Utku NOYAN
60717

COMP 304- Operating Systems: Assignment 3

Due: 13 May 2020

Notes: This is an individual assignment. Bring the hard copy to the exam and submit a soft copy through blackboard. No late assignment will be accepted. This assignment is worth 10% of your total grade.

Problem 1

(12 points) Consider the memory management methods of contiguous allocation, paging, and segmentation. Compare these methods with respect to the following: external fragmentation and code sharing among processes.

Problem 2

(10 pts) Suppose a computer that supports virtual memory and has 32-bit virtual addresses and uses page size of 8KB. If a process uses 100 pages of its virtual address space, how much space does its page table require if a single-level page table is used? Assume each table entry occupies 4 bytes.

Problem 3

(18 points) Consider the following page reference string:
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7

How many page faults would occur for the page replacement algorithms: LRU, FIFO, Optimal in the following cases? Explain your answer. Remember all frames are initially empty, so your first unique pages will all cost one fault each.
a) 5 frames are allocated for the process
b) 6 frames are allocated for the process

Problem 4

(10 points) Consider a system where for each frame we keep track of how many pages are associated with that frame by using a counter. When we need to replace pages, our page replacement algorithm searches for a frame with the smallest number of pages associated with it.
Initially all the counters are set to zero for all frames. When a new page is associated a frame, its counter is incremented.

Problem 1

Contigous memory allocation:

- Suffers from external fragmentation as address spaces are allocated contiguously and holes develop as old processes die and new processes are started.
- It doesn't allow process to share code because virtual memory segment of processes isn't divide into non-contigous fine-grained segments.

Segmentation:

- Suffers from external fragmentation as a segment of process is laid out contiguously in physical memory and fragmentation would occur as segments of dead processes are replaced by segments of new processes.
- Enables processes to share code, such as two distinct processes could share code segment but have different date segments.

Paging:

- Does not suffer from external fragmentation.
- Enables processes to share code at granularity of pages.

Problem2:

Virtual Address: 2^{32} byte

Page size = 8 KB = 2^{13} byte

# of pages = $\frac{\text{Virtual Address}}{\text{Page size}} = 2^{32}/2^{13} = 2^{19}$ pages

In single level page table, # of entries = 2^{19}

100 pages using = $\lceil 2^7 \rceil = 128$

↳ to represent 100 bit we need 7 bits

→ We shouldn't use less than 2^{13}, so 2^{13} will be the page.

Size of Page table = # of page table entries * Size of page table entry

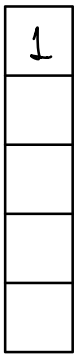
$= 2^{19} \times 4 = 2^{19} \times 4 / (1024 \times 1024)$ MB = 2 MB

Problem 3:

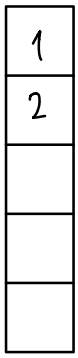
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7

a) 5 frames

LRU: Least recently used page will be replaced if it hasn't been used as recently as other pages.

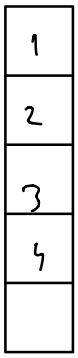


Income: 1
Fault

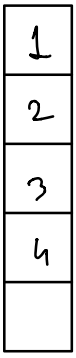


Income = 2
Fault

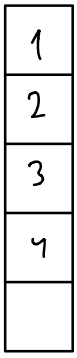
Income = 3
Fault



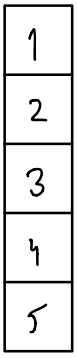
Income = 4
Fault



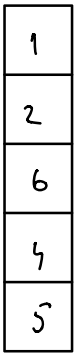
Income: 2
Hit



Income = 1
Hit

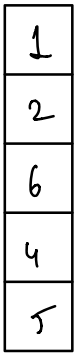


Income = 5
Fault



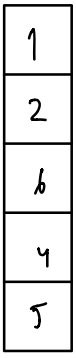
Income = 6
Fault

→ replaced with 3 since 1,2,7,4 recent hits.

