King Saud University
College of Computer and Information Sciences
Computer Science Department

College of Computer & Information Sciences
Computer Science Department

CSC227: Operating Systems
Course Project – S2– 1442

Date: February 14, 2021.
Due Date:
Phase 1: Sunday March 14 by 11:59 pm.
Phase 2: Saturday April 4 by 11:59 pm.

This project is to be solved in teams of no more than three students.

I. Problem Statement

This project will involve managing a contiguous region of memory of fixed size *M* partitions, where the size of each partition can be unequal. Your program must respond to four different requests:

1. Request for a contiguous block of memory.
2. Release of a contiguous block of memory.
3. Report detailed information about regions of free and allocated memory.
4. Exit the program.

Your program will allocate memory using one of the three contiguous memory allocation approaches [First-fit (F), Best-fit (B), or Worst-fit (W)], depending on the user-entered selection. This will require that your program keep track of the different holes representing available memory. When a request for memory arrives, it will allocate the memory from one of the available partitions based on the selected allocation strategy. Your program will also need to keep track of which region of memory has been allocated to which process in order to be able to handle a request for releasing an allocated memory or generating a status report.

Note: Groups consisting of only two students may select any **two** memory allocation approaches from [(F), (B), (W)].

II. Project Description

In this project, the students will write a program to allocate and de-allocate a contiguous block of memory and provide a status report about the current state of memory. The generated report should provide detailed information about each partition, which include [partition status, size of a partition, starting and end address of a partition, the process currently allocated in the partition, the size of the internal fragmentation]. The program should behave as follows:

Phase 1: Memory Initialization - By the end of this step, the memory has been initialized and status report displayed.

1. The program will prompt the user to enter the number of partitions (M).
2. The program will prompt the user to enter the size of each partition in KB.
3. The program will create a memory array of size *M* and initialize the partitions attributes:
   - Partition status: Allocated or Free.
   - Size of a partition in KB.
   - Starting address of a partition in Bytes.
   - Ending address of a partition in Bytes.

- The name of the process if Status = Allocated, or Null otherwise.
- The size of the internal fragmentation in KB if the status is "Allocated", -1 otherwise.

4. The program will output on the console a status report of each partition in the memory with the following information:
   - The address space of each partition represented by bytes (starting and ending address).
   - Partition status.
   - Partition size represented by KB.
   - The current allocated process to the partition (if applicable).
   - Internal fragmentation size for each partition in KB (if applicable).

Phase 2: Memory allocation - By the end of this step, the program should allocate processes if there is sufficient memory to be allocated to a request, otherwise, it will output an error message and reject the request. It should also be able to de-allocate processes from the memory.

The program will prompt the user to input her choice [1 (request), 2 (release), 3 (status report), or 4 (exit)] based on the list mentioned in the problem statement above.

1. If the user selects option 1 (request), the user should be requested to enter the allocation request's information, namely, the process name, the size of the process in KB, and the allocation strategy. For example, a request for allocating a process of size 40 KB using the worst fit strategy will be entered as follows: P0 40 W
   If there is insufficient memory to allocate to the request, it will output an error message and reject the request.
2. If the user selects option 2 (release), the user should be requested to enter the process name to be released from the memory. For example, a request for de-allocating (releasing) the memory that has been allocated to process P0 will appear as follows: P0
3. If the user selects option 3 (status report), your program will report list of detailed information about each block in memory as described in *phase1 (point 4)* and write this information to the output file (*Report.txt*).
4. If the user selects option 4 (exit), you should exit the program.

III. Deliverables

The Team Leader is required to submit a single zip folder on the behalf of the group in each phase containing the following. Note that performing the allocation request and release are only required in Phase 2.

1. Program code in softcopy. Java language can be used.
2. Program Output Report:
   - For **Phase I**: Screen shot showing sample input/output for one run in which a status report should be shown after configuring the memory and initializing the partitions attributes.
   - For **Phase II**: Screen shots showing sample input/output for a minimum of **five** runs, which include at least **three** allocation requests using the **three different** allocation strategies, and insufficient allocation request, and one release request. In each run, a status report should be shown after performing each request.

3. A Read Me file in PDF containing the following:
   a. Student names and IDs highlighting Team Leader.
   b. Table showing task distribution. *There should be a clear and practical distribution of tasks.*
   c. Instructions on how to execute the program.
   d. Student reflection on the simulation. This is a maximum of one paragraph evaluating the performance of the three allocation strategies and reflecting on the results of the simulation. Students may provide suggestions for improving the performance.
   e. Student peer evaluation form (see Team Work Evaluation table below).

IV. Evaluation Rubric

Code	
Fully satisfied	1
Partially satisfied	0.5
Not satisfied	0

Evaluation rubric is divided into two parts: First part evaluates Team Work and second part addresses the functional requirements. The code in grade assignment is shown to the right.

Team Work Evaluation.

Part 1: Team Work			
Criteria	**Student 1**	**Student 2**	**Student 3**
Work division: Contributed equally to the work			
Peer evaluation: Level of commitments (Interactivity with other team members), and professional behavior towards team & TA			
Project Discussion: Accurate answers, understanding of the presented work, good listeners to questions			
Time management: Attending on time, being ready to start the demo, good time management in discussion and demo.			
Total/4	0	0	0

Functional Requirements.

Part 2: Functional Requirements		
	Criteria	**Evaluation**
General	Overall quality of the code implementation (organization, clearness, design,…)	
	Subtotal/1	0
Phase 1	Create and initialize a memory configuration.	
	Compute the partition attributes and store them in appropriate variables.	
	Display the status report clearly, correctly and completely.	
	Subtotal/3	0
Phase 2	Logic for allocating available memory partition using **First-fit** strategy task is correct and complete.	
	Logic for allocating available memory partition using **Best-fit** strategy task is correct and complete.	
	Logic for allocating available memory partition using **Worst-fit** strategy task is correct and	

	complete.	
	Logic for handling the failure of allocation memory request is correct and complete.	
	Logic for de-allocating memory partition task is correct and complete.	
	Logic for calculating the internal fragmentation is correct and complete.	
	Program displays the desired output (Report.txt) correctly and completely.	
	Subtotal/7	0
	1. Program code as in III.1.	
Deliverables	2. Program Output Report as in III.2.	
	3. A Read Me file in PDF as in III.3.	
	Subtotal/3	0
	Total/14	0