National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Operating System	Course Code:	CS-205
	Program:	BS(Computer Science)	Semester:	Fall 2018
	Duration:	1 hour	Total Marks:	40
	Paper Date:	2nd October, 2018	Weight:	15%
	Section:	All	Page(s):	4
	Exam:	Mid-1	Roll No.	

Instructions/Notes: Answer questions on the question paper. Write answers clearly and precisely, if the answers are not easily readable then it will result in deduction of marks. Use extra sheet for rough work, **cutting and blotting on this sheet will result in deduction of marks**.

Question 1 (3 points): List four main components of computer system which the kernel has to manage.

(1) **CPU management/ Process Management / Process Scheduling**

(2) **Memory Management**

(3) **Permanent Storage Management/ File System**

(4) I/O including network management

Question 2 (2 points): The CPU is connected to a ______(a)______, which is connected to the device controller, which is connected to a ______(b)______ device. This is the path of communication that enables the hardware interrupts to occur.

(a) **bus/communication bus/ PCI bus**

(b) **peripheral**

Question 3 (2 points): Which of the following scheduling algorithms is non-preemptive?

(a) Round Robin

(b) **FCFS**

(c) Shortest Remaining Time First

Question 4 (2 points): There are machine level commands written inside the CPU which, when executed, can change the values of the registers built inside the device controllers.

(a) **True**

(b) False

Question 5 (5 points): Write in each cell what type of Inter Process Communication is discussed. Tick the correct column.

Scenario	Shared Memory	Message Passing
Done correctly, sharing of information is faster using this technique	X	
Processes use `write()` and `read()` system calls		X
In some forms only one way communication is possible at one time		X
The processes must use some synchronization mechanism	X	
A queue is used and mostly the queue is controlled by the kernel		X

Question 6 (2 points): Suppose a machine runs one instruction in one clock cycle. Now suppose a **program** has 10 instructions in its instruction stream. The program is loaded into memory and becomes a "process". There is no way that the process takes more than 10 clock ticks to finish, is this correct?

(a) Yes

(b) **No**

Question 7 (4 points): Name any two methods used for parameter passing between a process and the kernel.

(a) **Stack**

(b) **Register / Register and pointer to table**

Question 8 (5 points): Tell the output of the following code. Assume that the parent process running following code has the *PID* = 100. Each new `fork()` creates a new process. Each child process gets the process ID in following way. The first digits of the child process ID are all borrowed from the parent. The last digit is equal to the number of `fork()` done by the parent. For example, if parent whose *PID* = 100, the child created in result of the first fork will have the *PID* = 1001 and the child created in result of the second fork will have *PID* = 1002.
Assume that each instruction runs in the order. Meaning instructions written on smaller line numbers will necessarily execute before the instructions written on bigger line numbers.
Hints:The function `getpid()` returns the *PID* of the calling process. Take due diligence in this question, and cross verify each step. Remember what is the out of `fork()`.

```
# include <stdio.h>
int main(void)
{
int pid=0;
pid = fork();//(1001)
if ( pid == 0)
{
        printf("%d,", getpid()); // 1001
        pid = fork();//(10011)
        if ( pid == 0)
        {
                printf("%d,", getpid()); //10011
                pid = fork(); //(100111)
                if ( pid > 0)
                {
                        printf("%d,", getpid()); //10011
                        pid = fork(); //(100112)
                        if ( pid > 0)
                        {
                                printf("%d,", getpid()); //10011
                                pid = fork(); //(100113)
                                if ( pid == 0)
                                {
                                        printf("%d,", getpid()); //100113
                                }
                        }
                }
        }

}
else if (pid > 0)
{
        printf("%d,", getpid()); //100
}
return 0;
}
```

Solution: 1001,10011,10011,10011,100113,100

Question 9 (5 points): Inspired from the above code, write a similar code using `fork()` which prints the string "100,1001,1002,1003" (without the quotes). You can make the same assumption about the execution order as above. Meaning the line written before will execute before.

```
        # include <stdio.h>
        int main(void)
        {
                int pid=0;
                printf("%d,", getpid()); // 100
                pid = fork();//(1001)
                if ( pid == 0) // or just printf("%d,", pid);
                {
                        printf("%d,", getpid()); // 1001
                }
                else if (pid > 0)
                {
                        pid = fork();//(1002)
                        if ( pid == 0) // or just printf("%d,", pid);
                        {
                                printf("%d,", getpid()); // 1002
                        }
                        else if (pid > 0)
                        {
                                pid = fork();//(1003)
                                if ( pid == 0) // or just printf("%d,", pid);
                                {
                                        printf("%d,", getpid()); // 1003
                                }
                        }
                }
        }
```

Question 10 (10 points): Suppose in a machine the CPU executes one instruction per clock cycle. There are three Ethernet cards in the machine. Each machine runs some CPU cycle then reads data from any of the Ethernet cards. The processes arrive in order, i.e. P_1 then P_2 and then P_3

- Using the FCFS algorithm list down the order of execution of the processes.
- Calculate the total time of execution of all processes.
- Calculate the average waiting time.

Process Name	Length	CPU Burst	I/O Burst
P_1	3	Yes	No
P_2	6	Yes	No
P_3	8	Yes	No
P_1	12	No	Yes
P_2	4	No	Yes
P_3	7	No	Yes
P_1	7	Yes	No
P_2	5	Yes	No
P_3	3	Yes	No
P_1	13	No	Yes
P_2	10	No	Yes
P_3	7	No	Yes
P_1	3	Yes	No
P_2	25	Yes	No
P_3	12	Yes	No
P_1	17	No	Yes
P_2	15	No	Yes
P_3	8	No	Yes
P_1	3	Yes	No
P_2	3	Yes	No
P_3	3	Yes	No

Solutions:

Order of Processes: P_1,P_2,P_3, P_2,P_1,P_3, P_2,P_3,P_1, P_2,P_3,P_1

Total Execution Time: Through the help of Gantt Chart 92

Average Waiting Time: $\frac{0+3+9+7+4+5+27+0+18+0+0+0}{12} = \frac{73}{12}$