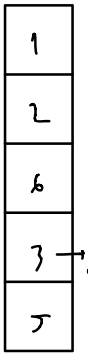


Income: 2
Hit

Income = 1
Hit

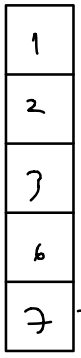


Income = 2
Hit



Income = 3
Fault

→ replaced with 3 since 1,7,4,5 recent hits

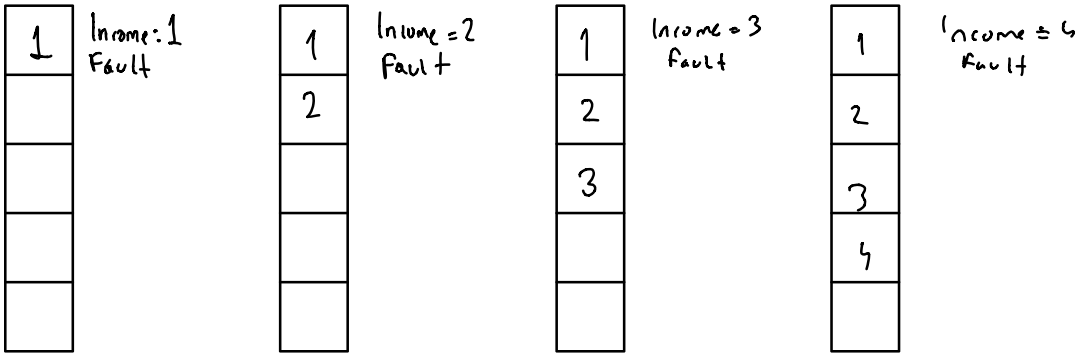


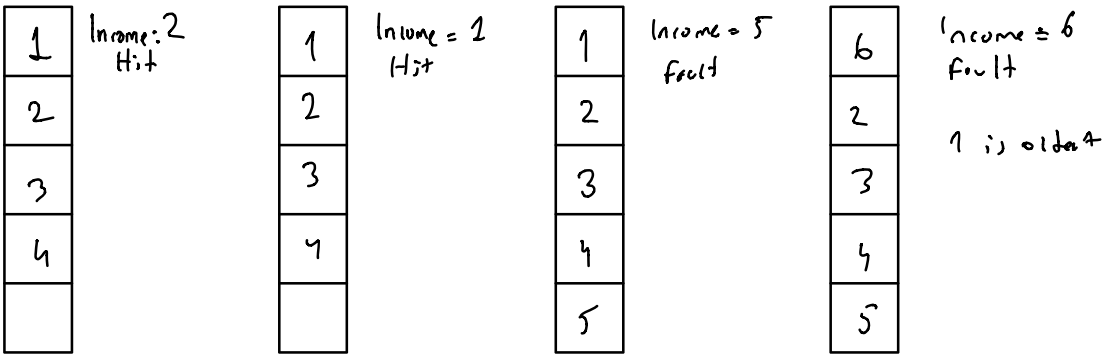
Income = 7
Fault

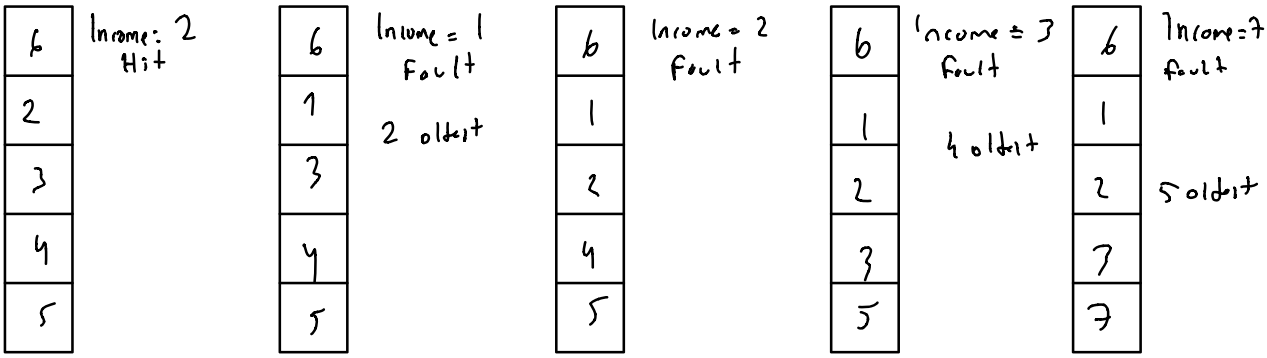
→ replaced with 5 since 1,7,2,4 recent hits.

