**Department of Computer Science and Engineering**

**Final Examination Fall 2022**

**CSE 321: Operating Systems**

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

Answer the following questions.

Figures in the right margin indicate marks.

| **1.**  **CO4** | **a)** A system has processes to execute of which **50%** is parallel. If the number of cores is increased from **1** to **4**, what will be the **increase** in performance?  **b) Distinguish** between **many-to-many** and **many-to-one** multithreading models. | [2]  [2] |
| --- | --- | --- |
| **2.**  **CO5** | **a)** Suppose a faculty member can take a maximum of 5 groups of students for doing a thesis under him in a semester. In a particular semester, a total of 9 groups applied for doing a thesis under his supervision. Among them, he selected 5 groups and kept the rest of the other groups on a waiting list for the next semester where groups will be selected according to a first come first serve manner from the waiting list if any of his slots gets free. **Logically explain** which synchronization method has been used here.  **b)** For Peterson’s problem below conditions will be applied.   * There are two processes: P1 and P2. * Each Statement takes 3ms to execute. * Context Switch will occur after 9ms. * Both the Critical & Remainder section contains 3 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 | | --- | --- | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | | [2]  [4] |
| **3.**  **CO5** | **a)** Consider the following snapshot of a system:   |  |  | **Allocation** | | | |  | **MAX** | | | | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **A** | **B** | **C** | **D** |  | **A** | **B** | **C** | **D** | | **P0** |  | 5 | 1 | 1 | 4 |  | 10 | 5 | 10 | 5 | | **P1** |  | 2 | 2 | 6 | 2 |  | 5 | 9 | 10 | 3 | | **P2** |  | 2 | 8 | 6 | 4 |  | 9 | 12 | 9 | 10 | | **P3** |  | 4 | 6 | 8 | 2 |  | 4 | 6 | 8 | 3 | |  |  |  |  |  |  |  |  |  |  |  | |  |  | **Available** | | | |  |  |  |  |  | |  |  | 1 | 1 | 0 | 3 |  |  |  |  |  |  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. 2. **P1** requests for **(1 0 0 1)**. **Check** the validity of the request. If the request is valid, does the system enter a **deadlock**?   **b)** Suppose, in an office, we have a set of resource types, R = {R1, R2, R3} and a set of processes, P = {P1, P2, P3, P4, P5}. **R1, R2, and R3** have **3, 2, and 1** instance respectively.   * P1 requests for 1 instance of R3 * P1 is holding 2 instances of R2 * P2 requests for 1 instance of R2 * P2 is holding 1 instance of R1 * P3 is holding 1 instance of R3 * P3 requests 1 instance of R1 * P4 is holding 1 instance of R1 * P5 is holding 1 instance of R1 * P5 requests 1 instance of R2   **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]  [2+3]  [3] |
| **4.**  **CO6** | **a)** At a particular time, the snapshot of the Main memory is given below for dynamic partitioning where gray portions of the memory represent occupied spaces. **Apply** **worst fit** and **first fit** algorithms to place processes with the space requirement of P1=300k, P2=200k, P3=149k, P4=146k, P5=100k, P6= 50k, P7=22k and P8=29k (in order). **Explain** which algorithm makes the most effective use of memory.    **b)** Assume that the page size is 3 bytes and the Physical Memory size is 36 bytes. **Show** the users’ view of memory which is mapped into physical memory.    **c)** If the page size is **2 KB**, **how many** frames will be needed in Main memory for a process size of **38,767 Bytes**? Is there any **internal fragmentation**? - If yes, **calculate** the value. [1 KB = 1024 Bytes] | [2+2+1]  [2]  [2] |
|  | **d)** What are the **differences** between **static** and **dynamic** techniques for partitioning main memory? | [3] |
| **5.**  **CO6** | Consider a computer with a main memory that has 3 frames and page reference string of 0-7 pages: **[7 0 6 4 2 6 1 2 0 5 1]**. The page reference string represents the order in which the pages are accessed by a program. **Apply FIFO** & **LRU** algorithm to **simulate** the page replacement that occurs when the main memory can hold at most 3 pages at a time. **Record** the number of **page faults** and compare the result. **Mention** which algorithm performs better in this scenario. | [4+1+1] |