Exam correction

Exercise 1 (6 marks):

1-Page size=1KB and physical memory size= 4 KB → number of frames= physical memory size/ Page size=4/1=4 frames.**(0.5 point)**

2- a. Number of program pages= size of the program / page size= 8KB/1KB=8 pages (from 0 to 7) **(0.5 point)**

b. Page number= logical @/page size, offset=remainder of the division **(1.5 point)**

Logical@	1	2076	85	1500	3648	100	4314	1025	89	5741	1219	4500	7658
(P,offset)	(0,1)	(2,28)	(0,85)	(1,476)	(3,576)	(0,100)	(4,218)	(1,1)	(0,89)	(5,621)	(1,195)	(4,404)	(7,490)

Logical@	4096	6999	7191	5140	128
(P,offset)	(4,0)	(6,855)	(7,23)	(5,20)	(0,128)

References sequence

0	**2**	**0**	**1**	**3**	**0**	**4**	**1**	**0**	**5**	**1**	**4**	**7**	**4**	**6**	**7**	**5**	**0**

3-

FIFO (1 point)

	0	**2**	**0**	**1**	**3**	**0**	**4**	**1**	**0**	**5**	**1**	**4**	**7**	**4**	**6**	**7**	**5**	**0**
F0	**0**	**0**	**0**	**0**	**0**	**0**	**4**	**4**	**4**	**4**	**4**	**4**	**7**	**7**	**7**	**7**	**7**	**0**
F1		**2**	**2**	**2**	**2**	**2**	**2**	**2**	**0**	**0**	**0**	**0**	**0**	**4**	**4**	**4**	**4**	**4**
F2				**1**	**1**	**1**	**1**	**1**	**1**	**5**	**5**	**5**	**5**	**5**	**6**	**6**	**6**	**6**
F3					**3**	**3**	**3**	**3**	**3**	**3**	**1**	**1**	**1**	**1**	**1**	**1**	**5**	**5**
	*	*		*	*		*		*	*	*		*	*	*		*	*

Page faults=13
Page fault rate=13/18=72%

LRU (1 point)

	0	**2**	**0**	**1**	**3**	**0**	**4**	**1**	**0**	**5**	**1**	**4**	**7**	**4**	**6**	**7**	**5**	**0**
F0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**7**	**7**	**7**	**7**	**7**	**7**
F1		**2**	**2**	**2**	**2**	**2**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**0**
F2				**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**5**	**5**
F3					**3**	**3**	**3**	**3**	**3**	**5**	**5**	**5**	**5**	**5**	**6**	**6**	**6**	**6**
	*	*		*	*		*			*			*		*		*	*

Page faults=10
Page fault rate=10/18=55%

LFU (1 point)

	0	**2**	**0**	**1**	**3**	**0**	**4**	**1**	**0**	**5**	**1**	**4**	**7**	**4**	**6**	**7**	**5**	**0**
F0	0_1	0_1	0_2	0_2	0_2	0_3	0_3	0_3	0_4	0_4	0_4	0_4	0_4	0_4	0_4	0_4	0_4	0_5
F1		2_1	2_1	2_1	2_1	2_1	4_1	4_1	4_1	4_1	4_1	4_2	4_2	4_3	4_3	4_3	4_3	4_3
F2				1_1	1_1	1_1	1_1	1_2	1_2	1_2	1_3	1_3	1_3	1_3	1_3	1_3	1_3	1_3
F3					3_1	3_1	3_1	3_1	3_1	5_1	5_1	5_1	7_1	7_1	6_1	7_1	51	5_1
	*	*		*	*		*			*			*		*	*	*	

Page faults=10

Page fault

The best algorithm is LRU and LFU because it gives the lowest page faults rate: 55% **(0.5 point)**

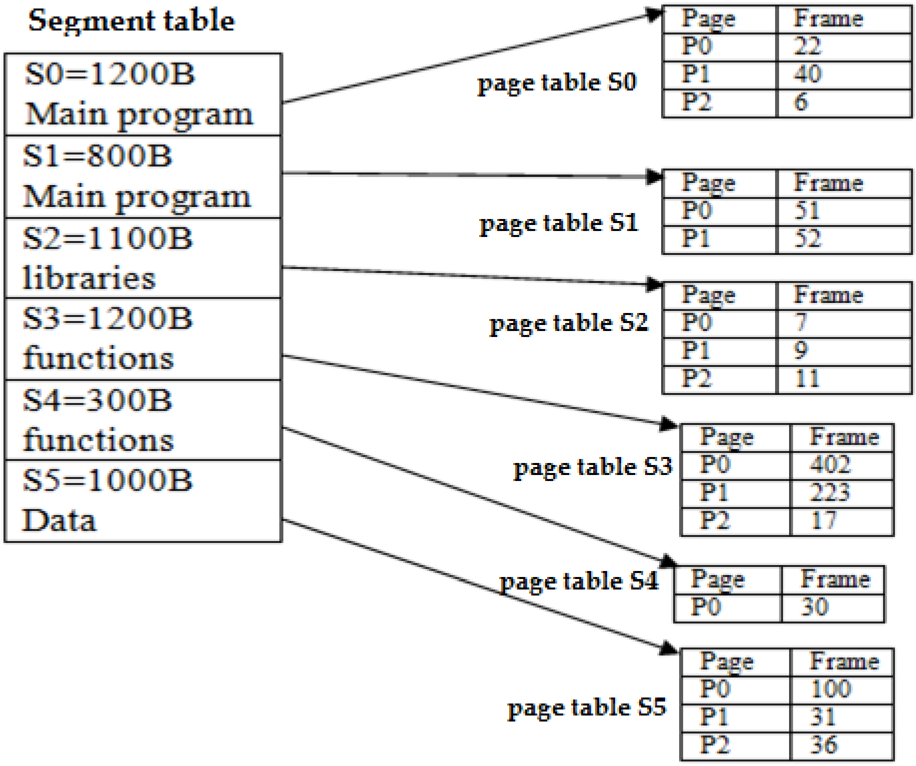
Exercise 2 (6 marks)