→ Total fault is 8 for LRU in 5 frames

FIFO: First In First Out. oldest page replaced in case of fault.

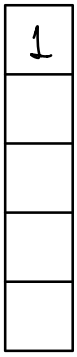




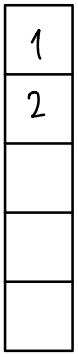


→ Total fault is 10 for FIFO in 5 frames

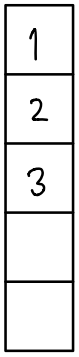
Optimal: Looks at future, replaces which is not going to be used futher



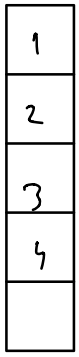
Income: 1
Fault



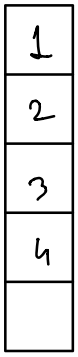
Income = 2
Fault



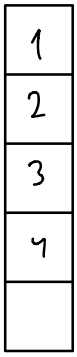
Income = 3
Fault



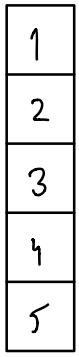
Income = 4
Fault



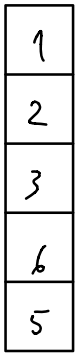
Income: 2
Hit



Income = 1
Hit



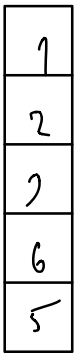
Income = 5
fault



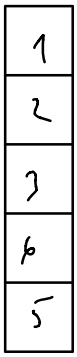
Income = 6
fault

don't see 4 in future

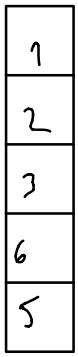
Income: 2
Hit



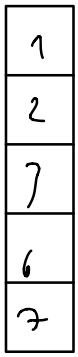
Income = 1
Hit



Income = 2



Income = 3
Hit



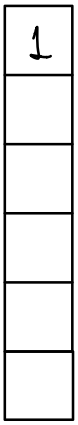
Income = 7
fault

↓ don't see 5 in future.

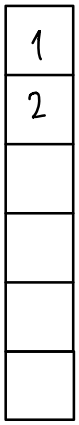
→ Total fault is 7 for optimal in 5 frames

b) 6 frames

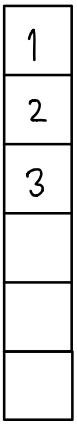
LRU: Least recently used page will be replaced if it hasn't been used as recently as other pages.



