Q1

1. A particular computer system provides a total memory of 4800 words to users. It supports multiprogramming with multiple-partition. At a given moment in time, the memory is occupied by four processes:

Starting Address of Process	Length (words)
1000	1200
3100	700
4000	400
4400	200

Q1 (2)

When a new process requests memory, the following method is employed:

- STEP 1: Use the Best-fit algorithm to allocate memory for this new process. Do Step 2 only if there is no hole which fits the memory request.
- STEP 2: Do memory compaction by moving occupied partitions towards lower memory addresses. Compaction stops when a sufficiently large hole is formed for the new memory request. If, at the end of compaction, there is still insufficient space, the process requesting for more memory is blocked.

Q1 (3)

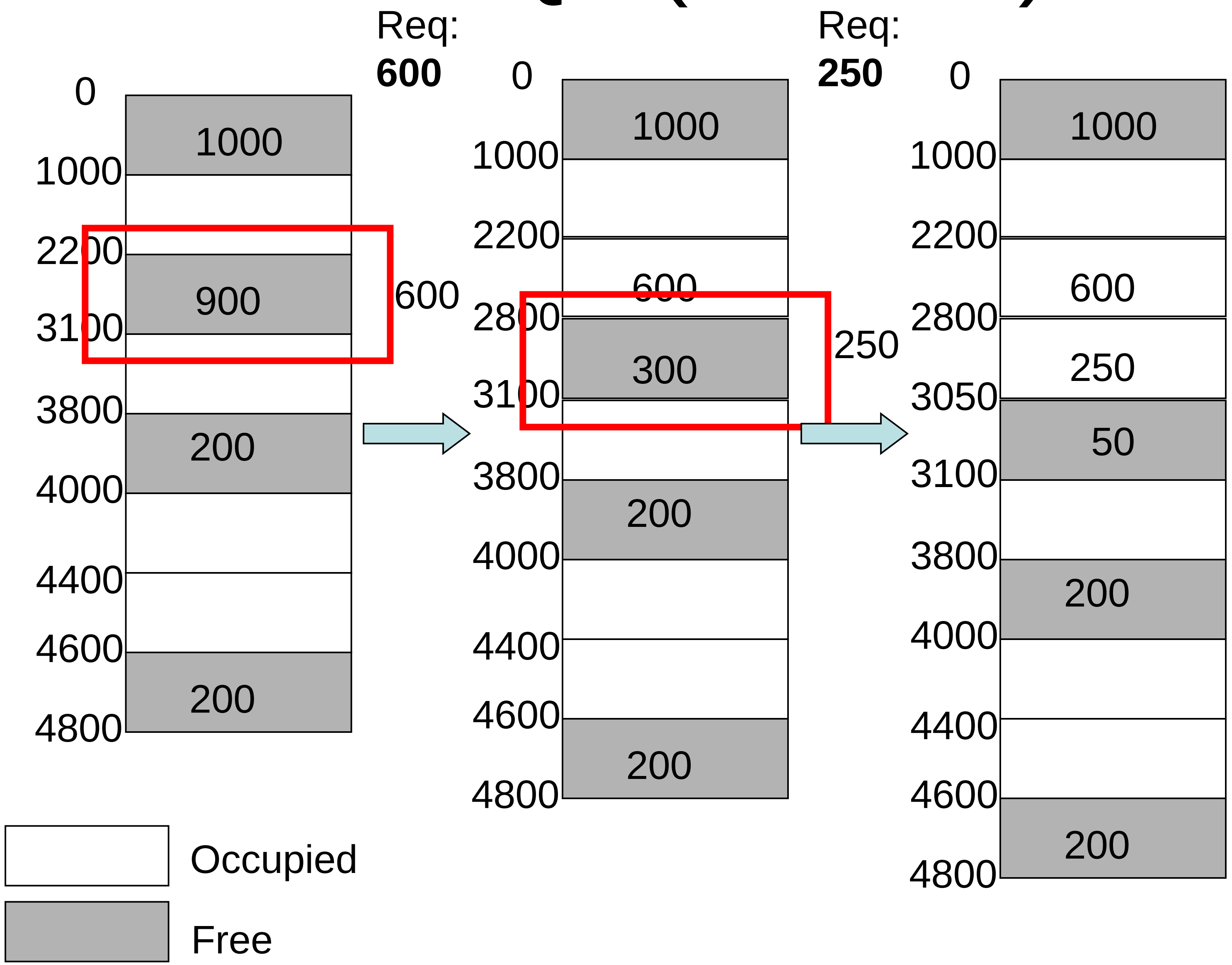
The Operating System receives three new memory requirements from processes: 600, 250 and 1050 words (requested in this order).

