**Question 1 (25 Points).**

Consider the following snapshot of a system. There are three types of resources: E, F and G and Four processes P0, P1, P2 and P3.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Allocation** | | | **Max** | | | **Available** | | |
| **E** | **F** | **G** | **E** | **F** | **G** | **E** | **F** | **G** |
| **P0** | 1 | 0 | 1 | 4 | 3 | 1 | 3 | 3 | 0 |
| **P1** | 1 | 1 | 2 | 2 | 1 | 4 |  |  |  |
| **P2** | 1 | 0 | 3 | 1 | 3 | 3 |  |  |  |
| **P3** | 2 | 0 | 0 | 5 | 4 | 1 |  |  |  |

Answer the following questions using the banker's algorithm:

(a) What is the content of the matrix Need? **(5 Points)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Need** | | |
| **E** | **F** | **G** |
| **P0** | 3 | 3 | 0 |
| **P1** | 1 | 0 | 2 |
| **P2** | 0 | 3 | 0 |
| **P3** | 3 | 4 | 1 |

(b) Is the system in the safe state? If yes, write the safe sequence of processes. Show the detailed steps. **(1+2+12 Points)**

P0: 3 3 0 <= 3 3 0 : true -> new available = 3 3 0 + 1 0 1 = 4 3 1

P1: 1 0 2 <= 4 3 1 : false

P2: 0 3 0 <= 4 3 1 : true -> new available = 4 3 1 + 1 0 3 = 5 3 4

P3: 3 4 1 <= 5 3 4 : false

P1: 1 0 2 <= 5 3 4 : true -> new available = 5 3 4 + 1 1 2 = 6 4 6

P3: 3 4 1 <= 6 4 6 : true -> new available = 6 4 6 + 2 0 0 = 8 4 6

=> [ 0, 2, 1, 3 ] : it is safe

(c) If a request from process P3 arrives for (3,3,1), can the request be granted immediately? If the answer is yes then write the reason. If the answer is no, then also write the reason. **(5 Points)**

pi <= need(i) : 3 3 1 <= 3 4 1 -> true

pi <= available(i): 3 3 1 <= 3 3 0 -> false

=> process P3 arrives for (3,3,1), the request can not grante because the process is greater than available.

**Question 2 (30 Points)**. Consider a main memory with five page frames and the following sequence of page references 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3.

(a) Calculate number of page faults using Optimal and Least Recently Used (LRU) algorithm. **(14 + 14 points)**

- Optimal: (xét dãy bên phải)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 8 | 2 | 3 | 9 | 1 | 6 | 3 | 8 | 9 | 3 | 6 | 2 | 1 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 1 | 1 |
|  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  |  |  | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
|  |  |  |  |  | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| yes | yes | yes | no | yes | yes | yes | no | no | no | no | no | no | yes | no |

=> number of page faults: 7

**Forn chữ : Consolas**

3 8 2 3 9 1 6 3 8 9 3 6 2 1 3

3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 1 | 1 |

| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

| | | | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |

| | | | | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |

yes|yes|yes|no |yes|yes|yes|no |no |no |no |no |no |yes|no |

=> number of page faults: 7

- Least Recently Used: (xét dãy bên trái)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 8 | 2 | 3 | 9 | 1 | 6 | 3 | 8 | 9 | 3 | 6 | 2 | 1 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  | 8 | 8 | 8 | 8 | 8 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  | 2 | 2 | 2 | 2 | 2 | 2 | 8 | 8 | 8 | 8 | 8 | 1 | 1 |
|  |  |  |  | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
|  |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| yes | yes | yes | no | yes | yes | yes | no | yes | no | no | no | yes | yes | no |

=> number of page faults: 9

- FIFO

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 8 | 2 | 3 | 9 | 1 | 6 | 3 | 8 | 9 | 3 | 6 | 2 | 1 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  | 8 | 8 | 8 | 8 | 8 | 8 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  | 2 | 2 | 2 | 2 | 2 | 2 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
|  |  |  |  | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 2 | 2 | 2 |
|  |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| yes | yes | yes | no | yes | yes | yes | yes | yes | no | no | no | yes | no | no |

=> number of page faults: 9

(b) Which algorithm is efficient for this problem? **(2 Points)**

Optimal is efficient for this problem because the result of number of page faults is fewer than Least Recently Used.