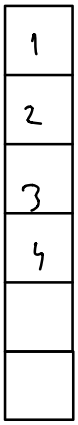
Income: 1
Fault



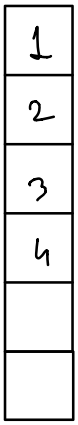
Income = 2
Fault



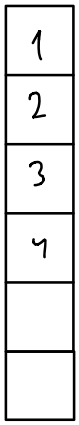
Income = 3
fault



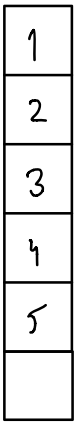
Income = 4
Fault



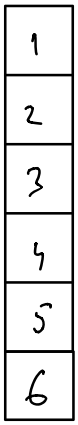
Income: 2
Hit



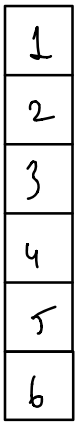
Income = 1
Hit



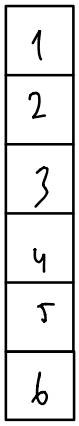
Income = 5
fault



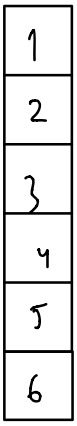
Income = 6
fault



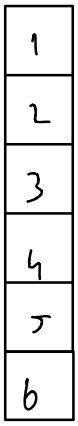
Income: 2
Hit



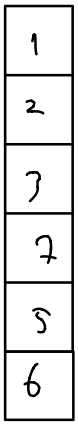
Income = 1
Hit



Income = 2
Hit



Income = 3
Hit

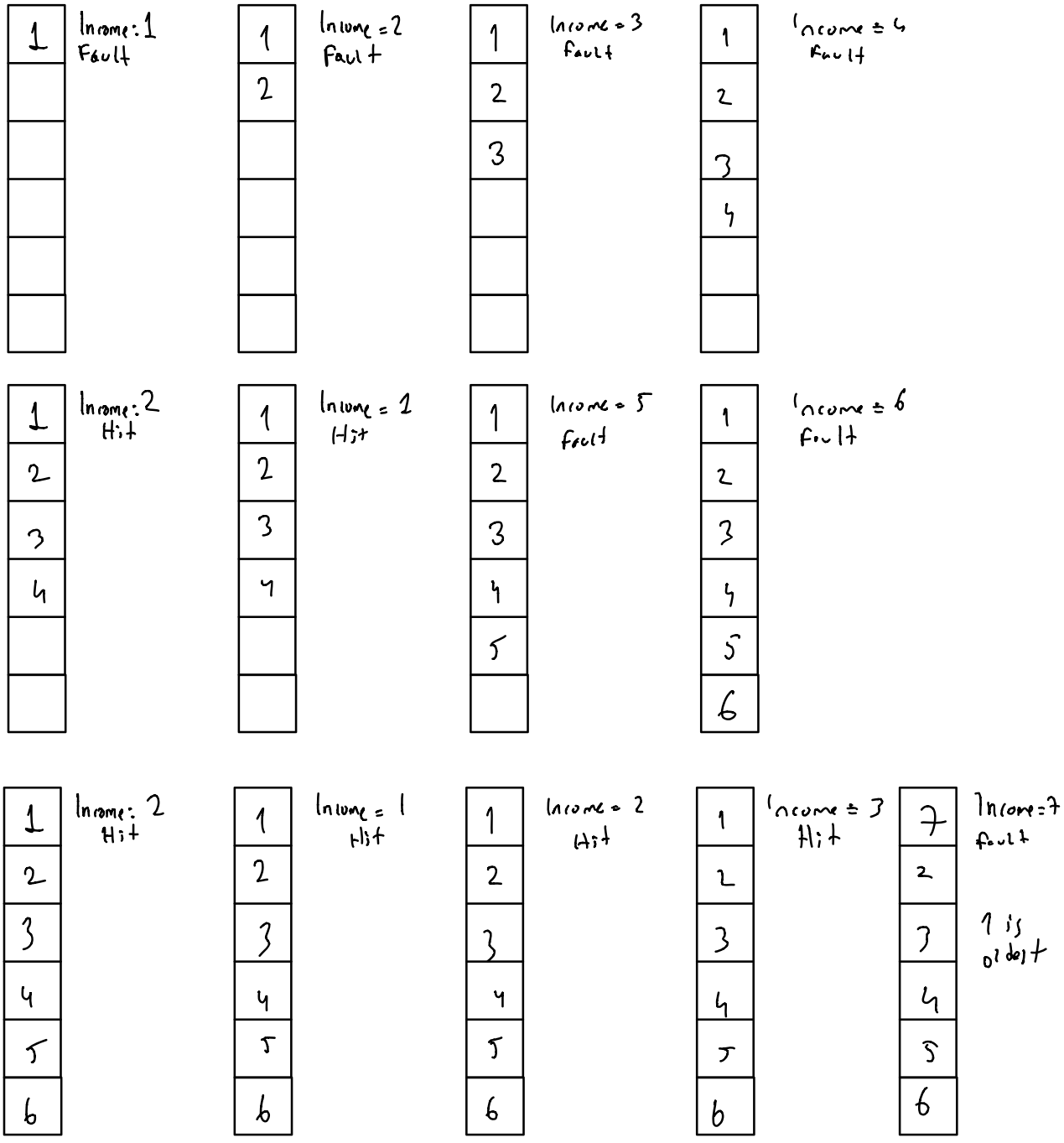


Income = 7
fault

+ replaced with 4 since 1,2,3,5 are recent hits.

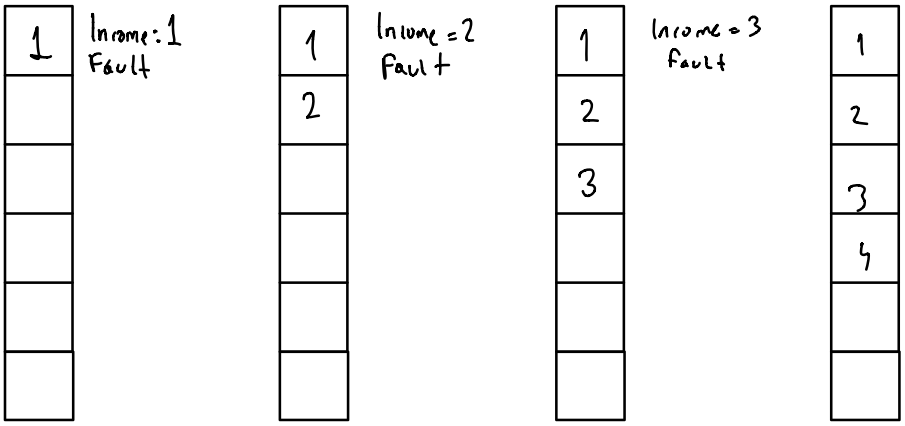
→ Total fault is 7 for LRU in 6 frames

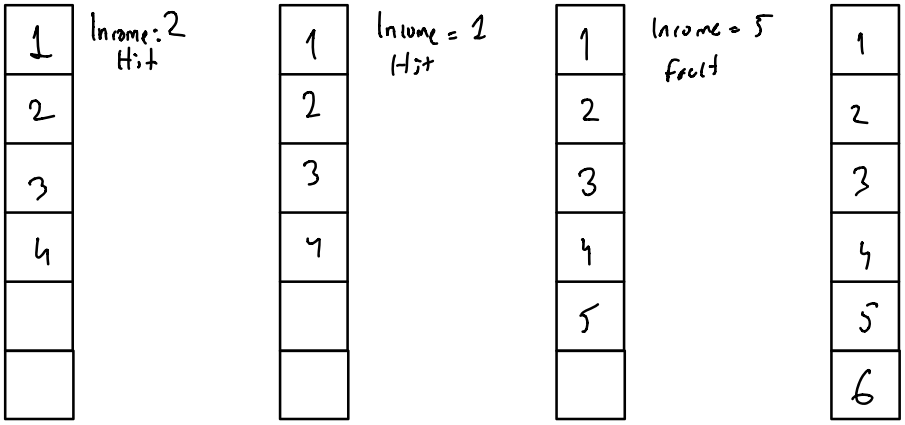
FIFO: First In First Out. Oldest page replaced in case of fault.

