Chapter 2 Exercises

1. Describe a present-day computing environment that might use each of the memory allocation schemes (single user, fixed, dynamic, and relocatable dynamic) described in the chapter. Defend your answer describing the advantages and disadvantages of the scheme in each case. ANS: Answers in these cases will vary. Look for a fundamental understanding of the comcepts presented about memory allocation. Students should present original scenarios here because answers can range from real-time or embedded computing described briefly in Chapter 1 to current environments.
2. How often should memory compaction/relocation be performed? Describe the advantages and disadvantages of performing it even more often. ANS: Memory compaction should be performed on a basis that the operator finds most efficient for the local stream of jobs. Some industry analysts suggest 75 percent. Performing it more often might keep more memory available but would necessarily result in increased overhead because processing cannot take place during compaction. Likewise, performing it less often might keep deallocated memory from being used in the most efficient manner possible.
3. Using your own words, explain the role of the bounds register. ANS: This hardware element, the bounds register, tracks the limits of addressable memory for this job and prevents it from trying to access memory that's allocated to another job.
4. Describe in your own words, the role of the relocation register. ANS: This hardware element, the relocation register, tracks the number of bytes that a job has been moved in main memory as a result of memory compaction. It is used to track each instruction so it can be found successfully after the job's relocation.
5. Give an example of computing circumstances that would favor first-fit allocation over best-fit. Explain your answer. ANS: If speed is of primary concern, first-fit would be preferable. Look for original examples for each student – examples of system that would need to be speedy.
6. Given the following information:

Job List:	**Memory Block List:**

Job Number	Memory Requested
Job A	690K
Job B	275K
Job C	760K

Memory Block	Memory Block Size
Block 1	900K (low-order memory)
Block 2	910K
Block 3	300K (high-order memory)

a. Use the first-fit algorithm to indicate which memory blocks are allocated to each of the three arriving jobs. ANS: Job A is allocated to Block 1, Job B is allocated to Block 2, Job C is waiting

b. Use the best-fit algorithm to indicate which memory blocks are allocated to each of the three arriving jobs. ANS: Job A is allocated to Block 1, Job B is allocated to Block 3, Job C is allocated to Block 2

7. Given the following information:

Job List:	
Job Number	**Memory Requested**
Job A	57K
Job B	920K
Job C	50K
Job D	701K

Memory Block List:	
Memory Block	**Memory Block Size**
Block 1	900K (low-order memory)
Block 2	910K
Block 3	200K
Block 4	300K (high-order memory)

a. Use the best-fit algorithm to indicate which memory blocks are allocated to each of the three arriving jobs. ANS: (Job A is allocated to Block 3, Job B is allocated to waiting, Job C is allocated to Block 4, Job D is allocated to Block 1)

b. Use the first-fit algorithm to indicate which memory blocks are allocated to each of the three arriving jobs. ANS: (Job A is allocated to Block 1, Job B is allocated to waiting, Job C is allocated to Block 2, Job D is allocated to waiting. Notice that

neither algorithm can allow Job B to move into memory because no partition is large enough for it.)

8. Next-fit is an allocation algorithm that starts by using the first-fit algorithm but keeps track of the partition that was last allocated, instead of restarting the search with Block 1, it starts searching from the most recently allocated block when a new job arrives. Using the following information:

Job List:	
Job Number	**Memory Requested**
Job A	590K
Job B	50K
Job C	275K
Job D	460K

Memory Block List:	
Memory Block	**Memory Block Size**
Block 1	100K (low-order memory)
Block 2	900K
Block 3	280K
Block 4	600K (high-order memory)

Indicate which memory blocks are allocated to each of the three arriving jobs, and explain in your own words what advantages the next-fit algorithm could offer. ANS: (Job A is allocated to Block 2, Job B is allocated to Block 3, Job C is allocated to Block 4, Job D is waiting because it doesn't fit into Block 1)

9. Worst-fit is an allocation algorithm that allocates the largest free block to a new job. It is the opposite of the best-fit algorithm. Compare it to next-fit conditions given in Exercise 8 and explain in your own words what advantages the worst-fit algorithm could offer. ANS: Job A is allocated to Block 2, Job B is allocated to Block 4, Job C is allocated to Block 3, Job D is waiting because it doesn't fit into Block 1, As an alternative, ask students to try the best-fit with this data and all four jobs will fit into the four available memory blocks.

10. Imagine an operating system that cannot perform memory deallocation. Name at least three effects on overall system performance that might result and explain your answer.

    ANS: Frequent downtime so the system can deallocation during system restarts, slower response because of memory allocation inefficiencies, inefficient use of memory, etc.

