**Department of Computer Science and Engineering**

**A**

**Final Examination Spring 2023**

**CSE 321: Operating Systems**

|  |  |
| --- | --- |
| **Duration:** 2 Hours | **Total Marks:** 40 |

Answer the following questions.

Figures in the right margin indicate marks.

|  |  |  |
| --- | --- | --- |
| **1.**  **CO4** | **a)** In a restaurant, there are 3 washrooms available for male guests and 1 washroom available for female guests. There is a strict rule maintained by the authorities that neither men can use the female washroom nor women can use any of the male washrooms. On a random day during lunch time, 9 male guests needed to use washroom facilities at the very same time. But none of them were allowed to use the female washroom although it was vacant. Therefore, only 3 guests were able to get access to washrooms at a time and others had to wait while maintaining a queue. If any of the washrooms gets vacant, a person from the queue can get access to that. Logically **explain** which synchronization method has been used here. | [3] |
|  | **b)** For Peterson’s problem below conditions will be applied.   * There are two processes: P1 and P2. * Each Statement takes 4ms to execute, P1 gets executed first * Context Switch will occur after 16ms. * Critical section contains 4 statements. * Remainder section contains 2 statements. * For P1: i = 0 and j = 1 * For P2: i = 1 and j = 0 * turn=0 * flag[0] = FALSE, flag[1] = TRUE   **The structure of process Pi in Peterson’s solution:**   |  | | --- | | **do{**  **flag[i] = true;**  **turn = j;**  **while(flag[j] == true && turn == 1){**  **//busy wait**  **}**  **//critical section**  **flag[i] = false;**  **//remainder section**  **}while(true);** | |  |
|  | **Complete** the table given below for processes P1 and P2 using **Peterson’s solution.**   |  |  | | --- | --- | | Process 1: i = 0, j = 1 | Process 2: i = 1, j = 0 | |  |  | |  |  | |  |  | |  |  | |  |  | | [4] |
| **2.**  **CO4** | **a)** Consider the following snapshot of a system:   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **Allocation** | | |  | **MAX** | | |  | **Available** | | | |  |  | **A** | **B** | **C** |  | **A** | **B** | **C** |  | **A** | **B** | **C** | | **P0** |  | 0 | 3 | 3 |  | 10 | 10 | 9 |  | 8 | 5 | 6 | | **P1** |  | 4 | 2 | 3 |  | 5 | 9 | 8 |  |  |  |  | | **P2** |  | 4 | 4 | 4 |  | 9 | 6 | 7 |  |  |  |  | | **P3** |  | 5 | 4 | 4 |  | 5 | 5 | 8 |  |  |  |  | | **P4** |  | 0 | 3 | 3 |  | 10 | 9 | 7 |  |  |  |  | |  |
|  | 1. Is the system in a safe state? **Apply Banker’s safety algorithm** to find out the safe sequence. You need to calculate the need matrix. | [4] |
|  | 1. **P0** requests for **(1 4 1),** **check** the validity of the request. If the request is valid, does the system enter a **deadlock**? | [1+4] |
|  | **b)** Suppose, in a workplace, we have a set of resource types, R = {R1, R2, R3, R4} and a set of processes, P = {P1, P2, P3, P4, P5}. **R1, R2, R3, and R4** have **2, 3, 1, and 2** instances respectively.   * P1 is holding 1 instance of R4 and 1 instance of R1 * P1 requests 1 instance of R2 * P2 requests for 1 instance of R4 * P2 is holding 2 instances of R2 and 1 instance of R3 * P3 requests for 1 instance of R1 * P3 is holding 1 instance of R2 * P4 is holding 1 instance of R3 * P4 requests 2 instances of R2 * P5 is holding 1 instance of R1 * P5 requests 1 instance of R4 |  |
|  | **Construct** a resource allocation graph for the above scenario. **Mention** the number of cycles found and **identify** whether there is a deadlock or not. | [4] |
| **3.**  **CO5** | **a)** Given fixed size memory partitions of 300k, 480k, 110k, 200k, 360k, and 550k **(in order, bottom to top**), **apply** first-fit and best-fit algorithms to place processes with the space requirement of 426k, 300k, 125k, 104k, 475 and 340k (in order). Which algorithm makes the most effective use of memory? Is there any external fragmentation? | [4+1+1] |
|  | **b)** Considering in Dynamic memory management technique at a certain time the memory looks like the following figure:    Where gray portions of the memory are free spaces. A process P5 = 300K arrives in the ready queue. How can you accommodate P5 in the memory? **Justify** your answer. | [2] |
|  | **c)** If the page size is **3 KB**, **how many** frames will be needed in Main memory for a process size of **32,167 Bytes**? Is there any **internal fragmentation**? - If yes, **calculate** the value. [1 KB = 1024 Bytes] | [2] |
| **4.**  **CO5** | **a)** Suppose there are two processes **P1 (16 B)** and **P2 (12 B)** with a page size of **4 B**. The main memory size is **40 B.** The page table for P1 and P2 are given below:   |  |  |  | | --- | --- | --- | | Page# | Frame# | Contents | | 0 | 8 | CSE110 | | 1 | 0 | CSE111 | | 2 | 3 | CSE220 |   Page table: P2   |  |  |  | | --- | --- | --- | | Page# | Frame# | Contents | | 0 | 7 | CSE110 | | 1 | 4 | CSE111 | | 2 | 6 | CSE221 | | 3 | 5 | CSE321 |   Page table: P1 |  |
|  | 1. **Draw** the memory representations consisting of the contents for both P1 and P2. | [2] |
|  | 1. **Find** the corresponding physical addresses of the following logical addresses: 2. Address 1010 of P1 3. Address 0111 of P2 | [2] |
|  | 1. **How** can you make efficient use of the main memory in this scenario? | [1] |
|  | **b)** Consider a computer with a main memory that has 4 frames and page reference string of 0-7 pages: **[2, 5, 7, 5, 2, 0, 2, 5, 6, 2, 6, 6]**. The page reference string represents the order in which the pages are accessed by a program. **Apply FIFO** & **OPT** algorithm to **simulate** the page replacement that occurs when the main memory can hold at most 4 pages at a time. **Record** the number of **page faults** and compare the result. **Mention** which algorithm performs better in this scenario. | [4+1] |