a) Show the memory allocation after all three requests have been serviced.

b) If the algorithm in Step 1 had been First-fit, show the memory allocation after all three requests have been serviced.

c) In this particular case, which algorithm yields a lower overhead in terms of total relocation of occupied partitions?

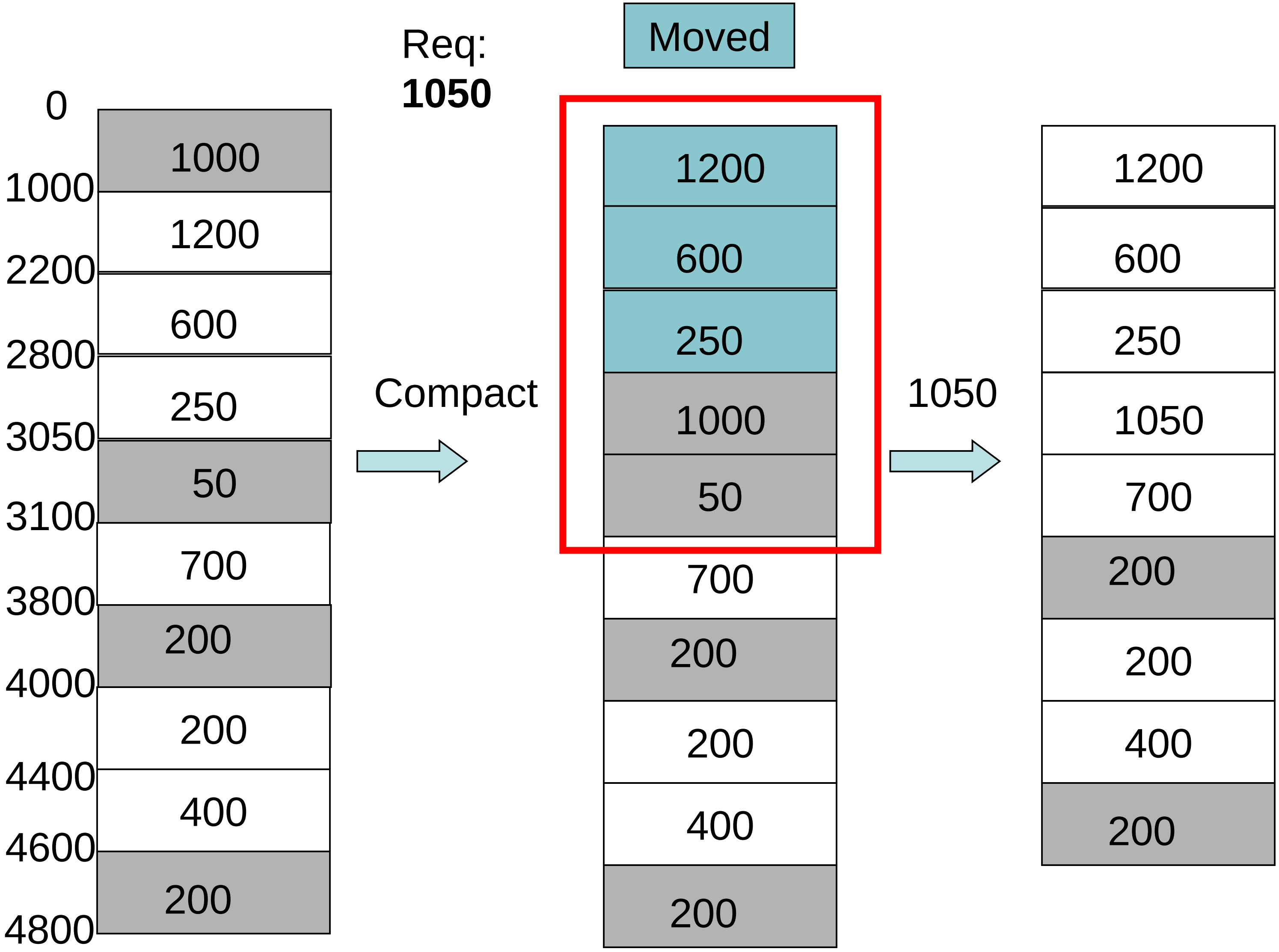
Q1 (Best Fit)