**Question 3 (25 Points).** Suppose, the disk drive has 100 cylinders, numbered 0 to 99. The drive is currently serving a request at cylinder number 49. Calculate the time required to satisfy the following pending disk requests if it takes 2ns/head movements using SSTF disk scheduling algorithm.

20, 34, 12, 50, 24, 44, 65, 78, 33.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 12 | 20 | 24 | 33 | 34 | 44 | 49 | 50 | 65 | 78 | 99 |
|  |  |  |  |  |  |  | (0) |  |  |  |  |
|  |  |  |  |  |  |  |  | (0) |  |  |  |
|  |  |  |  |  |  | (0) |  |  |  |  |  |
|  |  |  |  |  | (0) |  |  |  |  |  |  |
|  |  |  |  | (0) |  |  |  |  |  |  |  |
|  |  |  | (0) |  |  |  |  |  |  |  |  |
|  |  | (0) |  |  |  |  |  |  |  |  |  |
|  | (0) |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | (0) |  |  |
|  |  |  |  |  |  |  |  |  |  | (0) |  |

|  |  |
| --- | --- |
| **Move** | **Distance** |
| 49 -> 50 | 50 – 49 = 1 |
| 50 -> 44 | 50 – 44 = 6 |
| 44 -> 34 | 44 – 34 = 10 |
| 34 -> 33 | 34 – 33 = 1 |
| 33 -> 24 | 33 – 24 = 9 |
| 24 -> 20 | 24 – 20 = 4 |
| 20 -> 12 | 20 – 12 = 8 |
| 12 -> 65 | 65 – 12 = 53 |
| 65 -> 78 | 78 – 65 = 13 |

=> Total seek time = (1 + 6 + 10 + 1 + 9 + 4 + 8 + 53 + 13) \* 2 = 105 \* 2 = 210 ms

**Question 4 (20 Points)**

(a) Consider a page size of 512 bytes. Find the physical address if the logical address is

i) 588 and ii) 1124. Consider the following page table. **(10 Points)**

|  |  |
| --- | --- |
| **Page Number** | **Frame** |
| 0 | 5 |
| 1 | 4 |
| 2 | 6 |
| 3 | 1 |

|  |
| --- |
| Page Number = logical address / page size (chia lấy nguyên)  Offset = logical address % page size (chia lấy dư)  Physical Address = (frame number \* page size) + offset |

- logical address : 588

Page Number = 588 / 512= 1

Offset = 588 % 512= 76

Physical Address = (4 \* 512) + 76 = 2124

- logical address : 1124

Page Number = 1124 / 512= 2

Offset = 1124 % 512= 100

Physical Address = (6 \* 512) + 100 = 3172

(b) A **counting semaphore** S is initialized to 7. Then, 20 P operations and 15 V operations are performed on S. What is the final value of S? **(5 Points)**

