EAST WEST UNIVERSITY
Department of Computer Science and Engineering
B.Sc. in Computer Science and Engineering Program
Final Exam,

Course: **CSE 325 (Operating Systems), Section - 2**
Instructor: **Yeasir Rayhan, Lecturer, CSE**
Full Marks: **35 (35 will be counted for final grading)**
Time: **1 Hour and 45 Minutes for Exam + 10 Minutes for Submission**

Note: There are **SEVEN** questions, answer **ALL** of them. Course Outcome (CO), Cognitive Level and Mark of each question are mentioned at the right margin.

---

1. A system has four processes and five allocatable resources. The current allcoation and maximum needs are as follows. [CO2,C3, Mark: 5]
Find the smallest value of x for which this is a safe state.

	Allocated					Maximum					Available				
Process A	1	0	2	1	1	1	1	2	1	3	0	0	x	1	1
Process B	2	0	1	1	0	2	2	2	1	0					
Process C	1	1	0	1	0	2	1	3	1	0					
Process D	1	1	1	1	0	1	1	2	2	1					

2. There are 6 processes PO through P5 and 4 resource types: A (15 instances), B (6 instances), C (9 instances), D (10 instances). [CO2,C3, Mark: 5]
For the following current allocation and maximum need, determine whether a new request [3, 2, 3, 3] from P5 should be granted?

	Current Allocation				Maximum Need			
	A	B	C	D	A	B	C	D
P0	2	0	2	1	9	5	5	5
P1	0	1	1	1	2	2	3	3
P2	4	1	0	2	7	5	4	4
P3	1	0	0	1	3	3	3	2
P4	1	1	0	0	5	2	2	1
P5	1	0	1	1	4	4	4	4

3. *`Search-Insert-Delete`* problem: Three kinds of threads share access to a file: `Searchers`, `Inserters` and `Deleters`. `Searchers` merely examine the list; hence they can execute concurrently with each other. `Inserters` add new items to the end of the list; insertions must be mutually exclusive to preclude two `inserters` from inserting new items at about the same time. However, one insert can proceed in parallel with any number of searches. Finally, `deleters` remove items from anywhere in the list. At most one `Deleter` thread can access the list at a time, and deletion must also be mutually exlusive with searches and insertions. [CO2,C3, Mark: 5]

Write pseudocode for Searchers, Inserters and Deleters that enforce this kind of three way mutual exclusion above using semaphore and/or mutexes. Design your soulution in a way that it does not resutl in starvation.

4. Consider the following C++ code that forms part of a course registration system that keeps track of the students and courses in memory. (set is a container class provided by the standard library which stores a set of values) [CO2,C3, Mark: 5]

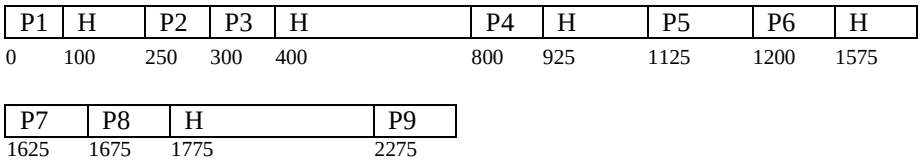
```
struct Course {
        pthread_mutex_t lock; string name; set<Student*> enrolled;
};
struct Student {
        pthread_mutex_t lock; string name; set<Course*> courses;
};

/*Removes a sinle course from a single student's
list of courses*/
void RemoveStudentFromCourse(Student *student, Course *course) {
        pthread_mutex_lock(&student->lock);
        pthread_mutex_lock(&course->lock);
        course->enrolled.erase(student);
        student->courses.erase(course);
        pthread_mutex_unlock(&course->lock);
        pthread_mutex_unlock(&student->lock);
}

/*Transfers all the studens from one course to
another course */
void TransferStudents(Course *from_course, Course *to_course) {
        pthread_mutex_lock(&from_course->lock);
        for (Student *student : from_course->enrolled) {
            pthread_mutex_lock(&student->lock) ;
            pthread_mutex_lock(&to_course->lock) ;
            student->courses.erase(from_course) ;
            to_course->enrolled.insert(student) ;
            student->courses.insert(to_course) ;
            pthread_mutex_unlock(&to_course->lock) ;
            pthread_mutex_unlock(&student->lock) ;
        }
        from_course->enrolled.clear();
        pthread_mutex_unlock(&from_course->lock) ;
}
```

If `RemoveStudentFromCourse` and `TransferStudents` are called in parallel from two different threads, then deadlock can occur.

(a) **Give** an example of the arguments to such parallel calls and identify which locks or other resources each thread could be waiting for when the deadlock occurs.

(b) **Propose** a change to `RemoveStudentFromCourse` and/or `TransferStudents` that will fix this deadlock. Avoid harming the functionality and efficiency of the functions with your change.
Do not rewrite the full function / functions. Just mention the changes that you would do.

5. A system holds memory as shown in the figure below. (P represents processes and H represents Holes. The corresponding starting address of a block is given below the block, e.g., the starting address of P3 in memory = 800) [CO3,C3, Mark: 5]



Consider the following sequence of process arrival and termination:

i) P9 (175 bytes) comes
ii) P10 (100 bytes) comes
iii) P3 leaves
iv) P12 (525 bytes) comes

Show memory representation after each step in first fit and best fit algorithm. Make sure you specify the starting address of the block which have been allocated for the incoming process and the starting address of the newly created holes if there are any.

6. A logical memory is addressed using 16 bits and the corresponding physical memory is addressed with 19 bits. The logical memory currently stores 9 pages. However, it can store upto 16 pages and the page or frame size is measured in bytes. [CO3,C3, Mark: 5]

i) **Determine** the size of the physical and virtual memory.
ii) **Determine** the number of frames the physical memory holds.
iii) Based on the page table given below **find** the corresponding physical address reference of the following logical address references (given in decimal). If a virtual page can not be mapped to a physical frame just put "Can not be mapped"

2354, 38947, 4260, 58902, 6698, 25600

Page number	Frame number
0	125
14	51
2	120
13	19
8	2
12	84
6	36
9	77

7. One problem with contiguous allocation is that the user must preallocate enough space for each file. If the file grows to be larger than the space allocated for it, special actions must be taken. One solution to this problem is to define a file structure consisting of an initial contiguous area (of a specified size). If this area is filled, the operating system automatically defines an overflow area that is linked to the initial contiguous area. If the overflow area is filled, another [CO3,C3, Mark: 5]

overflow area is allocated. Compare this implementation of a file with the standard contiguous and linked implementations