7(1), **1**(0)	5	3(1), 5(1), 7(1), **1**(0)	8	**3**(1), 5(1), 7(1), 8(1)	5	**3**(1), 5(1), 7(1), 8(1)
2	2(1), **5**(0), 7(0), 8(0)	8	2(1), **5**(0), 7(0), 8(1)	4	2(1), 4(1), **7**(0), 8(1)	0	2(1), 4(1), 0(1), **8**(1)	4	2(1), 4(1), 0(1), **8**(1)
2	2(1), 4(1), 0(1), **8**(1)	5	**2**(0), 4(0), 0(0), 5(1)	1	1(1), **4**(0), 0(0), 5(1)	0	1(1), **4**(0), 0(1), 5(1)	2	1(1), 2(1), **0**(1), 5(1)
6	1(0), 2(0), 6(1), **5**(0)	4	**1**(0), 2(0), 6(1), 4(1)	1	**1**(1), 2(0), 6(1), 4(1)	7	1(1), 7(1), **6**(1), 4(1)	2	1(0), 7(0), 2(1), **4**(0)

3. The same process with the same reference string again as in the last two exercises.

7, 0, 8, 1, 3, 5, 7, 5, 8, 5, 2, 8, 4, 0, 4, 2, 5, 1, 0, 2, 6, 4, 1, 7, 2

We use a working set (initially empty) of window size six. Show, in a table, how the working set changes after each reference. Assume that at any time, the process is given only those frames that can accommodate the current working set. Show, in a table, how the frame allocation changes with time, and which references encounter page faults.

Page access	Last-access list	WS (frame allocation)	Page fault
7	7	{ 7 }	Yes
0	7 – 0	{ 0, 7 }	Yes
8	7 – 0 – 8	{ 0, 7, 8 }	Yes
1	7 – 0 – 8 – 1	{ 0, 1, 7, 8 }	Yes
3	7 – 0 – 8 – 1 – 3	{ 0, 1, 3, 7, 8 }	Yes
5	7 – 0 – 8 – 1 – 3 – 5	{ 0, 1, 3, 5, 7, 8 }	Yes
7	0 – 8 – 1 – 3 – 5 – 7	{ 0, 1, 3, 5, 7, 8 }	No
5	8 – 1 – 3 – 5 – 7 – 5	{ 1, 3, 5, 7, 8 }	No
8	1 – 3 – 5 – 7 – 5 – 8	{ 1, 3, 5, 7, 8 }	No
5	3 – 5 – 7 – 5 – 8 – 5	{ 3, 5, 7, 8 }	No
2	5 – 7 – 5 – 8 – 5 – 2	{ 2, 5, 7, 8 }	Yes
8	7 – 5 – 8 – 5 – 2 – 8	{ 2, 5, 7, 8 }	No
4	5 – 8 – 5 – 2 – 8 – 4	{ 2, 4, 5, 8 }	Yes
0	8 – 5 – 2 – 8 – 4 – 0	{ 0, 2, 4, 5, 8 }	Yes
4	5 – 2 – 8 – 4 – 0 – 4	{ 0, 2, 4, 5, 8 }	No
2	2 – 8 – 4 – 0 – 4 – 2	{ 0, 2, 4, 8 }	No
5	8 – 4 – 0 – 4 – 2 – 5	{ 0, 2, 4, 5, 8 }	Yes
1	4 – 0 – 4 – 2 – 5 – 1	{ 0, 1, 2, 4, 5 }	Yes
0	0 – 4 – 2 – 5 – 1 – 0	{ 0, 1, 2, 4, 5 }	No
2	4 – 2 – 5 – 1 – 0 – 2	{ 0, 1, 2, 4, 5 }	No
6	2 – 5 – 1 – 0 – 2 – 6	{ 0, 1, 2, 5, 6 }	Yes
4	5 – 1 – 0 – 2 – 6 – 4	{ 0, 1, 2, 4, 5, 6 }	Yes
1	1 – 0 – 2 – 6 – 4 – 1	{ 0, 1, 2, 4, 6 }	No
7	0 – 2 – 6 – 4 – 1 – 7	{ 0, 1, 2, 4, 6, 7 }	Yes
2	2 – 6 – 4 – 1 – 7 – 2	{ 1, 2, 4, 6, 7 }	No

4. Consider a virtual-memory system which implements Enhanced Second-Chance page-replacement algorithm, with one reference bit and one dirty bit per page. The number of page frames is 4 (all initially empty). All the dirty bits and reference bits are initially 0. A process P generates the following page-reference string.

