National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Operating System	Course Code:	CS-205
	Program:	BS(Computer Science)	Semester:	Fall 2016
	Duration:	$2\frac{1}{2}$ hours	Total Marks:	75
	Paper Date:	22 December, 2016	Weight:	45%
	Section:	A-B	Page(s):	2
	Exam:	Final	Roll No.	

Instructions/Notes: Fill the answers in the first 10 pages of your answer sheet.

Question 1 (5 points): What is the purpose of command line interpreter? Why is it not made part of the kernel? give the most important reason.

Question 2 (5 points): What is micro kernel architecture? List at least **three** benefits.

Question 3 (5 points): In Linux the open-file table is used to maintain information about files that are currently open. Should the operating system maintain a separate table for each user or maintain just one table that contains references to files that are currently being accessed by all users? If two users try to access the same file, how operating system manages the situation?

Question 4 (10 points): There are three processes running on a machine, call them P_1, P_2 and P_3. There are three resources on that machine, call them R_1, R_2 and R_3. The processes are scheduled according to Round Robin scheduling, where the quantum size is Q. Each process runs for 15 quanta and then terminates. Each process tries to acquire *one* resource in each quantum, in following manner.

- P_1: R_1,$R_2$$R_3$
- P_2: R_2,$R_3$$R_1$
- P_3: R_3,$R_1$$R_2$

For example, process P_1 tries to acquire resource R_1 in first quantum, R_2 in second quantum and R_3 in third. While process P_2 acquires R_2 in first quantum, R_3 in second quantum and R_1 in third quantum. Similarly process P_3 starts from R_3 and proceeds to R_1 and R_2. No process releases the resource until its last quantum (quantum number 15). Explain, will there be any deadlock or not? If no, then why? If yes, then when the deadlock will occur? If the deadlock occurs, then propose a technique which prevents the deadlock. Draw diagrams for help.

Question 5 (10 points): Custom File System (CFS) uses indexed allocation for the files. Indexing is done at a single level. You have to implement a function **fetchByte()** to fetch a byte of a file from CFS. The block size in CFS is **P** bytes. The implementation may only contain the pseudo code. The parameters of the function are following

1. Number of the block where the index is stored, call it **INB**.
2. Logical byte number of the file whose data we have to fetch, call it **BN**.

Question 6 (10 points): Following is an inverted page table on a machine. A process whose ID is 1 accesses the following byte numbers, translate these addresses into physical addresses. The size of one page is **1000** bytes.

- 2345
- 87
- 4315
- 123

PID	Page #	Frame #
1	2	12
2	5	4
3	3	6
1	4	7
2	3	2
2	7	9
4	1	13
1	0	15
2	6	10

Question 7 (10 points): Suppose there is a dual core computer machine installed at a manufacturing unit. designing a custom OS for that machine. The machine has a large Translation Look Aside Buffer (TLB). The store 4×10^6 unique entries in it. The key for an entry can be made composite, for example composite key of Page number. There can be only two processes at a time running on that machine. The memory requiremen process will not exceed more than *4GB*. Memory page size on the machine is *4K* bytes.

Under the given circumstances provide the best technique of memory management which explorts the large size You have to answer the following questions **with reason**

- What will be the size of a process page table?
- Which type of page table should you use? given that you have a very large TLB.
- Should you consider to divide the page table into smaller parts or not?

Question 8 (10 points): Following is code for producer-consumer problem. There is a mistake in this code, you **identify** the mistake and then **fix** it. The best solution will be the one which only repositions a single statement, what is the problem in this code?

Producer	Consumer
sem_1=1,sem_2=0, buffer // among shared variables **buffer** is an infinite list of elements , rest are semaphor	

Producer:

```
while(true)
{
 item = produce()
 sem_1.wait()
 buffer.add(item)
→ sem_2.signal()
 sem_1.signal()
}
```

Consumer:

```
while(true)
{
 sem_1.wait()
 sem_2.wait()
 item = buffer.get()
 sem_1.signal()
 item.process()
}
```

96

44.3

Question 9 (10 points): In the above code the **buffer** object can store infinite number of elements. Modify th code so that it can work for a **buffer** which can only store *N* number of elements.

24/25
16/20
11/20
15/20

7.71/15
9.38/15

36.7m

54.93
53.3
45.31

51.21

8.
8.68
9.15
6.2