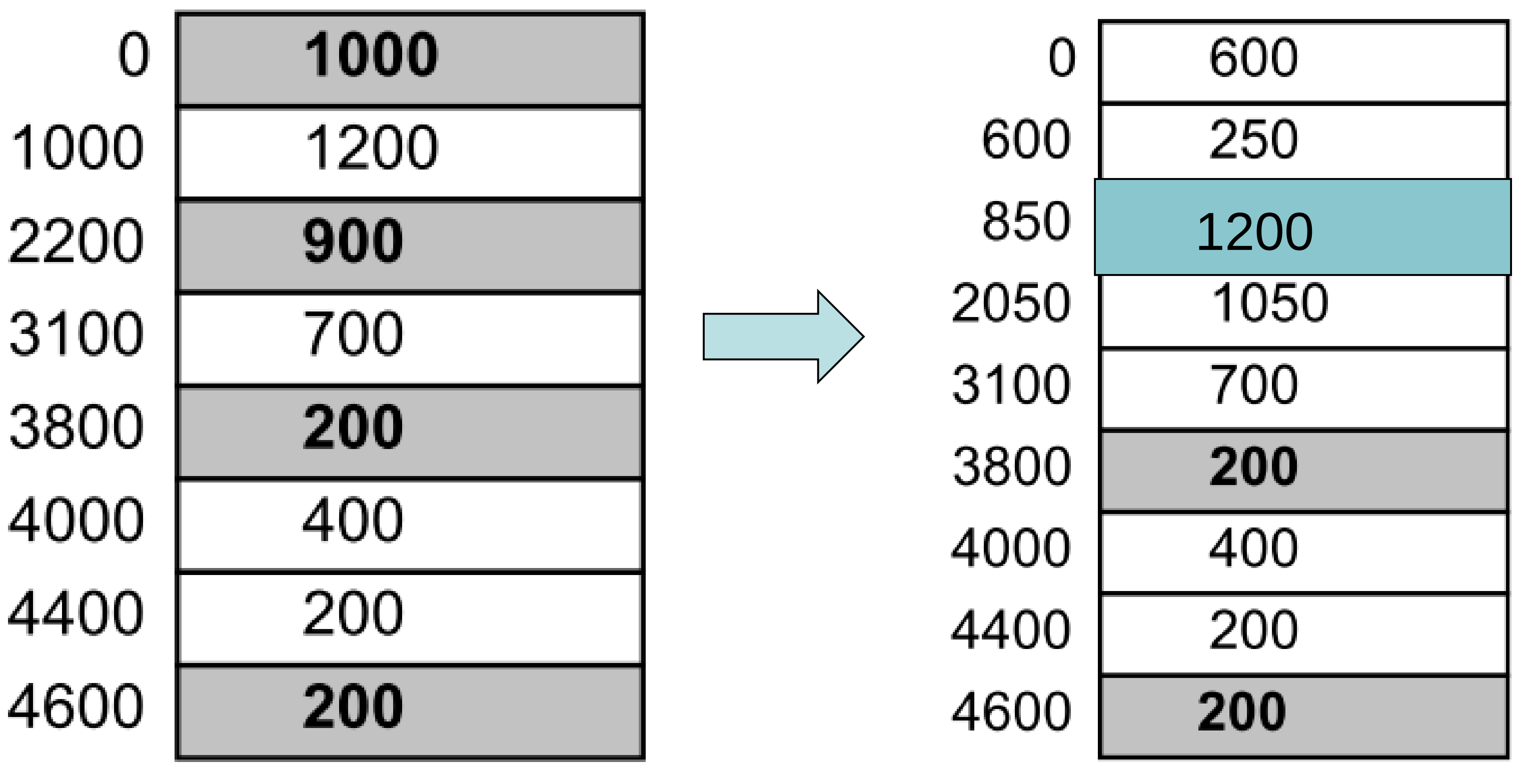
Req:
600
0
1000
1000
2200
900
600
3100
3800
200
4000
4400
4600
200
4800
0
1000
1000
2200
600
2800
300
250
3100
3800
200
4000
4400
4600
200
4800
Req:
250
0
1000
1000
2200
600
2800
250
3050
50
3100
3800
200
4000
4400
4600
200
4800
Occupied
Free