11. In a system using the relocatable dynamic partitions scheme, given the following situation (and using decimal form): Job Q is loaded into memory starting at memory location 42K.

    a. Calculate the exact starting address for Job Q in bytes. ANS: 43008 (42*1024 = 43008)

    b. If the memory block has 3K in fragmentation, calculate the size of the memory block.

    ANS: 46080 (3*1024 = 3072) + (42*1024 = 43008) = 46080

    c. Is the resulting fragmentation internal or external? Explain your reasoning. ANS: Look for a fundamental understanding of the differences between internal and external fragmentation.

12. In a system using the relocatable dynamic partitions scheme, given the following situation (and using decimal form): Job W is loaded into memory starting at memory location 5000K.

    a. Calculate the exact starting address for Job W in bytes. ANS: 5120000 (5000*1024 = 5120000)

    b. If the memory block has 3K in fragmentation, calculate the size of the memory block.

    ANS: 5130240 (10*1024 = 10240) + (5000*1024 = 5120000) = 5130240

    c. Is the resulting fragmentation internal or external? Explain your reasoning. ANS: Look for a fundamental understanding of the differences between internal and external fragmentation.

13. If the relocation register holds the value -83968, was the relocated job moved toward lower or higher addressable end of main memory? ANS: lower addressable end of memory) By how many kilobytes was it moved? ANS: negative 82 (-83968 / 1024 = -82) Explain your conclusion.

14. In a system using the fixed partitions memory allocation scheme, given the following situation (and using decimal form): After Job J is loaded into a partition of size 50K, the resulting fragmentation is 7168 bytes:

    a. What is the size of Job J in bytes? ANS: 44032 bytes (50*1024 = 51200) – 7168 = 44032 bytes

    b. What type of fragmentation is caused? ANS: Internal fragmentation

15. In a system using the dynamic partition memory allocation scheme, given the following situation (and using decimal form): After Job C of size 70K is loaded into a partition resulting in 7K of fragmentation, calculate the size (in bytes) of its partition ANS: 71680 (70*1024 = 71680) and identify the type of fragmentation that is caused? (External fragmentation) Explain your answer. (Students should show their calculations and explain how they know that the fragmentation is between partitions.)

Advanced Exercises

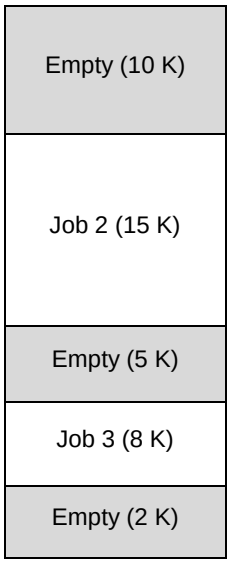
16. The relocation example presented in the chapter implies that compaction is done entirely in memory, without secondary storage. Can all free sections of memory be merged into one contiguous block using this approach? Why or why not? ANS: This question can be answered using the following pseudocode and example, which show that all free sections of memory can be merged into one contiguous block without using secondary storage as a temporary holding area. It could generate a discussion on additional hardware components needed to implement this method of compaction.

Example:

Main Memory

0 K

Operating System
Job 1 (5 K)



60 K

Job 4 requires 15K and a block that large is not yet available. Therefore, compaction is needed.

Pseudo code:

```
compaction-case = 0
k = 1
DO-WHILE there are free-partitions
        j = 1
      DO-WHILE there are active jobs
              IF free-partition-size(k) = active-job-size(j)
           THEN compaction-case = 1
                relocate active job(j) into free-partition(k)
         ELSE
                 IF free-partition-size(k) > active job-size(j)
              THEN compaction-case = 2
                 relocate active-job(j) into free-partition(k)
                free-partition-size (k) = free-partition-size (k) - active-job-size(j)
           ELSE compaction-case = 3
                 difference = active job-size(j) - free-partition-size(k)
                 beginning = 0
                DO-WHILE beginning not = active job-size(j)
                     ending = beginning + free-partition-size(k)
                     relocate active job(j) from beginning to ending
                     release active job(j) from beginning to ending
```

```
                    beginning = ending
                    difference = difference - free-partition-size(k)
                      IF difference < 0
                         THEN difference = difference - free-partition-size(k)
                              free-partition-size(k) = difference
                      END-IF
                  END-DO-WHILE
               END-IF
            END-IF
            j = j + 1
      END-DO-WHILE
            k = k + l
   END-DO-WHILE
   IF compaction-case = 0
      THEN "Compaction is not possible"
   ELSE
      perform update of free and busy lists
END-IF
```

17. To compact memory in some systems, some people suggest that all jobs in memory be copied to a secondary storage device and then reloaded (and relocated) contiguously into main memory, thus creating one free block after all jobs have been recopied into memory. Is this viable? Could you devise a better way to compact memory? Write your algorithm and explain why it is better. ANS: This type of compaction may prove to be more time consuming than the previous one because it requires access to a secondary storage device. If the operating system and computer are equipped with "block" transfer and buffers (these are discussed later in the text), then the transfer time could be optimized. In addition, if the disk was a device dedicated only to compaction, then its hardware components could also be set to optimize the seek time. When using secondary storage one needs to remember that a store and load operation are performed every time increasing the time needed to complete compaction.