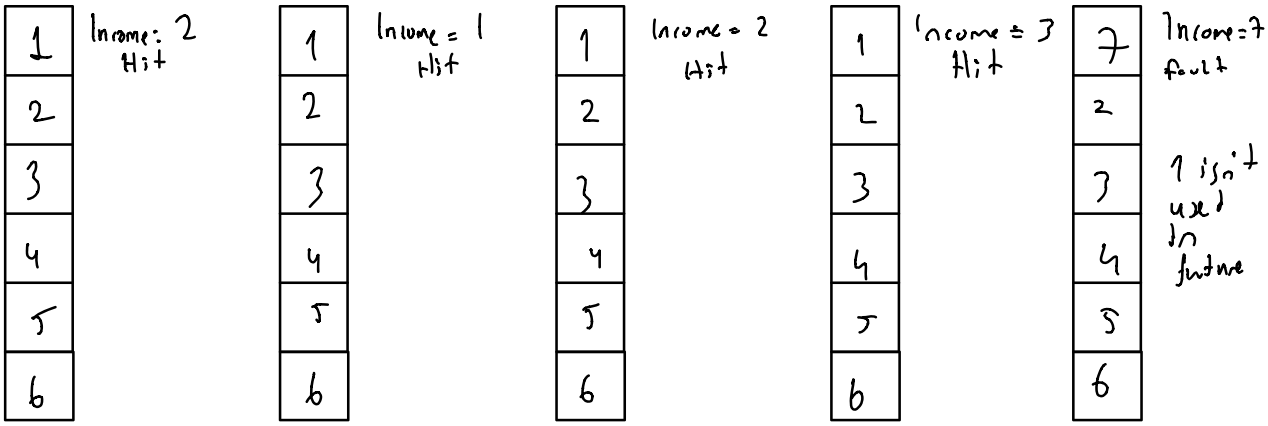


→ Total fault is 7 for FIFO in 6 frames

Optimal: Looks at future, replaces which is not going to be used futur







→ Total fault is 7 for optimn in 6 frames

Optimal has lowest fault rate since it looks future, but it is not applicable in real life.

Show how the number of page faults for this page replacement algorithm given the following reference string of a process with four frames in memory? Is this a good algorithm for page replacement?

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

Problem 5

(6+6 points)
Consider a file system developed by one of the KU graduates for Unix-based systems. In this file system, when a file is deleted, its disk space is reclaimed but links to that file can still exist.
a) What problems can arise if a new file is created with the same absolute path name?
b) How can you avoid the problem related to links? Explain your solution.

Problem 6

(6+5+5 points) Consider the working-set model with the working set window Δ being 7. Given the following system with 3 independent processes and their corresponding reference strings as shown below:

Time	0	1	2	3	4	5	6	7	8	9	10
P1	2	5	7	4	4	1	2	3	7	3	3
P2	1	1	1	3	3	4	4	3	5	6	7
P3	4	9	3	2	9	8	7	7	7	1	2

a) According to the given data above, what is the working set and working set size for each process at time 9 (inclusive)?
b) Does the system suffer from trashing at time 9 (inclusive) if there are total 10 frames in the physical memory? Why or why not? Show your work for credit.
c) How many frames should be in the physical memory so that the system doesn't suffer from trashing at time 7 (inclusive)? Show your work for credit.

Problem 7

(6+6 points) Assume page size is 1024 words and each row is stored in one page.
int A[][] = new int[1024][1024];

If the OS allocates 512 frames for a program and uses LRU page replacement algorithm

Problem 4

~~Least frequently used~~ replaces page that has smallest frequency, if all pages have same frequency, then apply FIFO.