0, 1, 3(W), 6, 2, 4, 5, 2(W), 5, 0(W), 3, 1(W)

In the string, an integer x indicates that the page x is accessed in the read-only mode, whereas x(W) indicates that the process P modifies the page x. Assume that the pointer (clock hand) is initially at Frame 0. The page-replacement algorithm replaces a page with (reference bit, dirty bit) equal to (0, 0) or (0, 1) only. If (0, 0) is found during one full rotation, the corresponding page is replaced. If not, yet another full rotation is made to find a (0, 0) entry for replacement. If that attempt fails too, a third rotation is carried out, and the first (0, 1) entry is chosen for replacement. After a page replacement, the pointer moves to the frame cyclically next to the replacement position.

For each page reference, show the contents of frames 0, 1, 2, 3 just after the reference, along with the reference-bit and dirty-bit values for the pages stored in all frames (so your answer should look like a 2-d table, with 4 columns (one for each frame) and 12 rows (one for each page in the reference string). Identify the page faults, and find the total number of page faults. Show the step-by-step procedure in the table.

Ref #	Page	F0	F1	F2	F3	Page fault?
1	0	0 (1, 0)	–	–	–	Yes
2	1	0 (1, 0)	1 (1, 0)	–	–	Yes
3	3	0 (1, 0)	1 (1, 0)	3 (1, 1)	–	Yes
4	6	0 (1, 0)	1 (1, 0)	3 (1, 1)	6 (1, 0)	Yes
5	2	2 (1, 0)	1 (0, 0)	3 (0, 1)	6 (0, 0)	Yes
6	4	2 (1, 0)	4 (1, 0)	3 (0, 1)	6 (0, 0)	Yes
7	5	2 (1, 0)	4 (1, 0)	3 (0, 1)	5 (1, 0)	Yes
8	2	2 (1, 1)	4 (1, 0)	3 (0, 1)	5 (1, 0)	No
9	5	2 (1, 1)	4 (1, 0)	3 (0, 1)	5 (1, 0)	No
10	0	2 (0, 1)	0 (1, 0)	3 (0, 1)	5 (0, 0)	Yes
11	3	2 (0, 1)	0 (1, 0)	3 (0, 1)	5 (0, 0)	No
12	1	2 (0, 1)	0 (1, 0)	3 (0, 1)	1 (0, 0)	Yes

5. An OS implements virtual memory using the LRU-approximation page-replacement algorithm based on reference bits. A small process is allocated four frames. Initiated by timer interrupts, the reference bits are checked at regular intervals. Initially, the reference bits are 0111 (page 0 is 0, the rest are 1). At the four subsequent timer interrupts, the values are 1011, 1010, 1101, 0010. If the page-replacement algorithm is used with 8-bit counters (all initialized to 0), show the contents (bit-wise) of the four counters after the last of the above timer interrupts. Assume that no page fault occurred during the above intervals.

	Timer INT 0	Timer INT 1	Timer INT 2	Timer INT 3	Timer INT 4
Reference bits	0111	1011	1010	1101	0010
Page 0	000000000	10000000	11000000	11100000	01110000
Page 1	100000000	01000000	00100000	10010000	01001000
Page 2	100000000	11000000	11100000	01110000	10111000
Page 3	100000000	11000000	01100000	10110000	01011000

6. Consider a 2-D array in the usual contiguous representation.

```
int A[1024][1024];
```

The system uses 4 KB pages/frames. The process is given three frames, of which the first frame is used to store the code. The remaining two frames are used to store the array A. Compute, for each of the following code snippets, how many page faults are generated. Use LRU page replacement.

(a)

```
for (i=0; i<1024; ++i)
   for (j=0; j<1024; ++j)
      A[i][j] = 0;
for (i=0; i<1024; ++i)
   A[i][i] = 1;
```

(b)

```
for (i=0; i<1024; ++i)
   for (j=0; j<1024; ++j)
      A[j][i] = 0;
for (i=0; i<1024; ++i)
   A[i][i] = 1;
```

(a) 1024 + 1024 = 2048
(b) 1024^2+ 1024 = 1049600