Q1 (Best Fit)

Req:
1050



Q1 (b): First Fit



Q1 (c)

The total movement of partitions:

- Best Fit: 2050 words (1200 + 600 + 250)
- First Fit: 1200 words.
- In this particular case, First Fit has lower overhead.

Q2

2. a) Identify memory allocation schemes that suffer from internal fragmentation.

b) Explain why memory compaction cannot be performed if absolute address format is used in the code.

Q2a

2. a) Identify memory allocation schemes that suffer from internal fragmentation.

→ fixed partitioning and paging

Q2

2. b) Explain why memory compaction cannot be performed if absolute address format is used in the code.

→If absolute address is used, the code and data cannot be moved around in the memory (otherwise, addressing error happens).

→Compaction requires moving code and data in the memory and so it cannot be performed.

Q3 Memory Access Time

Consider a paging system with the page table stored in memory.

a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

b) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

Q3 (a)

- 400 nanoseconds, two memory accesses:
  - One for the page table
  - One for the word in memory.

Q3 (b)

- Access time
  - 200, when the reference in associative registers.
  - 400, otherwise.
- Effective access time = 0.75 x (200 nanoseconds) + 0.25 x (400 nanoseconds) = 250 nanoseconds.

Q4

Consider a computer system with a 32-bit logical address and 1-Kbyte page size. The system has 1 Gbyes of physical memory.

a) Give the format of both the logical and physical addresses of this system.

b) How many entries are there in a page table?

c) If an inverted page table is used, how many entries are there?

Q4 (a)

Consider a computer system with a 32-bit logical address and 1-Kbyte page size. The system has 1 GByes of physical memory.

a) Give the format of both the logical and physical addresses of this system.

<u>Logical Address</u>

Page number	Page offset

<u>Physical Address</u>

Frame number	Page offset

	Logical Address		Physical Address	
Step 1: total number of bits	**32**		**30**	
Step 2: offset		**10**		**10**
Step 3: page/frame number	**22**		**20**	

Q4 (b)

Consider a computer system with a 32-bit logical address and 1-KByte page size. The system has 1 GBytes of physical memory

b) How many entries are there in a page table?

→ 2^{22}

Logical Address

Page number	Page offset
22 bits	10 bits

Q4 (c)

Consider a computer system with a 32-bit logical address and 1-Kbyte page size. The system has 1 Gbyes of physical memory

c) If an inverted page table is used, how many entries are there?

→ 2^{20}

Physical Address

Frame number	Page offset
20 bits	10 bits