Series 1: Memory Management (Contiguous Allocation) solution

Exercise 1

1.

The operating system is a set of programs for managing and controlling computer resources in order to provide an interface between the user and the physical machine.

- Resource management,
- provide a virtual machine to the user,
- process management,
- memory management,
- file management,
- I/O management,
- data protection and security

2-Depending on the memory management mode that is applied, when a program is loaded into main memory from disk, the program will be placed in a single area (**contiguous allocation**) or distributed among several areas (**non-contiguous allocation**).

- **Internal fragmentation**: The allocated memory may be larger than the required memory. This difference is internal to a partition but is not used.

-**External fragmentation**: When no partition is sufficient to accommodate a process, while the sum of the free partitions can load the process.

3.

a. **Fixed (static) partition**: consists in subdividing the memory into n partitions of fixed size. The number and size of each partition are fixed at system generation. Each partition can contain exactly one process.
b. **Variable (dynamic) partition**: consists in dividing the memory dynamically according to the demand. Each program is allocated a partition exactly equal to its size. When a program finishes its execution, its partition is recovered by the system to be allocated to another program completely or partially according to the demand.

Exercise 2

a-First fit : Choose the first suitable area. Queue={212, 417, 112 and 426 KB}

P1=100				
P2=500	**A**	A	A	A
P3=200			**C**	C
P4=300				
P5=600		**B**	B	B
initial state	**process A arrive**	**process B arrive**	**process C arrive**	**process D arrive →D waiting**

b- Best fit : choose the best suitable area. Queue={212, 417, 112 and 426 KB}

P1=100				
P2=500		**B**	**B**	**B**
P3=200			**C**	**C**
P4=300	**A**	**A**	**A**	**A**
P5=600				**D**
initial state	**process A arrive**	**process B arrive**	**process C arrive**	**process D arrive**

c-Worst fit : choose the wrong area. Queue={212, 417, 112 et 426 KO}

P1=100				
P2=500		**B**	**B**	**B**
P3=200				

P4=300			**C**	**C**
P5=600	**A**	**A**	**A**	**A**
initial state	**process A arrive**	**process B arrive**	**process C arrive**	**process D arrive**

Evaluation of the three allocation methods.

	First fit	**Best fit**	**Worst Fit**
Advantages	Simple, fast	Reduce internal fragmentation	Good management of memory space
Disadvantages	internal fragmentation	Complex	Complex

Exercise 3

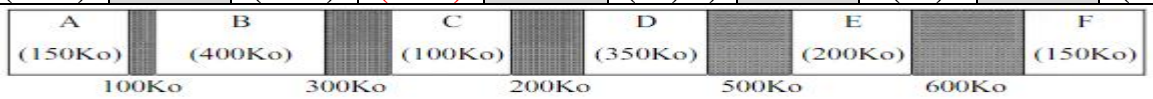
A)

1-First fit : Queue={ K (200 Ko), L (450 Ko), M (250 Ko) et O (50 Ko)}

Initial state

A (150Ko)	100Ko	B (400Ko)	K (200Ko)	100KO	C (100Ko)	200Ko	D (350)	L 450Ko	50Ko	E (200Ko)	600Ko	F (150KO)

A (150Ko)	100Ko	B (400Ko)	K (200Ko)	100KO	C (100)Ko)	200Ko	D (350)	500Ko	E (200Ko)	600Ko	F (150KO)



K(200Ko) :

L(450Ko) :

M(250Ko) :

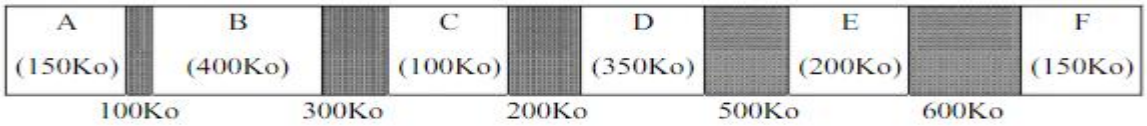
A (150Ko)	100Ko	**B** (400Ko)	**K** (200Ko)	100KO	**C** (100Ko)	200Ko	**D** (350)	**L** 450Ko	50Ko	**E** (200Ko)	**M** 250Ko	350Ko	**F** (150KO)

O(50Ko) :

A (150Ko)	**O** (50Ko)	50	**B** (400Ko)	**K** (200Ko)	100KO	**C** (100Ko)	200Ko	**D** (350)	**L** (450Ko)	50Ko	**E** (200Ko)	**M** (250Ko)	350Ko	**F** (150KO)

2-Best fit : Queue ={ K (200 Ko), L (450 Ko), M (250 Ko) et O (50 Ko)}

Initial state:



K(200Ko) :

A (150Ko)	100Ko	B (400Ko)	300KO	C (100Ko)	**K (200Ko)**	D (350)	500Ko	E (200Ko)	600Ko	F (150KO)

L(450Ko) :

A (150Ko)	100Ko	B (400Ko)	300KO	C (100Ko)	**K (200Ko)**	D (350)	**L (450Ko)**	50Ko	E (200Ko)	600Ko	F (150KO)

M(250Ko) :