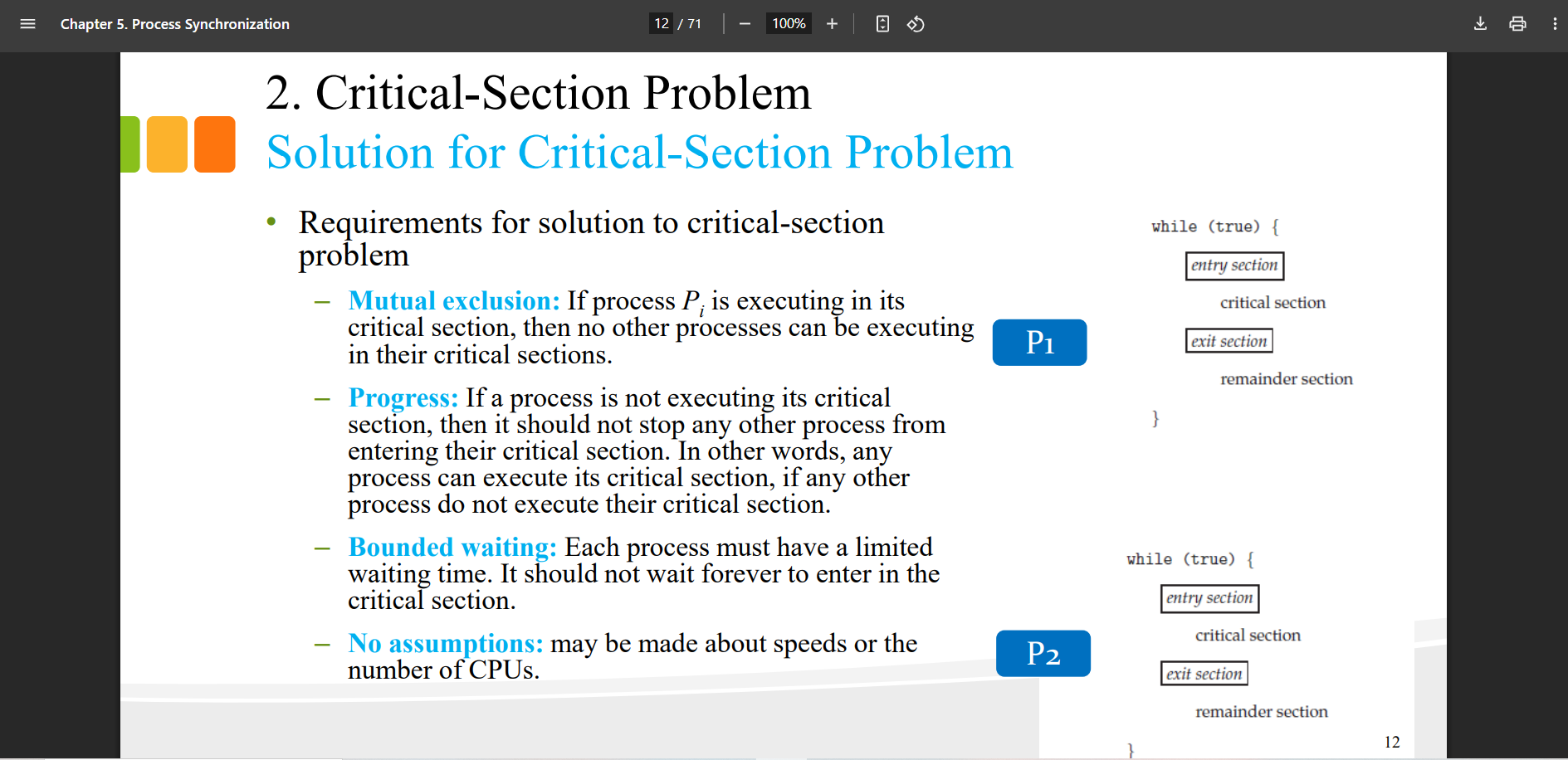
=> S = 2

P: 7 – 20 = -13

V: -13 + 15 = 2

(c) Does the following code supports Mutual Exclusion and Progress or not? Let's assume, turn is a Boolean variable randomly initialized. **(5 Points)**

|  |  |
| --- | --- |
| **Process P1** | **Process P2** |
| Entry Section: while(turn == 1);  (critical section);  Exist Section: turn=0; | Entry Section: while(turn == 0);  (critical section);  Exist Section: turn=1; |



- Mutual Exclusion:

=> Only one process is allowed in the critical section at any time so it supports Mutual Exclusion.

- Progress:

=> it not supports Progress because:

When turn =0 , while(turn ==1) false -> do critical section and turn =0 -> in process 2, while(turn ==0) it wil loop indefinitely.