	F	F	F	F	F	H	H	F	F	F	F	H	H	F	H	H	H	F	F	F	F	F
1	2	3	4	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2	
1	1	1	1	5	5	5	5	6	6	8	8	8	8	8	8	8	8	8	8	8	8	
	2	2	2	2	2	2	1	1	7	7	7	7	7	7	7	7	7	7	7	7	2	
		3	3	3	3	3	3	3	3	3	3	3	9	9	9	9	9	9	9	9		
			4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	5	4	2	

# of hit = 7

# of fault = 15

Frequency Table

1	1 0 1 0
2	1 0
3	1 2 1
4	1 2 1 0 1
5	1 0 1 0 1
6	1 0
7	1 2 3
8	1 2 3
9	1 2

a) It is not good algorithm since page faults are increased compose to other algorithms such as Optimal, LRU.

Problem 5

a) Suppose there exists two file such as: 1) fileA – old file
2) file B – new file

A user want to reach FileA through an existing link will eventually reach FileB. The access protection for fileA is used instead of the one that linked with FileB.

b) Problem can be solved by ensuring all links to deleted file are deleted also. We can handle this by maintaining a list of all links to a file, removing each of them when file is deleted. OR, maintain file reference list, deleting the file only after all links, or references to find that file have been deleted.

Problem 6

Time	0	1	2	3	4	5	6	7	8	9	10
P1	2	5	7	4	4	1	2	3	7	3	3
P2	1	1	1	3	3	4	4	3	5	6	7
P3	4	9	3	2	9	8	7	7	7	1	2

a) According to the given data above, what is the working set and working set size for each process at time 9 (inclusive)?
b) Does the system suffer from trashing at time 9 (inclusive) if there are total 10 frames in the physical memory? Why or why not? Show your work for credit.
c) How many frames should be in the physical memory so that the system doesn't suffer from trashing at time 7 (inclusive)? Show your work for credit.

a) working set 1 at virtual time 9 = {1, 2, 7, 4, 7} → set size = WSS1 = 5
working set 2 at virtual time 9 = {3, 4, 5, 6} → set size = WS12 = 4
working set 3 " " " " 9 = {1, 2, 7, 8, 9} → set size = WSS3 = 5

b) $D(t_9) = \sum_{i=1}^{3} WSS_i(t_9) = WSS_1 + WSS_2 + WSS_3 = 5 + 4 + 5 = 14$

Since D > n (14 > 10), system suffers from trashing.

c) working set 1 at virtual time = {1, 2, 7, 4, 5, 7} → WSS1 = 6
working set 2 at virtual time = {1, 3, 4, 3} → WSS2 = 3
working set 3 " " " = {2, 7, 7, 8, 9} → WSS3 = 5

$D(t_7) = \sum_{i=1}^{3} WSS_i(t_7) = 6 + 3 + 5 = 14$ > there are at least 15.

If D < n, system is all right.

Program 1
for (j = 0; j < A.length; j++)
... for (i = 0; i < A.length; i++)
..... A[i][j] = 0;

Program 2
for (i = 0; i < A.length; i++)
... for(j = 0; j < A.length; j++)
..... A[i][j] = 0;

How many page faults does each program experience?

Problem 8

(5+5 points)
a) On a system with paging, a process cannot access memory that it does not own. Why?

b) How could the operating system allow a process to access to other memory? (give a max 2 sentence answer)

Problem 7

Program 1:

Each iteration of i will cause page fault for that row but when it reaching accesi same row that page wait be there due to LRU. Total page fault will be 1024 * 1024 = 1048578

Program 2:

When it access the first value, it will result page fault for that row but when it again access that page is already present. For this case total page faults should be 1024. Addition to that when program code is first started, there probably another when program's stack is first used; probably would out to 1026 or 1027 page faults.

Problem 8:

An address on paging is offset and local page number. Physical page is found by searching table based on logical page number to generate physical page number, OS can limit process to accessing only those physical page allocated to the process since OS controls the contents of this table. Since page wan't be in the page table, there is'nt any possible way for process to represent page it doesn't own. OS should need to allow entries for non-process memory to be included to the process. This approach can be used when two or more processes need to swap data, they read/write same physical addresses.