Q1. + Q2: **(1 point+1 point)**

Segment size maximum is 1200 B then:

S0=1200B/400B=3pages No internal fragmentation	S1=800B/400B=2pages No internal fragmentation	**S2=1100B/400B=3pages** **Internal frag=100B**
S3=1200B/400B=3pages No internal fragmentation	**S4=300B/400B=1page** **Internal frag=100B**	**S5=1000B/400B=3pages** **Internal frag=200B**

3. **(1 point)**



4. Page size=400B=2^9 B, the number of bits we need to code the offset=9 **(1 point)**
5. 9+3=12, 16-12=4bits for the page number. **(1 point)**
6. **(1 point=0.25+025+0.25+0.25)**

a) 00000101 10001111→ S= 000=S0, P=0010=P2, offset=110001111=399→ frame=6, 6*400+399=2799→ word type: main program

b) 01100010 00001110→ S=011=S3 P=0001=P1 offset=000001110=14→ frame=223, 223*400+14=89214→ word type: functions

c) 01000000 11110000→ S=010=S2 P=0000=P0 offset=0 11110000=240→frame=7→ 7*400+240=3040→ word type:libraries

d)1010001110001100→S=101=S5,P=0001=P1,offset=1 10001100=396→frame=31→31*400+396=12796→ word type: Data

Exercise 3 (5 marks)

1. The question is to be considered in the context of a non-preemptive scheduling and without input / output. In this case, each process is executed in one go and each can be treated as a single block. To order these processes, we will start by choosing a first one among the n available and execute it entirely. Then we will choose a second one among the remaining (n-1) and so on. So we see that we have:
n*(n-1)*(n-2)*...*2*1=n! Ways to order these processes. **(0.5 point)**

2. If time quantum is too short: **(0.5 point)**

<u>Advantage</u>: the response time of the processes is too small which may be tolerated in interactive environment.

<u>Disadvantage</u>: it causes unnecessarily frequent context switch leading to more overheads resulting in less throughput.

3. In multi-level scheduling, the queue of ready processes is not unique: it is divided into several queues, each containing a given type of process. The advantage of this method is that the processes (system and users, for example) do not have the same needs (memory and processor time) and must therefore be schedulated differently according to their characteristics and priorities. **(0.5 point)**

4. a- Gantt chart: **(1 point)**

P2	P4	P3	P1

0P1 P3↑3 P4↑10 15 33 41 51

b-

Gantt chart : **(1 point)**

P2	P1	P3	P4

0P1 P3↑3 P4↑10 15 25 33 51

Time	Process priority			
	P1	**P2**	**P3**	**P4**
0	2	3	4	5
5	5+10/10=2	10/15=1	2+8/8=1	5
10	10+10/10=2	5/15=0	7+8/8=0	18/18=1
15	**15+10/10=3**	**0 Termined**	**12+8/8=3**	**5+18/18=0**
20	5/10=2	/	17+8/8=3	10+18/18=2
25	**0 Termined**	/	**22+8/8=4**	**15+18/18=2**
30	/	/	3/8=0	20+18/18=2
35	/	/	/	**16/18=1**

c- Calculate the average wait time as well as the average turnaround time. **(1 point)**

Non preemptive priority			Dynamic priority		
Process	**Turnaround time**	**Waiting time**	**Process**	**Turnaround time**	**Waiting time**
P1	51	41	P1	25	15
P2	15	0	P2	15	0
P3	38	30	P3	30	22
P4	23	5	P4	41	23
Average	**31.75**	**19**	**Average**	**27.75**	**15**

d- The average turnaround time and waiting time in dynamic priority are lowest then the classic priority. **(0.5 point)**

Exercise 4 (3 points):

1. **(1 point)**

External interrupt	Internal interrupt	System call
Failures, moving a mouse, key pressed	Overflow, dividing by zero, invalid memory accesses, Page fault, access to a privilege memory area	input-output requests

2. **(1 point)**
- a buffer zone,
- control bits: state bit (ready, busy), and command bit (read, write).

3. The processor is monopolized for the duration of the I/O operation. **(0.5 point)**

4. Synchronous input/output (I/O) occurs while applications processing cannot continue until the I/O operation is complete. In contrast, asynchronous I/O operations run in the background and do not block user applications. **(0.5 point)**