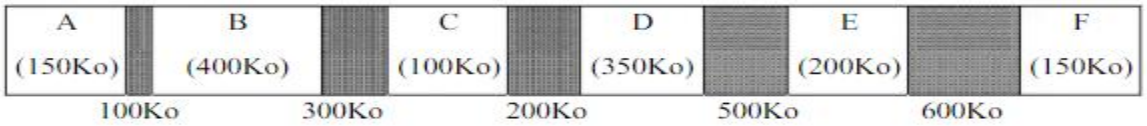
A (150Ko)	100Ko	B (400Ko)	**M (250Ko)**	50Ko	C (100Ko)	**K (200Ko)**	D (350)	**L (450Ko)**	50Ko	E (200Ko)	600Ko	F (150KO)

O(50Ko) :

A (150Ko)	100Ko	B (400Ko)	**M (250Ko)**	**O 50Ko**	C (100Ko)	**K (200Ko)**	D (350)	**L (450Ko)**	50Ko	E (200Ko)	600Ko	F (150KO)

3-Worst fit Queue ={ K (200 Ko), L (450 Ko), M (250 Ko) et O (50 Ko)}

Initial state



K(200Ko) :

A (150Ko)	100Ko	B (400Ko)	300Ko	C (100)Ko)	200Ko	D (350)	500Ko	E (200Ko)	**K (200Ko)**	400Ko	F (150KO)

L(450Ko) :

A (150Ko)	100Ko	B (400Ko)	300Ko	C (100)Ko)	200Ko	D (350)	**L (450Ko)**	50Ko	E (200Ko)	**K (200Ko)**	400Ko	F (150KO)

M(250Ko) :

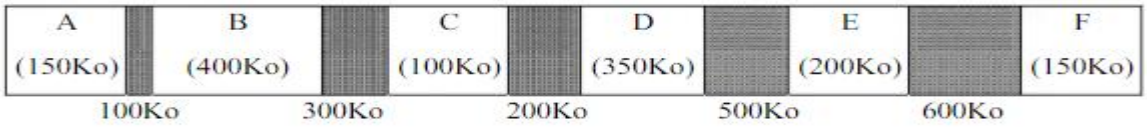
A (150Ko)	100Ko	B (400Ko)	300KO	C (100)Ko)	200Ko	D (350)	**L (450Ko)**	50Ko	E (200Ko)	**K (200Ko)**	**M (250Ko)**	150Ko	F (150KO)

O(50Ko) :

A (150Ko)	100Ko	B (400Ko)	**O (50KO)**	250Ko	C (100)Ko)	200Ko	D (350)	**L (450Ko)**	50Ko	E (200Ko)	**K (200Ko)**	**M (250Ko)**	150Ko	F (150KO)

B

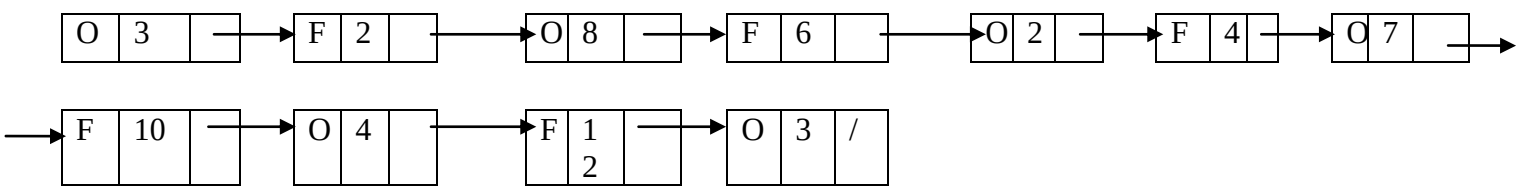
1- Bits table:



Example: for the full partition 150KB/50=3units, we simulate the three full units by 3 boxes of 1
For the empty partition 100Ko/50=2 units, we simulate the two empty units by 2 boxes of 0….etc

1	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
1	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
1									

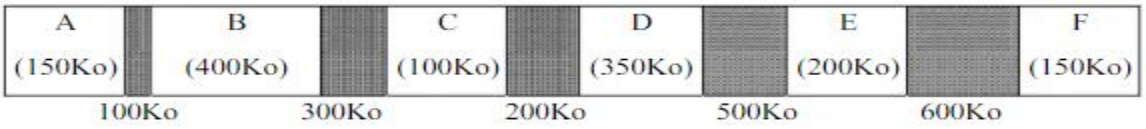
2-linked list:



C\

1. Represent the state of the memory after loading P=400KB according to the First Fit allocation strategy:

Initial state:



P=400KB:

A (150Ko)	100Ko	B (400Ko)	300Ko	C (100)Ko)	200Ko	D (350Ko)	**P (400Ko)**	100Ko	E (200Ko)	600Ko	F (150KO)

2. Describe the steps required to locate the correct location of the new process in memory using:

a. Bit table:

Calculate the size of the process in units and:

0. Initialize an index with 0, then iterate through the bit table,
1. Test if the current bit is equal to a zero, go to step 2 otherwise go to step 5.
2. Calculate the number of zeros (by incrementing the index),
3. If the result is less than the size of the process then advance, and return to 1
4. If the result equals the size of the process we will stop, then we replace each zero (forward backwards) by the number of the calculated index.
5. Advance and return to step1.