UNIVERSITI PUTRA MALAYSIA

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

10 %

PROJECT

Memory Placement Algorithm and Frame Replacement Algorithm

ASSIGNMENT TYPE: Group of two persons

OBJECTIVE:
The main objective of this project is to help student to understand the algorithms for memory placement used in operating system and the frame replacement in virtual memory, and how to implement the algorithm using computer program.

LEARNING OUTCOME (PLO):
CPS3-03: Construct computational solutions using appropriate algorithms and techniques for non-routine problems.

DUE DATE OF SUBMISSION: **4 JANUARY 2026 (SUNDAY), 11.59PM**
Submit through PutraBlast, PDF file.

INSTRUCTIONS:
Implement Java program for the following algorithms. The code must include comments about the methods in the code. Submit a report with the output screen capture, along with short explanations of the code and the output.

1. Implement the memory placement algorithms

Assume we have this scenario:

A system contains the following list of memory partitions with the specified size (unit measured by Kilobytes/Kb), where partition marked with 'X' assumed is occupied by a segment and thus can't be used for allocation/placement, and 'R' is the most recent placement inside the partitions:

	X		R		X	
100	20	80	50	50	120	100

Based the given scenario, create a single or several Java programs which perform(s) a simulation of memory placement algorithm using

- First Fit algorithm
- Next Fit algorithm
- Best Fit algorithm

To allow user to test the simulation, each program will allow the user to specify as input,

- a total (N) of new pages/segments to be placed inside the partitions
- a list of N integers (each integer, in Kb, represents the size of a new page/segment to be placed)

Assume the user entered some data as input (total=5 pages/segments). Then the program must display the resulting list of page/segment placements such as follows:

```
Total of pages to place: 5
List of pages to place (in Kb): 20 40 30 100 20
Placement result: 20, 40, 30, X, 20, R, -, X, 100
```

(Alternative output – all algorithms in 1 program)

```
Specify total insertion(s) : 5
Input 5 insertion(s) in Kb: 20 40 30 100 20
Result:
Scenario (R=recent, X=occupied) : -, X, -, R, -, X, -,
First-fit placement : 20, 40, 30, X, 20, R, -, X, 100,
Next-fit placement : 100, X, 20, R, 20, X, 40, 30,
Best-fit placement : 100, X, 40, 20, R, 20, 30, X, -,
```

Also, print an error message if the remaining memory in the existing scenario is too small to accept the page/segment placement specified in the input e.g. 'Error: the remaining memory is not enough to place more pages'.

2. Implement the frame replacement algorithm for virtual memory

Assume a computer system have 10 memory frames available inside the physical memory and is required to execute a process containing 20 pages. Assume a process P has been executed in the system and produced a sequence of 40 page demands as follows:

Page demands trace of process P

Demand	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Page	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

cont. below

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	2	1	1	2	6	7	8	9	1	5	6	7	7	11	15	16	11	20	2

Implement the following frame replacement algorithms (either in separated or a single Java program):

- **First-in-first-out (FIFO)** – select the candidate that is the first one that entered a frame
- **Least-recently-used (LRU)** – select the candidate that is the least referred / demanded
- **Optimal** – select the candidate based on future reference which such process will be the least immediately referred / demanded.

The program for each algorithm should allow the user to specify:

- the total of frames provided in memory (F)
- the total of page demands (N)
- a list of N page demands.

Use the demands' sequence to calculate the number of page faults produced by each algorithm (or your program may ask the input once and use the same input for all the algorithms).

Using the example given (10 frames, 40 page demands), compare which of the algorithms produce the least amount of page faults for process P.

Rubric for Project

Criteria	Full Marks	Descriptions	
Programming skill	5	5	Code is well structured and executes perfectly.
		4	Code is well structured but contains some irrelevant declarations.
		3	Code can be executed but data structure can still be improved.
		2	Code can be executed but data structure is not efficient.
		1	Code has runtime errors and/or incorrect output.
		0	No submission / Code cannot be executed.
Output	3	3	Output is neat and complete. Student shows **extra effort** in completing the assignment.
		2	Output is acceptable and fulfils the requirements of the assignment.
		1	Output does not satisfy all the requirements of the assignment. Student has minimal effort in completing it.
		0	No submission / Output is not relevant / Code cannot be executed.
Neatness of coding	2	2	Coding is neatly done with complete comments (explanations) for all functions.
		1	Coding is poorly done with insufficient comments (explanations) for each function.
		0	No submission / Coding is poorly done with no comments (explanations) for each function.