National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Advance Operating Systems	Course Code:	CS-505
	Program:	MS(Computer Sciences)	Semester:	Spring 2020
	Duration:	180+15 minutes	Total Marks:	50
	Paper Date:	8th July, 2020	Weight:	50%
	Section:	ALL	Page(s):	3
	Exam:	Final	Roll No.	

Instructions/Notes:

- Read all the questions carefully, all questions are compulsory.

- Follow the instructions given in each question, not following the instructions will result in the deduction of marks.

- Submit the solution at Learning Management System (LMS). Here LMS is Google Classroom, and it is the only preferred submission mechanism.

- Submission should be in the form of a SINGLE PDF file.

- Above mentioned 15 minutes are included only for the submission. The exam time is 180 minutes.

- Write your answers on a paper clearly so the scanned copies are clear and legible (readable).

- Readability is the responsibility of the student, if an answer is not readable, then it will be marked zero and the responsibility is upon the student.

- While creating the PDF put all the images in order, meaning answer of one question should be on contiguous pages.

- Write the page number with your own hand on each page of your answer book.

- No rough work should be included in the answer book. All pages uploaded to the LMS will be considered a valid answer.

- If your unintelligible rough work appears in the answer book along with the real answer, then the question will be marked zero.

- Duplicate answers to the same question will be marked zero.

- Whenever a numerical is under question you have to elaborate the steps taken to calculate the answer to the question.

- Re-submission is allowed, but only the last submission will be counted. Previous submissions will be discarded and will not be considered.

- **Reading the above guidelines is mandatory, not reading does not exonerate a student of any repercussions.**

Question 1 (10 points): We have studied distributed algorithm for mutual exclusion. Suppose you have four processes, namely 0, 1, 2, 3. You have to draw diagrams showing the state of the system after following actions. Draw a separate diagram for each action, clearly mentioning the question number. For example, for diagram related to part (1) you should prominently write (1.1) and then draw a diagram of step 1 of part 1. Then write (1.2) and draw a diagram of step 2 of part 1. Similarly, for all other parts.

- Process 0 tries to enter a critical section at time 7 and sends the relevant messages.

- Process 3 tries to enter into the same critical section and sends messages at time 13.

- The process which enters first, exits the critical section at time 25 and notifies the relevant process(es).

- Show the sequence of messages after the above action, highlighting which process may enter into the critical section now.

Question 2 (10 points): Given is a class (RT_task)representing a real time task. Implement the earliest deadline first with unforced idle time with the help of the declarations given below. You only have to return one task which should be selected among all tasks given to you in a vector named **jobs**.

```cpp
class RT_Task{
public:
        int priority;
        int arrival_time;
        int execution_time;
        int end_deadline;
};
// Implement the following function
RT_Task* EDFWUIT(std::vector<RT_Task> jobs, int curr_time) // pointers to all tasks are
    already in the list named "jobs". Sedcond parameter gives the current time.
{

}
```

Question 3 (10 points): According to Bully algorithm for leader election. Show using the diagram how a group selects its leader. The scenario is given as follows. Draw a separate diagram for each step written below, clearly mentioning the question number. For example, for diagram related to part (4) you should prominently write (4.1) and then draw a diagram of first step, then write (4.2) to draw a diagram of second step. Similarly, for all other parts.

1. There are five processes in the federation, numbered 0 through 4, highest numbered process is leader.

2. The leader crashes suddenly

3. Process numbered 1 realize this and starts an election.

4. Draw a diagrams of each step showing how the election procedure will proceed

5. Mention clearly who will be the leader at the end.

Question 4 (10 points): In the following table, two transactions are given with their operations. You can see that the result of the transaction does not remain the same when we change the order of execution. Suppose that transaction 1 is bound to run before transaction 2. We want to retain the same result that is produced by the sequential execution of the transactions i.e. first transaction 1, then transaction 2. Write three more **unique** schedules that retain the result of the transaction. By this it is meant that at the end of each schedule the values of x and y should be the same as of the end of sequential schedule. After writing the schedules, also write the intention list of each schedule you mentioned.
NOTE: You cannot rename the variables. The unique schedules should be literally unique, meaning there must be some visible difference in a schedule from all others.

Transaction 1		Transaction 2	
1	`BEGIN TRANSACTION`	1	`BEGIN TRANSACTION`
2	`y = 1`	2	`y = 1`
3	`x = 1`	3	`x = 1`
4	`y = 2 + y`	4	`x = 2 + x`
5	`END TRANSACTION`	5	`END TRANSACTION`

Sequential Schedule	y=1	x=1	y=2+y	y=1	x=1	x=2+x

Question 5 (10 points): There are four processes P_1, P_2, P_3 and P_4. Their start time is in order, meaning P_1 starts first and P_4 starts last. Following are the actions performed by the processes. Show them using a resource allocation graphs at each step. Secondly, keep in mind that we have implemented **wait and die** algorithm in our system, so be careful that some processes are allowed to wait and others are not. The action needed as suggested by **wait and die** algorithm should be shown in the diagrams.
Show every action of each step in a separate diagram.

1. P_1 acquires resource R_1.

2. P_2 acquires resource R_2.

3. P_3 acquires resource R_3.

4. P_4 requests resource R_1.

5. P_1 requests resource R_3.

6. P_3 requests resource R_2.

7. Update the final condition of the resource allocation graph.