National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Advanced Operating System	Course Code:	CS-505
	Program:	MS(Computer Science)	Semester:	Fall 2016
	Duration:	60	Total Marks:	35
	Paper Date:	October 17, 2016	Weight:	15%
	Section:	A	Page(s):	2
	Exam:	Midterm		

Instructions/Notes: You can use extra sheet for rough work. Only write on this sheet when you are sure, cutting on this sheet will deduct your marks.

NAME:____ ROLL#:______________________

Question 1 (10 points): Given the table, the highest priority task should run before the lower one. For same priority tasks use a scheme which gives the minimum waiting time. Write the **order of execution** and calculate the **waiting time** in the end. Give **reason** (or proof), why under the given conditions, is it impossible to achieve lower waiting time than your scheme?

Process ID	Time of Arrival	Priority	Computation Time
1	20	1	10
2	10	1	20
3	0	1	30
4	20	2	100
5	20	2	10
6	20	2	20
7	30	3	30
8	30	3	60
9	30	3	20
10	40	4	10
11	40	4	50

Question 2 (7 points): Write the steps performed by the kernel to switch two processes.

•

•

•

•

•

•

•

Question 3 (8 points): Suppose that each memory page in our operating system has a size of 200 bytes. The frame size is also equal to the page size. Provide the physical addresses of the bytes, given their logical addresses as under. The table given below shows which page is loaded onto which frame. (**Note:** recall that although the logical address here is two dimensional, the physcial address is always one dimensional.)

(a) (3,30) = ______ (c) (8,0) = ______

(b) (6,10) = ______ (d) (7,23) = ______

Page #	1	2	3	4	5	6	7	8	9	10	11	12	13
Frame #	5	9	13	2	8	11	7	3	6	4	12	10	1

Question 4 (10 points): Three processes are running in parallel sharing variables i and j, which are initialized as $i = 0$ and $j = 0$. You have to synchronize the following processes such that the output on the console is a continuous string of even integers as $2, 4, 6, 8, \ldots$

NOTE: You can change the following code in only one manner, that is, calling the **wait** and **signal** methods of semaphores. You are allowed to use as many semaphores as you like. No other change in the code is allowed. Also, you have to mention the initial values of all the semaphores.

Process 1	Process 2	Process 3

`while(true)`
`{`
 	
`j++;`
 	
`}`
	`while(true)`
	`{`
	
	`i++;`
	
	`}`
		`while(true)`
		`{`
		
		`cout << i+j<<",";`
		
		`}`