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	**7**(1), 0(1)	8	**7**(1), 0(1), 8(1)	1	**7**(1), 0(1), 8(1), 1(1)	3	**3**(1), 0(0), 8(0), 1(0)
5	3(0), **5**(1), 8(0), 1(0)	7	3(0), 5(0), **7**(1), 1(0)	5	3(0), 5(1), **7**(1), 1(0)	8	3(0), 5(1), 7(0), **8**(1)	5	3(0), 5(1), 7(0), **8**(1)
2	**2**(1), 5(1), 7(0), 8(0)	8	**2**(1), 5(1), 7(0), 8(1)	4	2(0), 5(0), **4**(1), 8(1)	0	**0**(1), 5(0), 4(0), 8(0)	4	**0**(1), 5(0), 4(1), 8(0)
2	0(0), **2**(1), 4(1), 8(0)	5	0(0), 2(0), 4(0), **5**(1)	1	**1**(1), 2(1), 4(1), 5(0)	0	1(0), 2(0), 4(0), **0**(1)	2	1(0), 2(1), 4(0), **0**(1)
6	**6**(1), 2(1), 4(0), 0(0)	4	**6**(1), 1(1), 4(1), 0(0)	1	**6**(1), 1(1), 4(1), 0(0)	7	6(0), 1(0), 4(0), **7**(1)	2	**2**(1), 1(0), 4(0), 7(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 of window size six. The working set is initially empty. 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 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