18. Given the memory configuration in the figure below, answer the following questions. At this point, Job 4 arrives requesting a block of 100K.

    a. Can Job 4 be accommodated? Why or why not?

b. If relocation is used, what are the contents of the relocation registers for Job 1, Job 2, and Job 3 after compaction?

c. What are the contents of the relocation register for Job 4 after it has been loaded into memory?

d. An instruction that is part of Job 1 was originally loaded into memory location 22K. What is its new location after compaction?

e. An instruction that is part of Job 2 was originally loaded into memory location 55K. What is its new location after compaction?

f. An instruction that is part of Job 3 was originally loaded into memory location 80K. What is its new location after compaction?

g. If an instruction was originally loaded into memory location 110K, what is its new location after compaction?

ANS: a. Job 4 cannot be accommodated because there is not enough contiguous free memory available.

b.

Contents of Relocation Registers	Job Number
0	1
-20480 (-20K)	2
-30720 (-30K)	3

c. The relocation register for Job 4 is set to zero because it has just been loaded into memory.

d. Its location has not changed because Job 1 has not been relocated.

e. Its location is: 55K - 20K = 35K (or 35840)

f. Its location is: 80K - 30K = 50K (or 51200)

g. Its location is: 110K - 30K = 80K (or 81920)

h. That value does not change. The true address is computed when execution occurs.

Programming Exercises

Exercises **19**, **20,** and **21** are best explained during a few class sessions using a sample paper-and-pencil model to highlight the structure of the program modules and their interrelationships. Sample outputs are very helpful to the students as well as a sample of the analysis that can be performed on the data collected.

Typically, the procedures used in the simulation are:

i. Main Procedure: contains the calls to all other procedures.

ii. Input Procedure: where all data is submitted.

iii. Time Procedure: where the system clock is updated to indicate how long it took to process all jobs and where the time to run a job (job clock) is updated.

iv. Waiting Procedure: used if a job is put on a waiting queue. The job's waiting clock is updated and the waiting queue count is increased.

v. Snapshot Procedure: used to print out the status of the job table and the memory table at even intervals. This gives a good trace of how the system is behaving.

vi. Winding Procedure: used to handle all jobs still active or waiting after there are no more incoming jobs.

vii. Statistics Procedure: used to produce the typical measures requested by the exercise.

A policy has to be developed to handle the conflict of which job will be served first: the one in the waiting queue or the incoming one. A related policy must be defined to determine whether no more incoming jobs will be accepted until the ones in the waiting queue are served or the system will flip-flop between incoming and waiting jobs.

The data for the jobs is best stored into a table with columns indicating: job size, job time to run, waiting time and status (new, running, waiting, too big or done). The data for the memory blocks is best stored into a table with columns indicating: block size, status (busy or free), number of jobs currently using block, counter for number of times block has been used.

Sample Snapshot:

Job Table

Job Number	Run Time	Job Size	Job Status	Wait Time	Completion Time
01	05	576	done	00	05
02	04	419	done	00	04
03	08	329	done	00	08

04	02	203	done	00	02
05	02	255	done	00	02
06	06	699	done	00	06
07	08	894	running	00	
08	10	074	running	00	
09	07	393	running	00	
10	06	689	running	01	
11	05	658	waiting	03	
12	08	382	running	00	
13	09	914	new		
14	10	042	new		
15	10	022	new		
16	07	754	new		
17	03	321	new		
18	01	138	new		
19	09	985	new		
20	03	361	new		
21	07	754	new		
22	02	271	new		
23	08	839	new		
24	05	595	new		
25	10	076	new		

Memory Table

Block Number	Block Size	Block Status	Job Number	Number of Times Used	Internal Fragmentation
01	950	busy	10	3	261
02	700	busy	08	2	626
03	450	busy	12	2	068
04	850	busy	2		457
05	300	free		1	
06	900	busy	07	1	006
07	100	free			
08	550	free			
09	150	free			
10	050	free			